Green House software

1.0

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

- AlarmSensors . . . . . . **??**
- GreenHouse . . . . . . **??**
- Hardware . . . . . . **??**
  - KeyboardHardware . . . . . . **??**
  - ScreenHardware . . . . . . **??**
  - SwitchHardware . . . . . . **??**
- MonitoringSystem . . . . . . **??**
- std::runtime_error
  - FileCloseError . . . . . . **??**
  - FileCorruptError . . . . . . **??**
  - FileLockError . . . . . . **??**
  - FileNotFoundError . . . . . . **??**
  - FileOpenError . . . . . . **??**
  - FilePermissionError . . . . . . **??**
  - FileReadError . . . . . . **??**
  - FileWriteError . . . . . . **??**
- Sensor . . . . . . **??**
  - AirQualitySensor . . . . . . **??**
  - HydrometerSensor . . . . . . **??**
  - LightSensor . . . . . . **??**
  - PhSensor . . . . . . **??**
  - PressureSensor . . . . . . **??**
  - TemperatureSensor . . . . . . **??**
- User . . . . . . **??**
  - UserAdmin . . . . . . **??**
  - UserEmployee . . . . . . **??**
  - UserGuest . . . . . . **??**
- UserNameComparator . . . . . . **??**
- UserPtrComparator . . . . . . **??**
- UsersDatabase . . . . . . **??**
- UsersServer . . . . . . **??**

Chapter 2

Class Index

2.1 Class List

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

AirQualitySensor . . . . . ??
AlarmSensors . . . . . ??
FileCloseError . . . . . ??
FileCorruptError . . . . . ??
FileLockError . . . . . ??
FileNotFoundError . . . . . ??
FileOpenError . . . . . ??
FilePermissionError . . . . . ??
FileReadError . . . . . ??
FileWriteError . . . . . ??
GreenHouse . . . . . ??
Hardware . . . . . ??
HydrometerSensor . . . . . ??
KeyboardHardware . . . . . ??
LightSensor . . . . . ??
MonitoringSystem . . . . . ??
PhSensor . . . . . ??
PressureSensor . . . . . ??
ScreenHardware . . . . . ??
Sensor . . . . . ??
SwitchHardware . . . . . ??
TemperatureSensor . . . . . ??
User . . . . . ??
UserAdmin . . . . . ??
UserEmployee . . . . . ??
UserGuest . . . . . ??
UserNameComparator . . . . . ??
UserPtrComparator . . . . . ??
UsersDatabase . . . . . ??
UsersServer . . . . . ??

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

src/AirQualitySensor.cpp . . . ??
src/AirQualitySensor.h
This is the class AirQualitySensor. It contains the attributes and methods of the AirQualitySensor class . . . ??
src/AlarmSensors.cpp . . . ??
src/AlarmSensors.h
This is the class AlarmSensors. It contains the attributes and methods of the AlarmSensors class ??
src/Exceptions.h
This file contains the attributes and methods of the Exceptions class . . . ??
src/GreenHouse.cpp . . . ??
src/GreenHouse.h
This is the class GreenHouse. It contains the attributes and methods of the GreenHouse class, this class is the main of the hole system . . . ??
src/Hardware.cpp . . . ??
src/Hardware.h
This is the class Hardware. It contains the attributes and methods of the Hardware class, this class is the parent of the hole hardware system . . . ??
src/HydrometerSensor.cpp . . . ??
src/HydrometerSensor.h
This is the class HydrometerSensor. It contains the attributes and methods of the HydrometerSensor class . . . ??
src/KeyboardHardware.cpp . . . ??
src/KeyboardHardware.h
This is the class KeyboardHardware. It contains the attributes and methods of the KeyboardHardware class, this class is a child of the Hardware class . . . ??
src/LightSensor.cpp . . . ??
src/LightSensor.h
This is the class LightSensor. It contains the attributes and methods of the LightSensor class . ??
src/main.cpp . . . ??
src/mainAirQualitySensor.cpp . . . ??
src/mainAlarmSensors.cpp . . . ??
src/mainGreenHouse.cpp . . . ??
src/mainHardware.cpp . . . ??
src/mainHydrometerSensor.cpp . . . ??
src/mainKeyboardHardware.cpp . . . ??

src/mainLightSensor.cpp . . . ??
src/mainMonitoringSystem.cpp . . . ??
src/mainPhSensor.cpp . . . ??
src/mainPressureSensor.cpp . . . ??
src/mainScreenHardware.cpp . . . ??
src/mainSensor.cpp . . . ??
src/mainSwitchHardware.cpp . . . ??
src/mainTemperatureSensor.cpp . . . ??
src/mainUser.cpp . . . ??
src/mainUserAdmin.cpp . . . ??
src/mainUserEmployee.cpp . . . ??
src/mainUserGuest.cpp . . . ??
src/mainUsersDatabase.cpp . . . ??
src/mainUsersServer.cpp . . . ??
src/MonitoringSystem.cpp . . . ??
src/MonitoringSystem.h
This is the class MonitoringSystem. It contains the attributes and methods of the MonitoringSystem class, this class . . . ??
src/PhSensor.cpp . . . ??
src/PhSensor.h
This is the class PhSensor. It contains the attributes and methods of the PhSensor class . . . ??
src/PressureSensor.cpp . . . ??
src/PressureSensor.h
This is the class PressureSensor. It contains the attributes and methods of the PressureSensor class . . . ??
src/ScreenHardware.cpp . . . ??
src/ScreenHardware.h
This is the class ScreenHardware. It contains the attributes and methods of the ScreenHardware class, this class is a child of the Hardware class. This class is used to display the output of the system and ask for an input before with the keyboard . . . ??
src/Sensor.cpp . . . ??
src/Sensor.h
This is the class Sensor. It contains the attributes and methods of the Sensor class . . . ??
src/SwitchHardware.cpp . . . ??
src/SwitchHardware.h
This is the class SwitchHardware. It contains the attributes and methods of the SwitchHardware class, this class is a child of the Hardware class . . . ??
src/TemperatureSensor.cpp . . . ??
src/TemperatureSensor.h
This is the class TemperatureSensor. It contains the attributes and methods of the TemperatureSensor class . . . ??
src/User.cpp . . . ??
src/User.h
This is the class User. It contains the attributes and methods of the User class . . . ??
src/UserAdmin.cpp . . . ??
src/UserAdmin.h
This is the class UserAdmin. It contains the attributes and methods of the UserAdmin class . . . ??
src/UserEmployee.cpp . . . ??
src/UserEmployee.h
This is the class UserEmployee. It contains the attributes and methods of the UserEmployee class . . . ??
src/UserGuest.cpp . . . ??
src/UserGuest.h
This is the class UserGuest. It contains the attributes and methods of the UserGuest class . . . ??
src/UsersDatabase.cpp . . . ??
src/UsersDatabase.h
This is the class UsersDatabase. It contains the attributes and methods of the UsersDatabase class . . . ??

src/UsersServer.cpp . . . . ??
src/UsersServer.h
This is the class UsersServer. It contains the attributes and methods of the UsersServer class ??

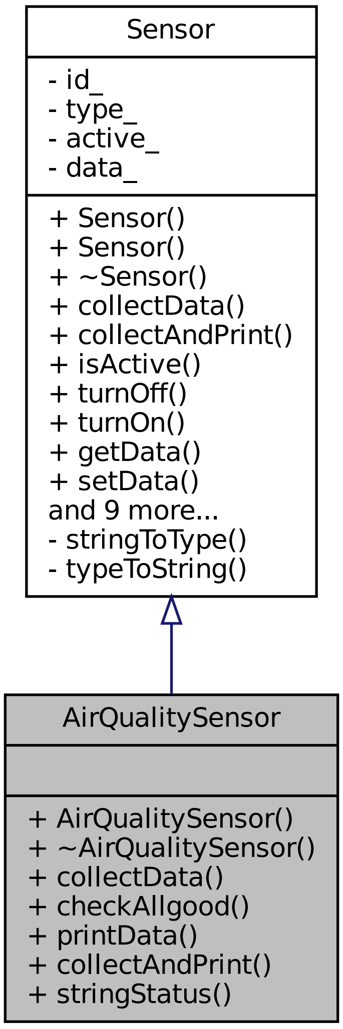
Chapter 4

Class Documentation

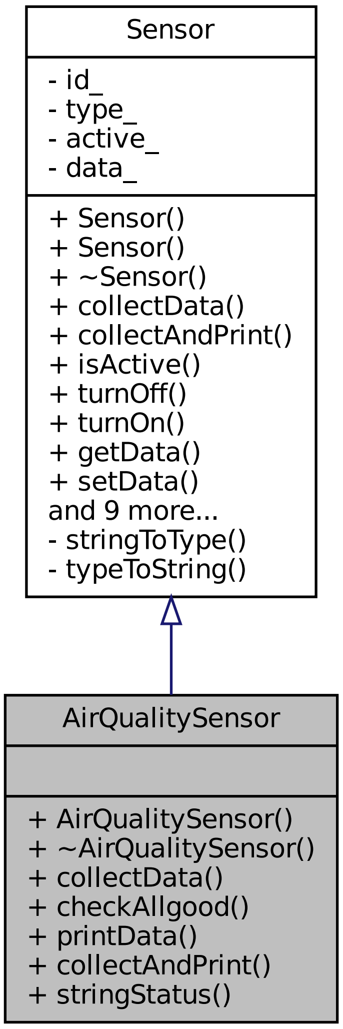
4.1 AirQualitySensor Class Reference

```
#include <AirQualitySensor.h>
```

Inheritance diagram for AirQualitySensor:



Collaboration diagram for AirQualitySensor:



Public Member Functions

- AirQualitySensor (int id, bool active)

    Construct a new Air Quality Sensor object.
- ∼AirQualitySensor () override

    Destroy the Air Quality Sensor object.
- void collectData () override

    Collect data of the Air Quality Sensor.
- bool checkAllgood () const override

    Check if the Air Quality Sensor is working properly.
- void printData () const override

    Print the data of the Air Quality Sensor.
- void collectAndPrint ()

Collect and print the data of the Air Quality Sensor.

- std::string stringStatus () const

  This method returns if the Air Quality Sensor is active or not and if its active, it returns if its good or bad the data.

Friends

- std::ostream & operator<< (std::ostream &os, const AirQualitySensor &sensor)

  This method prints the AirQualitySensor object.

Additional Inherited Members

4.1.1 Detailed Description

Definition at line 15 of file AirQualitySensor.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 AirQualitySensor()

```
AirQualitySensor::AirQualitySensor (
              int id,
              bool active )  [explicit]
```

Construct a new Air Quality Sensor object.

Parameters

id	
active	

Returns

AirQualitySensor object

Definition at line 9 of file AirQualitySensor.cpp.

```
      : Sensor(id, Sensor::Types::AIR_QUALITY, active) {}
```

4.1.2.2 ∼AirQualitySensor()

```
AirQualitySensor::~AirQualitySensor ( )  [override]
```

Destroy the Air Quality Sensor object.

Definition at line 12 of file AirQualitySensor.cpp.

```
{}
```

4.1.3 Member Function Documentation

4.1.3.1 checkAllgood()

`bool AirQualitySensor::checkAllgood ( ) const [override], [virtual]`

Check if the Air Quality Sensor is working properly.

Returns

true if the Air Quality Sensor is working properly

false if the Air Quality Sensor is not working properly

Reimplemented from Sensor.

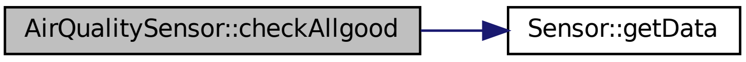
Definition at line 24 of file AirQualitySensor.cpp.

```
                                              {
    // Por debajo de 65 microgramos/m3 se considera buena calidad del aire
    float data = Sensor::getData();

    if (data <= 65) {
      return true;
    } else {
      return false;
    }
}
```

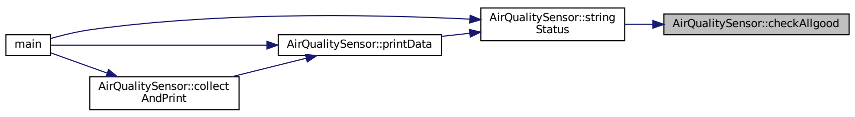
References Sensor::getData().

Referenced by stringStatus().

Here is the call graph for this function:



Here is the caller graph for this function:


main
AirQualitySensor::printData
AirQualitySensor::collect AndPrint
AirQualitySensor::string Status
AirQualitySensor::checkAllgood

4.1.3.2 collectAndPrint()

```
void AirQualitySensor::collectAndPrint ( )  [virtual]
```

Collect and print the data of the Air Quality Sensor.

Reimplemented from Sensor.

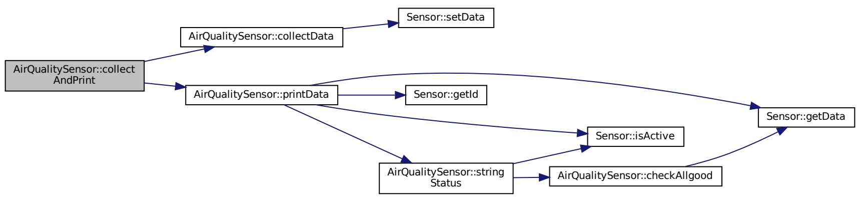
Definition at line 65 of file AirQualitySensor.cpp.

```
65                                      {
66    collectData();
67    printData();
68 }
```

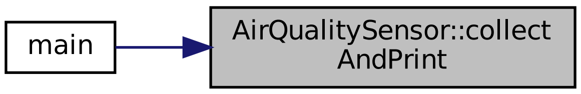
References collectData(), and printData().

Referenced by main().

Here is the call graph for this function:


AirQualitySensor::collect AndPrint
AirQualitySensor::collectData
Sensor::setData
AirQualitySensor::printData
Sensor::getId
Sensor::getData
Sensor::isActive
AirQualitySensor::string Status
AirQualitySensor::checkAllgood

Here is the caller graph for this function:



4.1.3.3 collectData()

```
void AirQualitySensor::collectData ( )  [override], [virtual]
```

Collect data of the Air Quality Sensor.

This method collects the data of the Air Quality Sensor and stores it in the data attribute.

Reimplemented from Sensor.

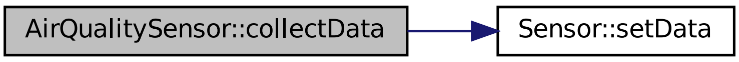
Definition at line 14 of file AirQualitySensor.cpp.

```
                                    {
    // Generamos un numero random entre 0 y 70 para simular la calidad del aire en
    // microgramos/m3
    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_int_distribution<> dis(0, 70);
    int airQuality = dis(gen);
    Sensor::setData(airQuality);
}
```

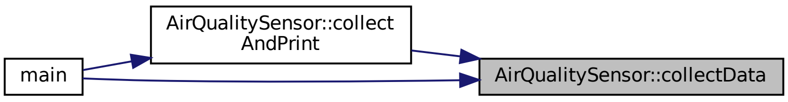
References Sensor::setData().

Referenced by collectAndPrint(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:


AirQualitySensor::collect AndPrint
main
AirQualitySensor::collectData

4.1.3.4 printData()

`void AirQualitySensor::printData ( ) const [override], [virtual]`

Print the data of the Air Quality Sensor.

Reimplemented from Sensor.

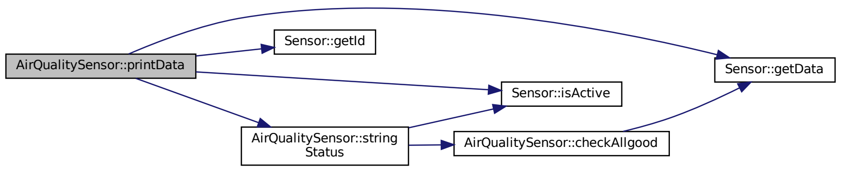
Definition at line 52 of file AirQualitySensor.cpp.

```
                                        {
    // Imprimimos las particulas por microgramo/m3, el id del sensor, y si todo
    // esta bien o no
    if (Sensor::isActive()) {
      std::cout « "Air Quality Sensor with "
                « "ID: " « Sensor::getId() « " - Data: " « Sensor::getData()
                « " microgram/m3 - Status: " « stringStatus() « std::endl;
    } else {
      std::cout « "Air Quality Sensor ID: " « Sensor::getId() « " - INACTIVE"
                « std::endl;
    }
}
```

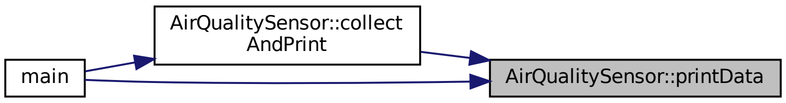
References Sensor::getData(), Sensor::getId(), Sensor::isActive(), and stringStatus().

Referenced by collectAndPrint(), and main().

Here is the call graph for this function:


AirQualitySensor::printData
Sensor::getId
Sensor::isActive
Sensor::getData
AirQualitySensor::string Status
AirQualitySensor::checkAllgood

Here is the caller graph for this function:


main
AirQualitySensor::collect AndPrint
AirQualitySensor::printData

4.1.3.5 stringStatus()

```
std::string AirQualitySensor::stringStatus ( ) const
```

This method returns if the Air Quality Sensor is active or not and if its active, it returns if its good or bad the data.

Returns

std::string

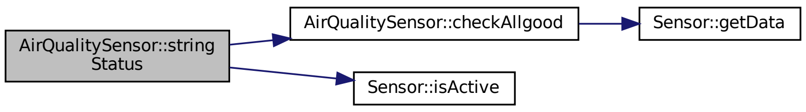
Definition at line 40 of file AirQualitySensor.cpp.

```
                                                {
  if (Sensor::isActive()) {
    if (this->checkAllgood()) {
      return "ACTIVE - GOOD STATUS";
    } else {
      return "ACTIVE - BAD STATUS";
    }
  } else {
    return "INACTIVE";
  }
}
```

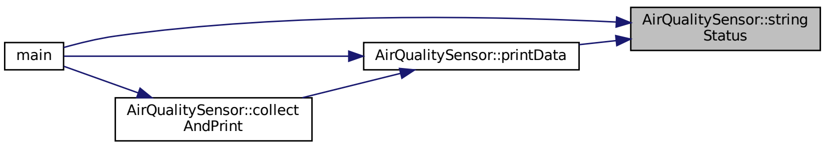
References checkAllgood(), and Sensor::isActive().

Referenced by main(), and printData().

Here is the call graph for this function:


AirQualitySensor::string Status
AirQualitySensor::checkAllgood
Sensor::getData
Sensor::isActive

Here is the caller graph for this function:


main
AirQualitySensor::printData
AirQualitySensor::string Status
AirQualitySensor::collect AndPrint

4.1.4 Friends And Related Function Documentation

4.1.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const AirQualitySensor & sensor )  [friend]
```

This method prints the AirQualitySensor object.

Parameters

os	
sensor	

Returns

std::ostream&

Definition at line 35 of file AirQualitySensor.cpp.

```
                                                                        {
    sensor.printData();
    return os;
}
```

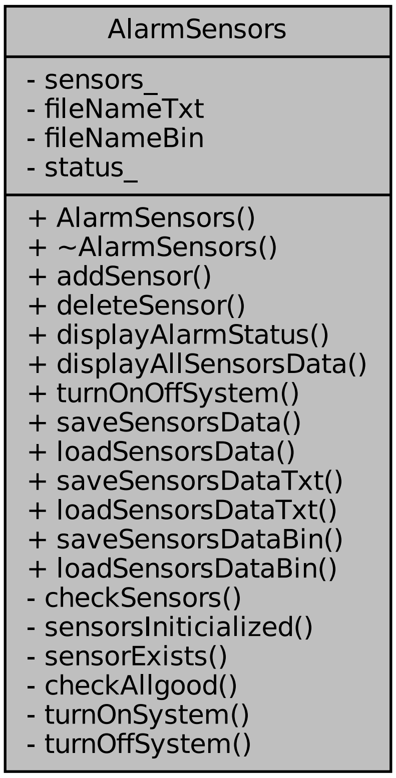
The documentation for this class was generated from the following files:

- src/AirQualitySensor.h
- src/AirQualitySensor.cpp

4.2 AlarmSensors Class Reference

```
#include <AlarmSensors.h>
```

Collaboration diagram for AlarmSensors:



Public Member Functions

- AlarmSensors ()

  Construct a new Alarm Sensors object.
- ∼AlarmSensors ()

  Destroy the Alarm Sensors object.
- void addSensor (int id, std::string type)

  Add a Sensor object.
- void deleteSensor (int id)

  Delete a Sensor object.
- void displayAlarmStatus ()

  Display the Alarm Status.
- void displayAllSensorsData ()

  Display all Sensors Data.
- void turnOnOffSystem (int input)

This method turns on or off the system.
- void saveSensorsData ()

  This method saves the sensors data to a file, one .txt and other one .dat.
- void loadSensorsData ()

  This method loads the sensors data from a file .dat, but you can change to loads the sensor from a .txt.
- void saveSensorsDataTxt ()

  This method saves the sensors data to a file .txt.
- void loadSensorsDataTxt ()

  This method loads the sensors data from a file .txt.
- void saveSensorsDataBin ()

  This method saves the sensors data to a file .dat.
- void loadSensorsDataBin ()

  This method loads the sensors data from a file .dat.

Private Member Functions

- int checkSensors ()

  Check the Sensors.
- bool sensorsIniticialized ()

  Check if the Sensors are Initialized.
- bool sensorExists (int id)

  Check if a Sensor exists.
- bool checkAllgood ()

  Check if the Sensors have good measurements.
- void turnOnSystem ()

  Turn on the System.
- void turnOffSystem ()

  Turn off the System.

Private Attributes

- std::set< Sensor * > sensors_

  This is the set of Sensor pointers.
- std::string fileNameTxt = "sensors.txt"

  This is the name of the file .txt.
- std::string fileNameBin = "sensors.dat"

  This is the name of the file .dat.
- bool status_ = true

  The status of the alarm.

4.2.1 Detailed Description

Definition at line 25 of file AlarmSensors.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 AlarmSensors()

```
AlarmSensors::AlarmSensors ( )  [explicit]
```

Construct a new Alarm Sensors object.

Definition at line 19 of file AlarmSensors.cpp.

```
                              {
  // El set de sensores se inicializa con un sensor de cada tipo
  sensors_.insert(new TemperatureSensor(1, true));
  sensors_.insert(new AirQualitySensor(2, true));
  sensors_.insert(new HydrometerSensor(3, true));
  sensors_.insert(new PressureSensor(4, true));
  sensors_.insert(new LightSensor(5, true));
  sensors_.insert(new PhSensor(6, true));
}
```

4.2.2.2 ∼AlarmSensors()

```
AlarmSensors::∼AlarmSensors ( )
```

Destroy the Alarm Sensors object.

Definition at line 29 of file AlarmSensors.cpp.

```
                               {
  // Destructor que elimina todos los sensores
  for (auto sensor : sensors_) {
    delete sensor;
  }
}
```

4.2.3 Member Function Documentation

4.2.3.1 addSensor()

```
void AlarmSensors::addSensor (
              int id,
              std::string type )
```

Add a Sensor object.

Parameters

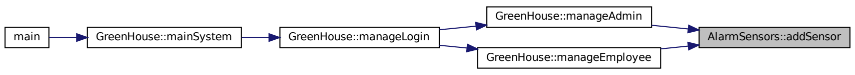
id	
type	

Definition at line 46 of file AlarmSensors.cpp.

```
                                                        {
  // Si el sensor ya existe no se puede añadir
  if (sensorExists(id)) {
    cout « "Sensor already exists" « endl;
    return;
  }
  // Pasar a mayusculas el tipo de sensor
  for (auto &c : type) {
    c = toupper(c);
  }
  // Añadir un sensor al set de sensores
  if (type == "TEMPERATURE") {
    sensors_.insert(new TemperatureSensor(id, true));
  } else if (type == "AIR_QUALITY") {
    sensors_.insert(new AirQualitySensor(id, true));
  } else if (type == "HYDROMETER") {
    sensors_.insert(new HydrometerSensor(id, true));
  } else if (type == "PRESSURE") {
    sensors_.insert(new PressureSensor(id, true));
  } else if (type == "LIGHT") {
    sensors_.insert(new LightSensor(id, true));
  } else if (type == "PH") {
    sensors_.insert(new PhSensor(id, true));
  } else {
    cout « "Sensor type not valid" « endl;
  }
}
```

Referenced by GreenHouse::manageAdmin(), and GreenHouse::manageEmployee().

Here is the caller graph for this function:



4.2.3.2 checkAllgood()

```
bool AlarmSensors::checkAllgood ( )  [private]
```

Check if the Sensors have good measurements.

Returns

true if the Sensors have good measurements

false if the Sensors do not have good measurements

Definition at line 102 of file AlarmSensors.cpp.

```
                                  {
  // Si los sensores tienen buenas mediciones
  for (auto sensor : sensors_) {
    if (!sensor->checkAllgood()) {
      return false;
    }
  }
  return true;
}
```

4.2.3.3 checkSensors()

```
int AlarmSensors::checkSensors ( )  [private]
```

Check the Sensors.

Returns

int

Definition at line 112 of file AlarmSensors.cpp.

```
                              {
  if (!sensorsIniticialized()) {
    return -1;
  } else {
    if (checkAllgood()) {
      return 1;
    } else {
      return 0;
    }
  }
}
```

4.2.3.4 deleteSensor()

```
void AlarmSensors::deleteSensor (
              int id )
```

Delete a Sensor object.

Parameters

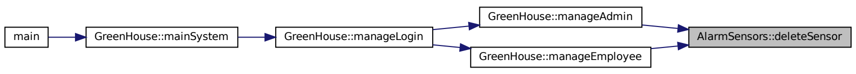
id	

Definition at line 74 of file AlarmSensors.cpp.

```
                                      {
  bool found = false;
  // Eliminar un sensor del set de sensores
  for (auto sensor : sensors_) {
    if (sensor->getId() == id) {
      std::cout « "Sensor with id: " « id « " deleted" « std::endl;
      sensors_.erase(sensor);
      delete sensor;
      found = true;
      break;
    }
  }
  if (!found) {
    std::cout « "Sensor with id: " « id « " not found" « std::endl;
  }
}
```

Referenced by GreenHouse::manageAdmin(), and GreenHouse::manageEmployee().

Here is the caller graph for this function:



4.2.3.5 displayAlarmStatus()

```
void AlarmSensors::displayAlarmStatus ( )
```

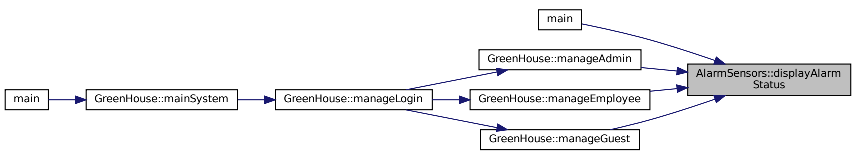
Display the Alarm Status.

Definition at line 124 of file AlarmSensors.cpp.

```
124                                       {
125   if (status_) {
126     if (checkSensors() == 1) {
127       cout « "---------------------------------------------------" « endl;
128       cout « "|                                                 |" « endl;
129       cout « "|         All sensors are in good status          |" « endl;
130       cout « "|                                                 |" « endl;
131       cout « "---------------------------------------------------" « endl;
132     } else if (checkSensors() == 0) {
133       // Dibujar el logo de alarma, el triangulo con la exclamacion en el medio
134       cout « "---------------------------------------------------" « endl;
135       cout « "|                                                 |" « endl;
136       cout « "|  ¡¡One or more sensors are not in good status¡¡ |" « endl;
137       cout « "|                                                 |" « endl;
138       cout « "---------------------------------------------------" « endl;
139     } else if (checkSensors() == -1) {
140       cout « "---------------------------------------------------" « endl;
141       cout « "|                                                 |" « endl;
142       cout « "|  ¡¡One or more sensors are not initialized¡¡    |" « endl;
143       cout « "|    do a collect of data to initialize them      |" « endl;
144       cout « "|                                                 |" « endl;
145       cout « "---------------------------------------------------" « endl;
146     } else {
147       cout « "---------------------------------------------------" « endl;
148       cout « "|                                                 |" « endl;
149       cout « "|            ¡¡Error in the system¡¡              |" « endl;
150       cout « "|                                                 |" « endl;
151       cout « "---------------------------------------------------" « endl;
152     }
153   } else {
154     cout « "---------------------------------------------------" « endl;
155     cout « "|                                                 |" « endl;
156     cout « "|                The system its off               |" « endl;
157     cout « "|                                                 |" « endl;
158     cout « "---------------------------------------------------" « endl;
159   }
160 }
```

Referenced by main(), GreenHouse::manageAdmin(), GreenHouse::manageEmployee(), and GreenHouse↩ ::manageGuest().

Here is the caller graph for this function:



4.2.3.6 displayAllSensorsData()

```
void AlarmSensors::displayAllSensorsData ( )
```

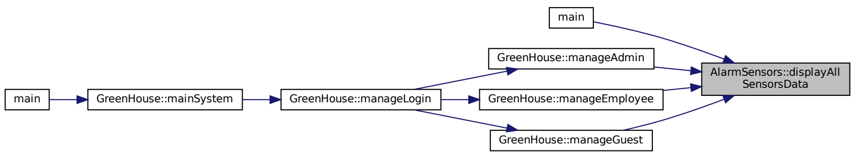
Display all Sensors Data.

Definition at line 162 of file AlarmSensors.cpp.

```
162                                     {
163   // Imprimir los datos de todos los sensores
164   for (auto sensor : sensors_) {
165     sensor->collectAndPrint();
166   }
167 }
```

Referenced by main(), GreenHouse::manageAdmin(), GreenHouse::manageEmployee(), and GreenHouse↩
::manageGuest().

Here is the caller graph for this function:



4.2.3.7 loadSensorsData()

```
void AlarmSensors::loadSensorsData ( )
```

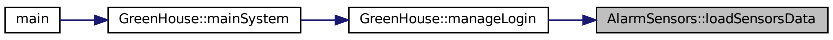
This method loads the sensors data from a file .dat, but you can change to loads the sensor from a .txt.

Definition at line 290 of file AlarmSensors.cpp.

```
290                               {
291   // Cargar los datos de los sensores de un archivo binario
292   loadSensorsDataBin();
293   // Cargar los datos de los sensores de un archivo de texto
294   // loadSensorsDataTxt();
295 }
```

Referenced by GreenHouse::manageLogin().

Here is the caller graph for this function:



4.2.3.8 loadSensorsDataBin()

```
void AlarmSensors::loadSensorsDataBin ( )
```

This method loads the sensors data from a file .dat.

Definition at line 262 of file AlarmSensors.cpp.

```
                                       {
  // Cargar los datos de los sensores de un archivo binario usando trunc
  ifstream file(fileNameBin, ios::binary);
  int id;
  string type;
  int data;
  while (file.read(reinterpret_cast<char *>(&id), sizeof(int))) {
    // Strings cannot be read directly as binary data due to their dynamic size
    // First, read the length of the string
    size_t typeLength;
    if (file.read(reinterpret_cast<char *>(&typeLength), sizeof(size_t))) {
      // Resize the string to the read length
      type.resize(typeLength);
      // Now, read the string data
      file.read(&type[0], typeLength);
    }
    file.read(reinterpret_cast<char *>(&data), sizeof(int));
    addSensor(id, type);
    for (auto sensor : sensors_) {
      if (sensor->getId() == id) {
        sensor->setData(data);
      }
    }
    std::cout « "Sensor with id: " « id « " loaded (binary)" « std::endl;
  }
  file.close();
}
```

4.2.3.9 loadSensorsDataTxt()

```
void AlarmSensors::loadSensorsDataTxt ( )
```

This method loads the sensors data from a file .txt.

Definition at line 243 of file AlarmSensors.cpp.

```
                                       {
  // Cargar los datos de los sensores de un archivo de texto
  ifstream file;
  file.open(fileNameTxt);
  int id;
  string type;
  int data;
  while (file » id » type » data) {
    addSensor(id, type);
    for (auto sensor : sensors_) {
      if (sensor->getId() == id) {
        sensor->setData(data);
      }
    }
    std::cout « "Sensor with id: " « id « " loaded (txt)" « std::endl;
  }
  file.close();
}
```

4.2.3.10 saveSensorsData()

```
void AlarmSensors::saveSensorsData ( )
```

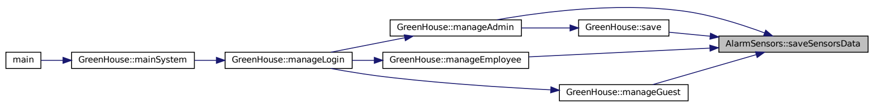
This method saves the sensors data to a file, one .txt and other one .dat.

Definition at line 236 of file AlarmSensors.cpp.

```
                                      {
  // Guardar los datos de los sensores en un archivo binario
  saveSensorsDataBin();
  // GUardar los datos de los sensores en un archivo de texto
  saveSensorsDataTxt();
}
```

Referenced by GreenHouse::manageAdmin(), GreenHouse::manageEmployee(), GreenHouse::manageGuest(), and GreenHouse::save().

Here is the caller graph for this function:



4.2.3.11 saveSensorsDataBin()

```
void AlarmSensors::saveSensorsDataBin ( )
```

This method saves the sensors data to a file .dat.

Definition at line 217 of file AlarmSensors.cpp.

```
                                         {
  ofstream file(fileNameBin, ios::binary | ios::trunc);
  for (auto sensor : sensors_) {
    int id = sensor->getId();
    file.write(reinterpret_cast<char *>(&id), sizeof(int));

    size_t typeLength = sensor->getType().length();
    file.write(reinterpret_cast<char *>(&typeLength), sizeof(size_t));
    file.write(sensor->getType().c_str(), typeLength);

    int data = sensor->getData();
    file.write(reinterpret_cast<char *>(&data), sizeof(int));

    std::cout « "Sensor with id: " « sensor->getId() « " saved (binary)"
              « std::endl;
  }
  file.close();
}
```

4.2.3.12 saveSensorsDataTxt()

```
void AlarmSensors::saveSensorsDataTxt ( )
```

This method saves the sensors data to a file .txt.

Definition at line 204 of file AlarmSensors.cpp.

```
                                          {
  // Guardar los datos de los sensores en un archivo de texto
  ofstream file;
  file.open(fileNameTxt);
  for (auto sensor : sensors_) {
    file « sensor->getId() « " " « sensor->getType() « " "
         « sensor->getData() « endl;
    std::cout « "Sensor with id: " « sensor->getId() « " saved (txt)"
              « std::endl;
  }
  file.close();
}
```

4.2.3.13 sensorExists()

```
bool AlarmSensors::sensorExists (
               int id ) [private]
```

Check if a Sensor exists.

Parameters

id	

Returns

true if the Sensor exists

false if the Sensor does not exist

Definition at line 36 of file AlarmSensors.cpp.

```
                                         {
  // Ver si existe un sensor
  for (auto sensor : sensors_) {
    if (sensor->getId() == id) {
      return true;
    }
  }
  return false;
}
```

4.2.3.14 sensorsIniticialized()

```
bool AlarmSensors::sensorsIniticialized ( ) [private]
```

Check if the Sensors are Initialized.

Returns

true if the Sensors are Initialized

false if the Sensors are not Initialized

Definition at line 91 of file AlarmSensors.cpp.
```
                                         {
  // Los sensores estan iniciados si todos los sensores no tienen el valor por
  // defecto de -1
  for (auto sensor : sensors_) {
    if (sensor->getData() == -1) {
      return false;
    }
  }
  return true;
}
```

4.2.3.15 turnOffSystem()

`void AlarmSensors::turnOffSystem ( ) [private]`

Turn off the System.

Definition at line 176 of file AlarmSensors.cpp.
```
                                  {
  // Apagar todos los sensores del set
  for (auto sensor : sensors_) {
    sensor->turnOff();
  }
}
```

4.2.3.16 turnOnOffSystem()

```
void AlarmSensors::turnOnOffSystem (
              int input )
```

This method turns on or off the system.

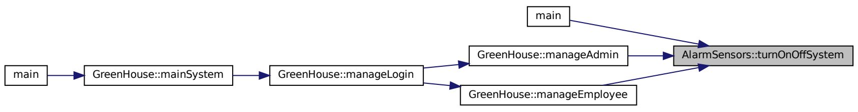
Parameters

input	

Definition at line 183 of file AlarmSensors.cpp.
```
                                               {
  // Si el input es igual a true entonces encender todos los sensores
  if (input == 1) {
    turnOnSystem();
    cout « "----------------------------------------------------" « endl;
    cout « "|                                                  |" « endl;
    cout « "|                  System turned on                |" « endl;
    cout « "|                                                  |" « endl;
    cout « "----------------------------------------------------" « endl;
    status_ = true;
  } else if (input == 2) {
    turnOffSystem();
    cout « "----------------------------------------------------" « endl;
    cout « "|                                                  |" « endl;
    cout « "|                  System turned off               |" « endl;
    cout « "|                                                  |" « endl;
    cout « "---------------------------------------------------" « endl;
    status_ = false;
  }
}
```

Referenced by main(), GreenHouse::manageAdmin(), and GreenHouse::manageEmployee().

Here is the caller graph for this function:



4.2.3.17 turnOnSystem()

`void AlarmSensors::turnOnSystem ( )  [private]`

Turn on the System.

Definition at line 169 of file AlarmSensors.cpp.

```
                              {
  // Encender todos los sensores del set
  for (auto sensor : sensors_) {
    sensor->turnOn();
  }
}
```

4.2.4 Member Data Documentation

4.2.4.1 fileNameBin

`std::string AlarmSensors::fileNameBin = "sensors.dat"  [private]`

This is the name of the file .dat.

Definition at line 131 of file AlarmSensors.h.

4.2.4.2 fileNameTxt

`std::string AlarmSensors::fileNameTxt = "sensors.txt"  [private]`

This is the name of the file .txt.

Definition at line 126 of file AlarmSensors.h.

4.2.4.3 sensors_

`std::set<Sensor *> AlarmSensors::sensors_ [private]`

This is the set of Sensor pointers.

Definition at line 119 of file AlarmSensors.h.

4.2.4.4 status_

`bool AlarmSensors::status_ = true [private]`

The status of the alarm.

Definition at line 183 of file AlarmSensors.h.

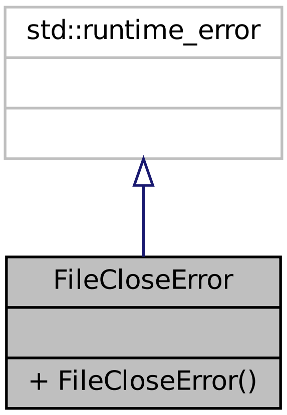
The documentation for this class was generated from the following files:

- src/AlarmSensors.h
- src/AlarmSensors.cpp

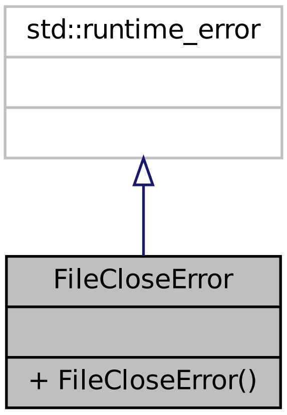
4.3 FileCloseError Class Reference

`#include <Exceptions.h>`

Inheritance diagram for FileCloseError:



Collaboration diagram for FileCloseError:



Public Member Functions

- FileCloseError (const std::string &filename)

  Construct a new File Close Error object.

4.3.1 Detailed Description

Definition at line 25 of file Exceptions.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 FileCloseError()

```
FileCloseError::FileCloseError (
                const std::string & filename )  [inline], [explicit]
```

Construct a new File Close Error object.

Parameters

filename	

Definition at line 32 of file Exceptions.h.

```
        : std::runtime_error("Error closing file: " + filename) {}
```

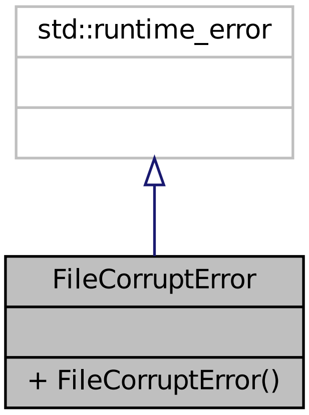
The documentation for this class was generated from the following file:

- src/Exceptions.h

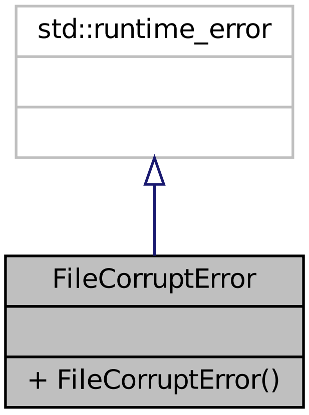
4.4 FileCorruptError Class Reference

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

Inheritance diagram for FileCorruptError:



Collaboration diagram for FileCorruptError:



Public Member Functions

- FileCorruptError (const std::string &filename)

  Construct a new File Corrupt Error object.

4.4.1 Detailed Description

Definition at line 91 of file Exceptions.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 FileCorruptError()

```
FileCorruptError::FileCorruptError (
             const std::string & filename )  [inline], [explicit]
```

Construct a new File Corrupt Error object.

Parameters

filename	

Definition at line 98 of file Exceptions.h.

```
        : std::runtime_error("File is corrupt: " + filename) {}
```

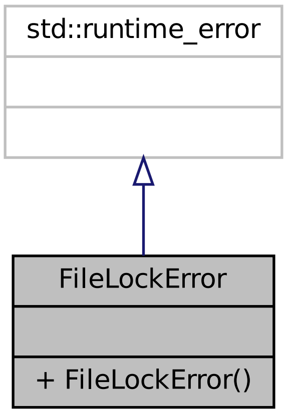
The documentation for this class was generated from the following file:

- src/Exceptions.h

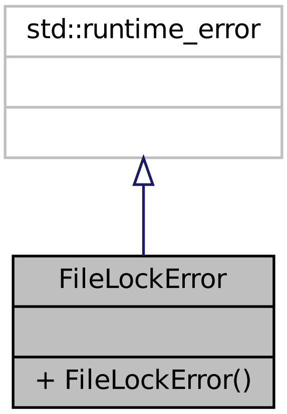
4.5 FileLockError Class Reference

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

Inheritance diagram for FileLockError:



Collaboration diagram for FileLockError:



Public Member Functions

- FileLockError (const std::string &filename)

    Construct a new File Lock Error object.

4.5.1 Detailed Description

Definition at line 80 of file Exceptions.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 FileLockError()

```
FileLockError::FileLockError (
             const std::string & filename )  [inline], [explicit]
```

Construct a new File Lock Error object.

Parameters

filename	

Definition at line 87 of file Exceptions.h.

```
88       : std::runtime_error("File is locked: " + filename) {}
```

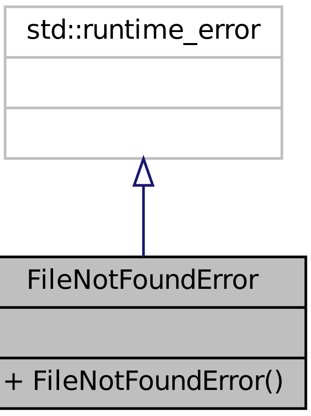
The documentation for this class was generated from the following file:

- src/Exceptions.h

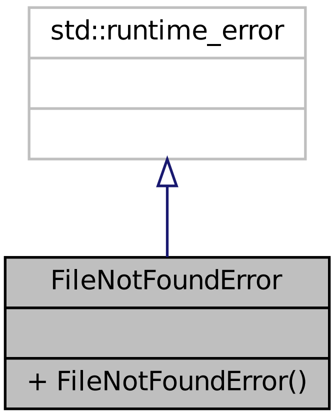
4.6 FileNotFoundError Class Reference

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

Inheritance diagram for FileNotFoundError:



Collaboration diagram for FileNotFoundError:



Public Member Functions

- FileNotFoundError (const std::string &filename)

    Construct a new File Not Found Error object.

4.6.1 Detailed Description

Definition at line 69 of file Exceptions.h.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 FileNotFoundError()

```
FileNotFoundError::FileNotFoundError (
            const std::string & filename )  [inline], [explicit]
```

Construct a new File Not Found Error object.

Parameters

filename	

Definition at line 76 of file Exceptions.h.

```
77        : std::runtime_error("File not found: " + filename) {}
```

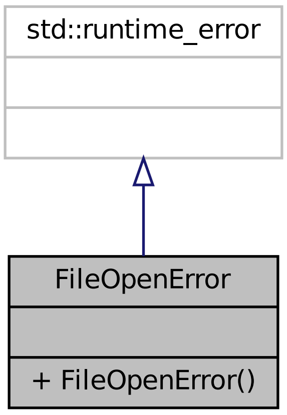
The documentation for this class was generated from the following file:

- src/Exceptions.h

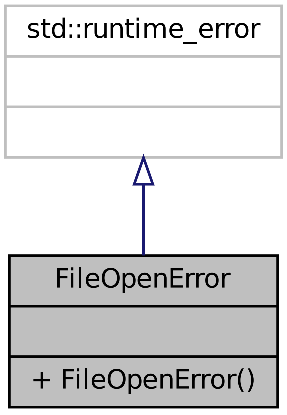
4.7 FileOpenError Class Reference

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

Inheritance diagram for FileOpenError:



Collaboration diagram for FileOpenError:



Public Member Functions

- FileOpenError (const std::string &filename)

  Construct a new File Open Error object.

4.7.1 Detailed Description

Definition at line 14 of file Exceptions.h.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 FileOpenError()

```
FileOpenError::FileOpenError (
            const std::string & filename )  [inline], [explicit]
```

Construct a new File Open Error object.

Parameters

filename	

Definition at line 21 of file Exceptions.h.

```
22        : std::runtime_error("Error opening file: " + filename) {}
```

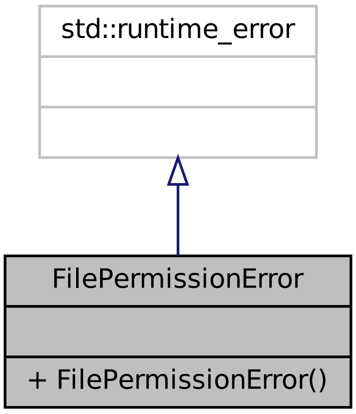
The documentation for this class was generated from the following file:

- src/Exceptions.h

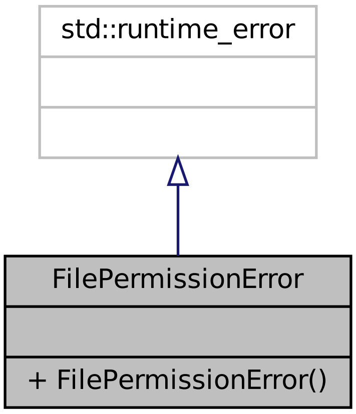
4.8 FilePermissionError Class Reference

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

Inheritance diagram for FilePermissionError:



Collaboration diagram for FilePermissionError:



Public Member Functions

- FilePermissionError (const std::string &filename)

  Construct a new File Permission Error object.

4.8.1 Detailed Description

Definition at line 58 of file Exceptions.h.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 FilePermissionError()

```
FilePermissionError::FilePermissionError (
            const std::string & filename )  [inline], [explicit]
```

Construct a new File Permission Error object.

Parameters

filename	

Definition at line 65 of file Exceptions.h.

```
66        : std::runtime_error("Permission denied: " + filename) {}
```

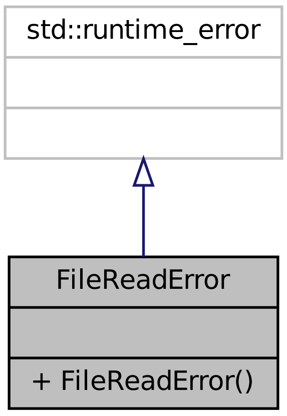
The documentation for this class was generated from the following file:

- src/Exceptions.h

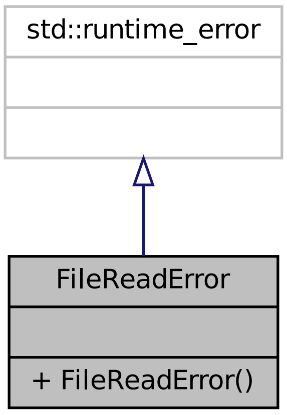
4.9 FileReadError Class Reference

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

Inheritance diagram for FileReadError:



Collaboration diagram for FileReadError:



Public Member Functions

- FileReadError (const std::string &filename)

  Construct a new File Read Error object.

4.9.1 Detailed Description

Definition at line 36 of file Exceptions.h.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 FileReadError()

```
FileReadError::FileReadError (
             const std::string & filename )  [inline], [explicit]
```

Construct a new File Read Error object.

Parameters

filename	

Definition at line 43 of file Exceptions.h.

```
44        : std::runtime_error("Error reading file: " + filename) {}
```

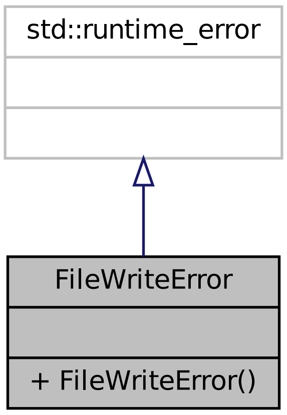
The documentation for this class was generated from the following file:

- src/Exceptions.h

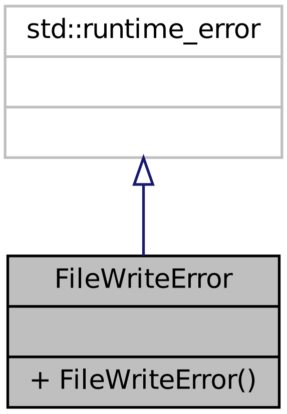
4.10 FileWriteError Class Reference

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

Inheritance diagram for FileWriteError:



Collaboration diagram for FileWriteError:



Public Member Functions

- FileWriteError (const std::string &filename)

    Construct a new File Write Error object.

4.10.1 Detailed Description

Definition at line 47 of file Exceptions.h.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 FileWriteError()

```
FileWriteError::FileWriteError (
            const std::string & filename )  [inline], [explicit]
```

Construct a new File Write Error object.

Parameters

filename	

Definition at line 54 of file Exceptions.h.

```
        : std::runtime_error("Error writing file: " + filename) {}
```

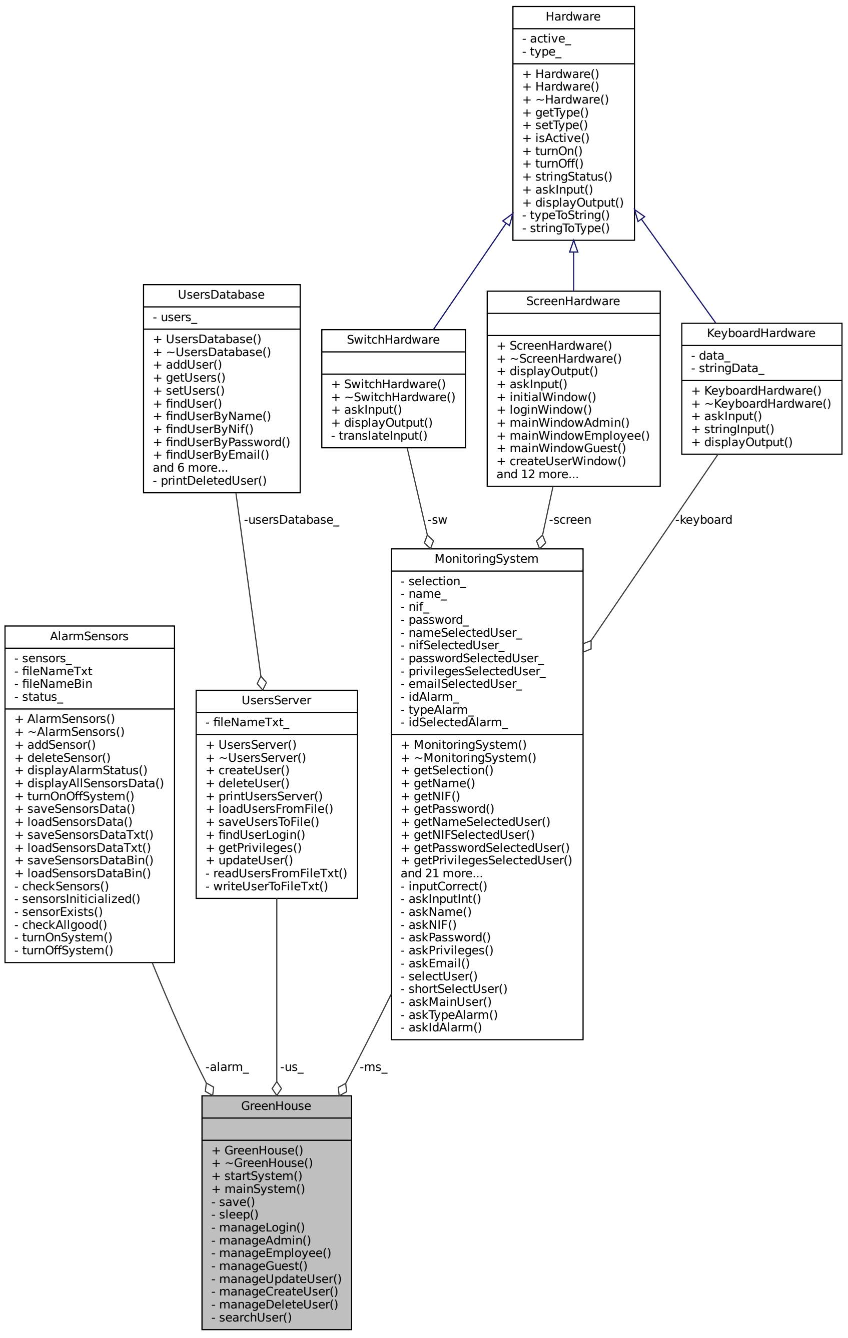
The documentation for this class was generated from the following file:

- src/Exceptions.h

4.11 GreenHouse Class Reference

```
#include <GreenHouse.h>
```

Collaboration diagram for GreenHouse:



Public Member Functions

- GreenHouse ()

  Construct a new Green House object.
- ∼GreenHouse ()

  Destroy the Green House object.
- void startSystem ()

  Start the system.

- void mainSystem ()

  Main system, after the system is started, you call this method to manage the system with your options that contains your privilege.

Private Member Functions

- void save ()

  Load the system.
- void sleep ()

  Save the system.
- void manageLogin ()

  Manage the login.
- void manageAdmin ()

  Manage the admin.
- void manageEmployee ()

  Manage the employee.
- void manageGuest ()

  Manage the guest.
- void manageUpdateUser ()

  Manage the update user.
- void manageCreateUser ()

  Manage the create user.
- void manageDeleteUser ()

  Manage the delete user.
- bool searchUser (std::string name, std::string password, std::string nif)

  Manage search user.

Private Attributes

- AlarmSensors ∗ alarm_

  This attribute is the AlarmSensors object (set of pointers to sensors).
- MonitoringSystem ∗ ms_

  This attribute is the MonitoringSystem object.
- UsersServer ∗ us_

  This attribute is the UsersServer object (set of pointers to users).

4.11.1 Detailed Description

Definition at line 16 of file GreenHouse.h.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 GreenHouse()

```
GreenHouse::GreenHouse ( )
```

Construct a new Green House object.

Definition at line 16 of file GreenHouse.cpp.

```
    : alarm_(new AlarmSensors()),

      ms_(new MonitoringSystem(new ScreenHardware(true),
                               new KeyboardHardware(true),
                               new SwitchHardware(true))),
      us_(new UsersServer()) {
  // Constructor ahora inicializa todos los atributos privados correctamente.
}
```

4.11.2.2 ∼GreenHouse()

```
GreenHouse::∼GreenHouse ( )
```

Destroy the Green House object.

Definition at line 26 of file GreenHouse.cpp.

```
                          {
  // Destructor ahora elimina todos los atributos privados correctamente.
  delete alarm_;
  delete ms_;
  delete us_;
}
```

References alarm_, ms_, and us_.

4.11.3 Member Function Documentation

4.11.3.1 mainSystem()

```
void GreenHouse::mainSystem ( )
```

Main system, after the system is started, you call this method to manage the system with your options that contains your privilege.

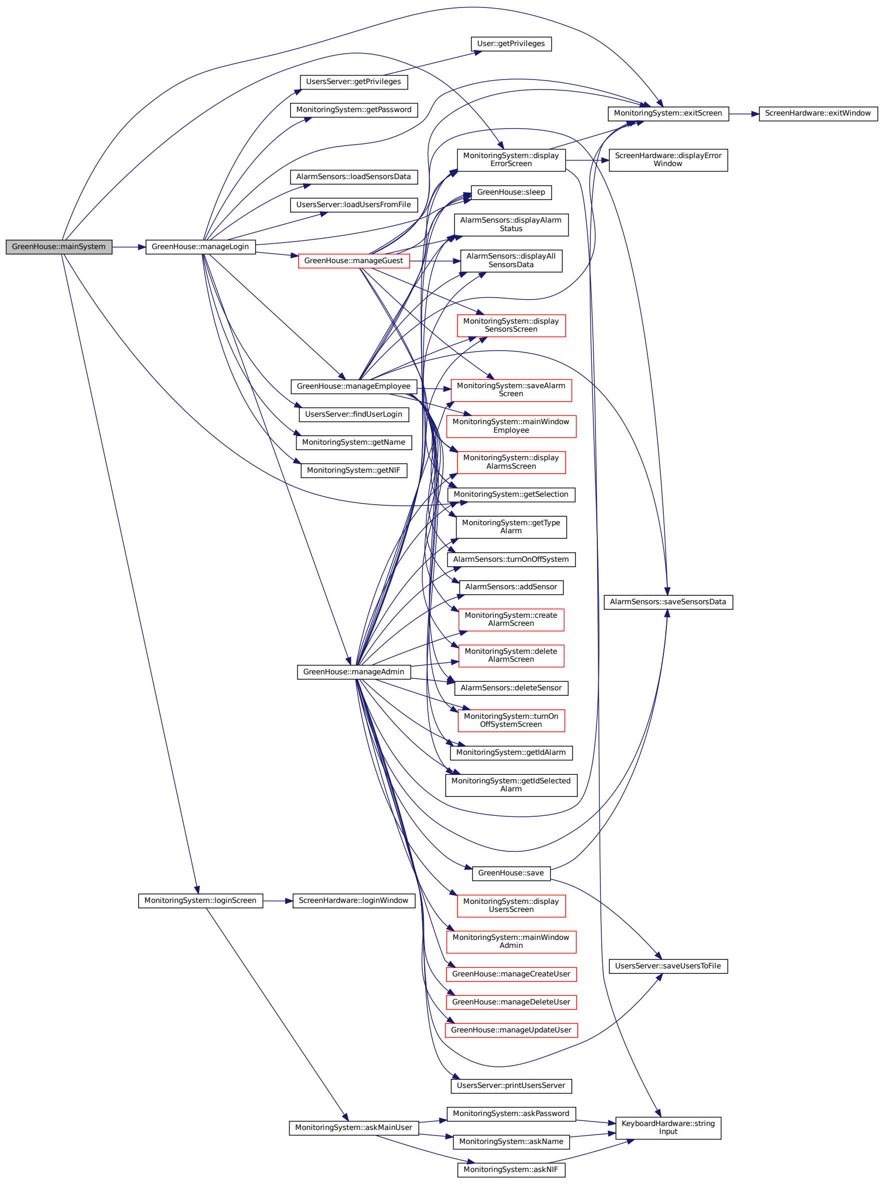
Definition at line 251 of file GreenHouse.cpp.

```
                               {
  // SI la seleccion en startSystem() es 1, entonces se ejecuta el loginScreen()
  // Hacemos mejor un switch para que sea más fácil de leer
  switch (ms_->getSelection()) {
  case 1:
    ms_->loginScreen();
    manageLogin();
    break;
  case 2:
    ms_->exitScreen();
    break;
  default:
    ms_->displayErrorScreen();
    break;
  }
  /* if (ms_->getSelection() == 1)
  {
      us_->loadUsersFromFile();
      ms_->loginScreen();

  } else if (ms_->getSelection() == 2)
  {
      ms_->exitScreen();
  } else {
      ms_->displayErrorScreen();
  }
  */
}
```

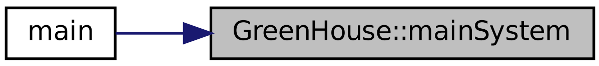
References MonitoringSystem::displayErrorScreen(), MonitoringSystem::exitScreen(), MonitoringSystem::get⮐Selection(), MonitoringSystem::loginScreen(), manageLogin(), and ms_.

Referenced by main().

Here is the call graph for this function:


GreenHouse::mainSystem
GreenHouse::manageLogin
User::getPrivileges
UsersServer::getPrivileges
MonitoringSystem::getPassword
MonitoringSystem::exitScreen
ScreenHardware::exitWindow
MonitoringSystem::display ErrorScreen
ScreenHardware::displayError Window
AlarmSensors::loadSensorsData
UsersServer::loadUsersFromFile
GreenHouse::sleep
AlarmSensors::displayAlarm Status
GreenHouse::manageGuest
AlarmSensors::displayAll SensorsData
MonitoringSystem::display SensorsScreen
GreenHouse::manageEmployee
MonitoringSystem::saveAlarm Screen
UsersServer::findUserLogin
MonitoringSystem::mainWindow Employee
MonitoringSystem::getName
MonitoringSystem::display AlarmsScreen
MonitoringSystem::getNIF
MonitoringSystem::getSelection
MonitoringSystem::getType Alarm
AlarmSensors::turnOnOffSystem
AlarmSensors::addSensor
AlarmSensors::saveSensorsData
MonitoringSystem::create AlarmScreen
MonitoringSystem::delete AlarmScreen
GreenHouse::manageAdmin
AlarmSensors::deleteSensor
MonitoringSystem::turnOn OffSystemScreen
MonitoringSystem::getIdAlarm
MonitoringSystem::getIdSelected Alarm
GreenHouse::save
MonitoringSystem::loginScreen
ScreenHardware::loginWindow
MonitoringSystem::display UsersScreen
MonitoringSystem::mainWindow Admin
UsersServer::saveUsersToFile
GreenHouse::manageCreateUser
GreenHouse::manageDeleteUser
GreenHouse::manageUpdateUser
UsersServer::printUsersServer
MonitoringSystem::askPassword
MonitoringSystem::askMainUser
KeyboardHardware::string Input
MonitoringSystem::askName
MonitoringSystem::askNIF

Here is the caller graph for this function:



4.11.3.2 manageAdmin()

```
void GreenHouse::manageAdmin ( )  [private]
```

Manage the admin.

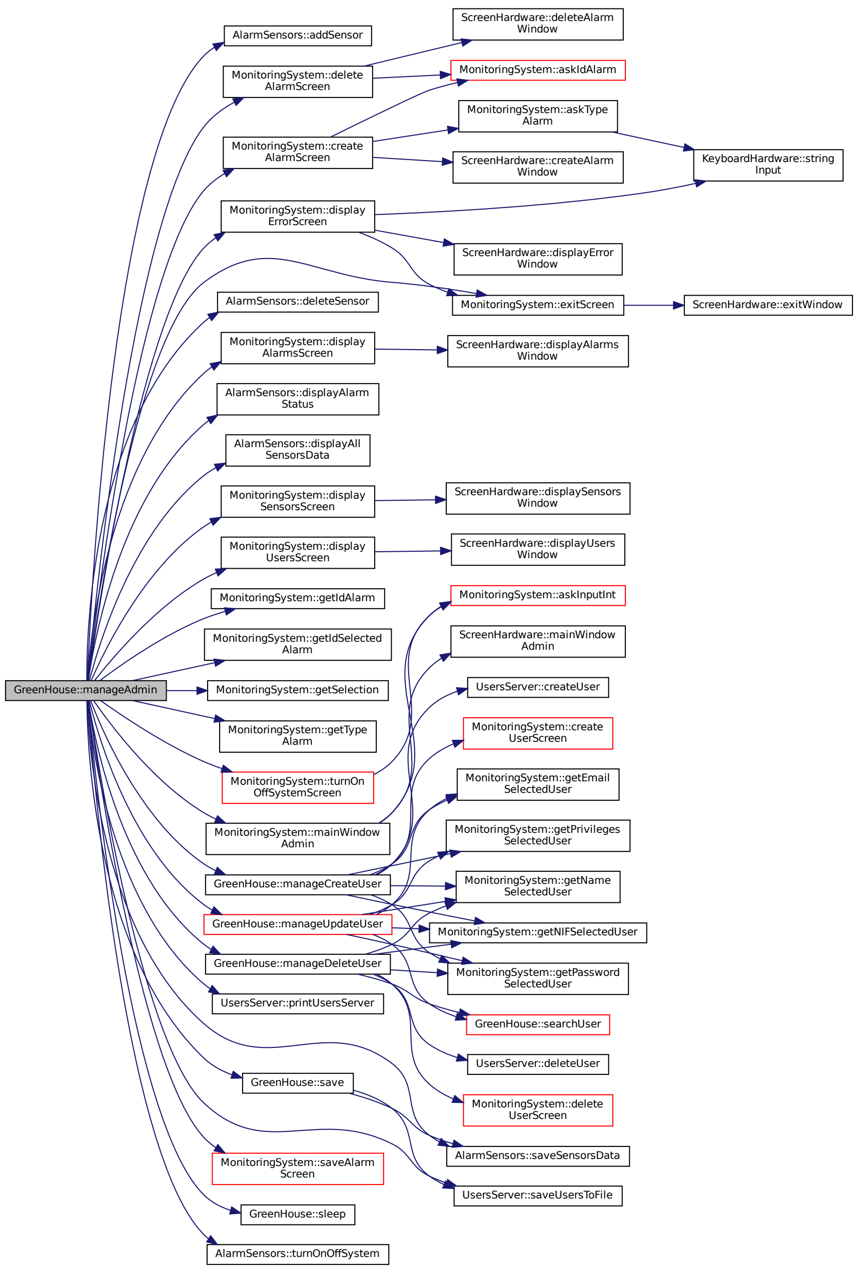
Definition at line 81 of file GreenHouse.cpp.

```
                              {
  bool exit = false;
  // Mostramos la ventana de admin
  do {
    // Poner unos sefgundos de espera para que se vea el mensaje
    sleep();
    ms_->mainWindowAdmin();
    switch (ms_->getSelection()) {
    case 1:
      manageCreateUser();
      // us_->saveUsersToFile();
      break;
    case 2:
      manageDeleteUser();
      // us_->saveUsersToFile();
      break;
    case 3:
      manageUpdateUser();
      // us_->saveUsersToFile();
      break;
    case 4:
      ms_->displayUsersScreen();
      us_->printUsersServer();
      break;
    case 5:
      ms_->createAlarmScreen();
      alarm_->addSensor(ms_->getIdAlarm(), ms_->getTypeAlarm());
      break;
    case 6:
      ms_->deleteAlarmScreen();
      alarm_->deleteSensor(ms_->getIdSelectedAlarm());
      break;
    case 7:
      ms_->displaySensorsScreen();
      alarm_->displayAllSensorsData();
      break;
    case 8:
      ms_->displayAlarmsScreen();
      alarm_->displayAlarmStatus();
      break;
    case 9:
      ms_->turnOnOffSystemScreen();
      alarm_->turnOnOffSystem(ms_->getSelection());
      break;
    case 10:
      us_->saveUsersToFile();
      break;
    case 11:
      ms_->saveAlarmScreen();
      alarm_->saveSensorsData();
      sleep();
      break;
    case 12:
      save();
      sleep();
      ms_->exitScreen();
      exit = true;
      break;
    default:
      ms_->displayErrorScreen();
      break;
    }
  } while (!exit);
}
```

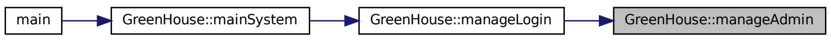
References AlarmSensors::addSensor(), alarm_, MonitoringSystem::createAlarmScreen(), MonitoringSystem::deleteAlarmScreen(), AlarmSensors::deleteSensor(), MonitoringSystem::displayAlarmsScreen(), AlarmSensors::displayAlarmStatus(), AlarmSensors::displayAllSensorsData(), MonitoringSystem::displayErrorScreen(), MonitoringSystem::displaySensorsScreen(), MonitoringSystem::displayUsersScreen(), MonitoringSystem::exitScreen(), MonitoringSystem::getIdAlarm(), MonitoringSystem::getIdSelectedAlarm(), MonitoringSystem::getSelection(), MonitoringSystem::getTypeAlarm(), MonitoringSystem::mainWindowAdmin(), manageCreateUser(), manageDeleteUser(), manageUpdateUser(), ms_, UsersServer::printUsersServer(), save(), MonitoringSystem::saveAlarmScreen(), AlarmSensors::saveSensorsData(), UsersServer::saveUsersToFile(), sleep(), AlarmSensors::turnOnOffSystem(), MonitoringSystem::turnOnOffSystemScreen(), and us_.

Referenced by manageLogin().

Here is the call graph for this function:


GreenHouse::manageAdmin
AlarmSensors::addSensor
MonitoringSystem::delete AlarmScreen
MonitoringSystem::create AlarmScreen
MonitoringSystem::display ErrorScreen
AlarmSensors::deleteSensor
MonitoringSystem::display AlarmsScreen
AlarmSensors::displayAlarm Status
AlarmSensors::displayAll SensorsData
MonitoringSystem::display SensorsScreen
MonitoringSystem::display UsersScreen
MonitoringSystem::getIdAlarm
MonitoringSystem::getIdSelected Alarm
MonitoringSystem::getSelection
MonitoringSystem::getType Alarm
MonitoringSystem::turnOn OffSystemScreen
MonitoringSystem::mainWindow Admin
GreenHouse::manageCreateUser
GreenHouse::manageUpdateUser
GreenHouse::manageDeleteUser
UsersServer::printUsersServer
GreenHouse::save
MonitoringSystem::saveAlarm Screen
GreenHouse::sleep
AlarmSensors::turnOnOffSystem
ScreenHardware::deleteAlarm Window
MonitoringSystem::askIdAlarm
MonitoringSystem::askType Alarm
ScreenHardware::createAlarm Window
KeyboardHardware::string Input
ScreenHardware::displayError Window
MonitoringSystem::exitScreen
ScreenHardware::exitWindow
ScreenHardware::displayAlarms Window
ScreenHardware::displaySensors Window
ScreenHardware::displayUsers Window
MonitoringSystem::askInputInt
ScreenHardware::mainWindow Admin
UsersServer::createUser
MonitoringSystem::create UserScreen
MonitoringSystem::getEmail SelectedUser
MonitoringSystem::getPrivileges SelectedUser
MonitoringSystem::getName SelectedUser
MonitoringSystem::getNIFSelectedUser
MonitoringSystem::getPassword SelectedUser
GreenHouse::searchUser
UsersServer::deleteUser
MonitoringSystem::delete UserScreen
AlarmSensors::saveSensorsData
UsersServer::saveUsersToFile

Here is the caller graph for this function:



4.11.3.3 manageCreateUser()

```
void GreenHouse::manageCreateUser ( )  [private]
```

Manage the create user.

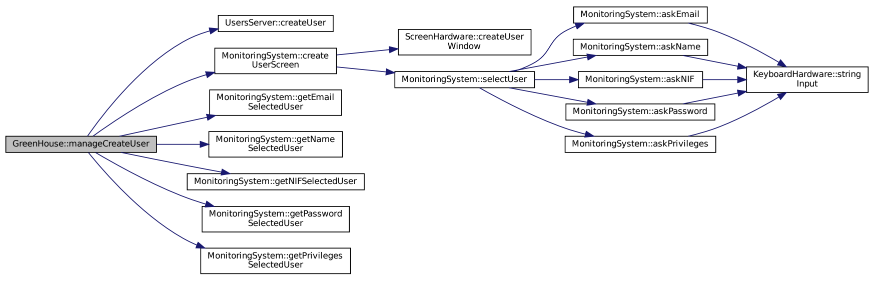
Definition at line 53 of file GreenHouse.cpp.

```
53                                  {
54   ms_->createUserScreen();
55   us_->createUser(ms_->getNameSelectedUser(), ms_->getNIFSelectedUser(),
56                   ms_->getPasswordSelectedUser(),
57                   ms_->getPrivilegesSelectedUser(),
58                   ms_->getEmailSelectedUser());
59 }
```

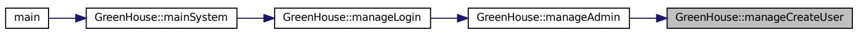
References UsersServer::createUser(), MonitoringSystem::createUserScreen(), MonitoringSystem::get←EmailSelectedUser(), MonitoringSystem::getNameSelectedUser(), MonitoringSystem::getNIFSelectedUser(), MonitoringSystem::getPasswordSelectedUser(), MonitoringSystem::getPrivilegesSelectedUser(), ms_, and us_.

Referenced by manageAdmin().

Here is the call graph for this function:


GreenHouse::manageCreateUser
UsersServer::createUser
MonitoringSystem::create UserScreen
MonitoringSystem::getEmail SelectedUser
MonitoringSystem::getName SelectedUser
MonitoringSystem::getNIFSelectedUser
MonitoringSystem::getPassword SelectedUser
MonitoringSystem::getPrivileges SelectedUser
ScreenHardware::createUser Window
MonitoringSystem::selectUser
MonitoringSystem::askEmail
MonitoringSystem::askName
MonitoringSystem::askNIF
MonitoringSystem::askPassword
MonitoringSystem::askPrivileges
KeyboardHardware::string Input

Here is the caller graph for this function:



4.11.3.4 manageDeleteUser()

```
void GreenHouse::manageDeleteUser ( )  [private]
```

Manage the delete user.

Definition at line 43 of file GreenHouse.cpp.

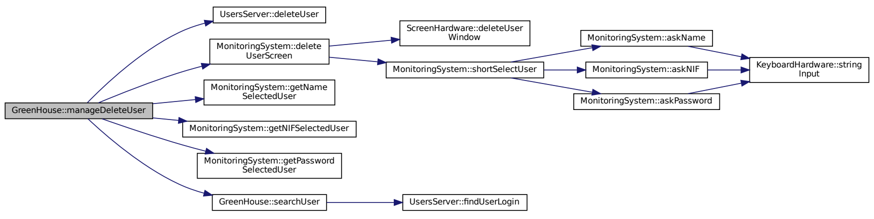
```
43                                  {
44   ms_->deleteUserScreen();
45   if (searchUser(ms_->getNameSelectedUser(), ms_->getPasswordSelectedUser(),
46                  ms_->getNIFSelectedUser())) {
47     us_->deleteUser(ms_->getNIFSelectedUser());
48   } else {
49     printf("Usuario no encontrado\n");
```

```
  }
}
```

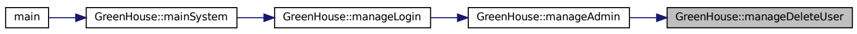
References UsersServer::deleteUser(), MonitoringSystem::deleteUserScreen(), MonitoringSystem::getName↩ SelectedUser(), MonitoringSystem::getNIFSelectedUser(), MonitoringSystem::getPasswordSelectedUser(), ms_, searchUser(), and us_.

Referenced by manageAdmin().

Here is the call graph for this function:


GreenHouse::manageDeleteUser
UsersServer::deleteUser
MonitoringSystem::delete UserScreen
MonitoringSystem::getName SelectedUser
MonitoringSystem::getNIFSelectedUser
MonitoringSystem::getPassword SelectedUser
GreenHouse::searchUser
ScreenHardware::deleteUser Window
MonitoringSystem::shortSelectUser
UsersServer::findUserLogin
MonitoringSystem::askName
MonitoringSystem::askNIF
MonitoringSystem::askPassword
KeyboardHardware::string Input

Here is the caller graph for this function:



4.11.3.5 manageEmployee()

```
void GreenHouse::manageEmployee ( )  [private]
```

Manage the employee.

Definition at line 146 of file GreenHouse.cpp.

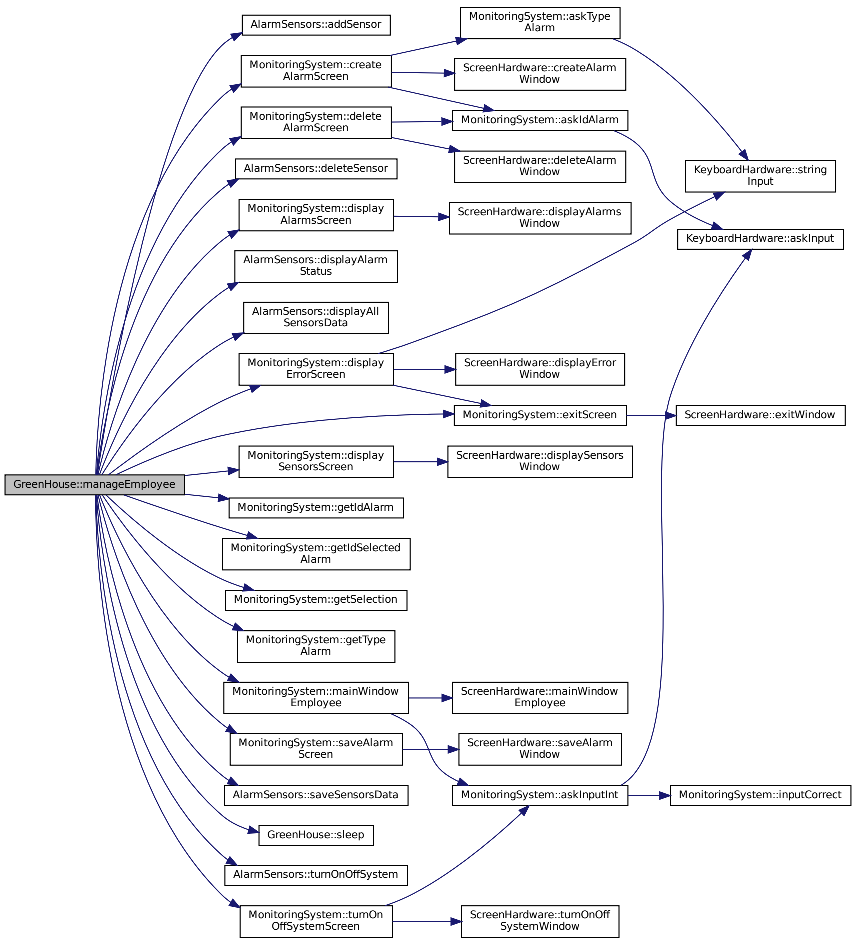
```
                                {
  bool exit = false;
  // Mostramos la ventana de employee
  do {
    sleep();
    ms_->mainWindowEmployee();
    switch (ms_->getSelection()) {
    case 1:
      ms_->createAlarmScreen();
      alarm_->addSensor(ms_->getIdAlarm(), ms_->getTypeAlarm());
      break;
    case 2:
      ms_->deleteAlarmScreen();
      alarm_->deleteSensor(ms_->getIdSelectedAlarm());
      break;
    case 3:
      ms_->displaySensorsScreen();
      alarm_->displayAllSensorsData();
      break;
    case 4:
      ms_->displayAlarmsScreen();
      alarm_->displayAlarmStatus();
      break;
    case 5:
      ms_->turnOnOffSystemScreen();
      alarm_->turnOnOffSystem(ms_->getSelection());
```

```
        break;
      case 6:
        ms_->saveAlarmScreen();
        alarm_->saveSensorsData();
        break;
      case 7:
        ms_->exitScreen();
        ms_->saveAlarmScreen();
        alarm_->saveSensorsData();
        exit = true;
        break;
      default:
        ms_->displayErrorScreen();
        break;
      }
    } while (!exit);
}
```

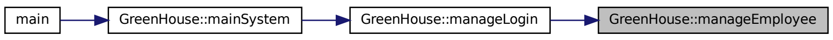
References AlarmSensors::addSensor(), alarm_, MonitoringSystem::createAlarmScreen(), MonitoringSystem::deleteAlarmScreen(), AlarmSensors::deleteSensor(), MonitoringSystem::displayAlarmsScreen(), AlarmSensors::displayAlarmStatus(), AlarmSensors::displayAllSensorsData(), MonitoringSystem::displayErrorScreen(), MonitoringSystem::displaySensorsScreen(), MonitoringSystem::exitScreen(), MonitoringSystem::getIdAlarm(), MonitoringSystem::getIdSelectedAlarm(), MonitoringSystem::getSelection(), MonitoringSystem::getTypeAlarm(), MonitoringSystem::mainWindowEmployee(), ms_, MonitoringSystem::saveAlarmScreen(), AlarmSensors::saveSensorsData(), sleep(), AlarmSensors::turnOnOffSystem(), and MonitoringSystem::turnOnOffSystemScreen().

Referenced by manageLogin().

Here is the call graph for this function:


GreenHouse::manageEmployee
AlarmSensors::addSensor
MonitoringSystem::create AlarmScreen
MonitoringSystem::delete AlarmScreen
AlarmSensors::deleteSensor
MonitoringSystem::display AlarmsScreen
AlarmSensors::displayAlarm Status
AlarmSensors::displayAll SensorsData
MonitoringSystem::display ErrorScreen
MonitoringSystem::display SensorsScreen
MonitoringSystem::getIdAlarm
MonitoringSystem::getIdSelected Alarm
MonitoringSystem::getSelection
MonitoringSystem::getType Alarm
MonitoringSystem::mainWindow Employee
MonitoringSystem::saveAlarm Screen
AlarmSensors::saveSensorsData
GreenHouse::sleep
AlarmSensors::turnOnOffSystem
MonitoringSystem::turnOn OffSystemScreen
MonitoringSystem::askType Alarm
ScreenHardware::createAlarm Window
MonitoringSystem::askIdAlarm
ScreenHardware::deleteAlarm Window
ScreenHardware::displayAlarms Window
ScreenHardware::displayError Window
MonitoringSystem::exitScreen
ScreenHardware::displaySensors Window
ScreenHardware::mainWindow Employee
ScreenHardware::saveAlarm Window
MonitoringSystem::askInputInt
ScreenHardware::turnOnOff SystemWindow
KeyboardHardware::string Input
KeyboardHardware::askInput
ScreenHardware::exitWindow
MonitoringSystem::inputCorrect

Here is the caller graph for this function:



4.11.3.6 manageGuest()

```
void GreenHouse::manageGuest ( )  [private]
```

Manage the guest.

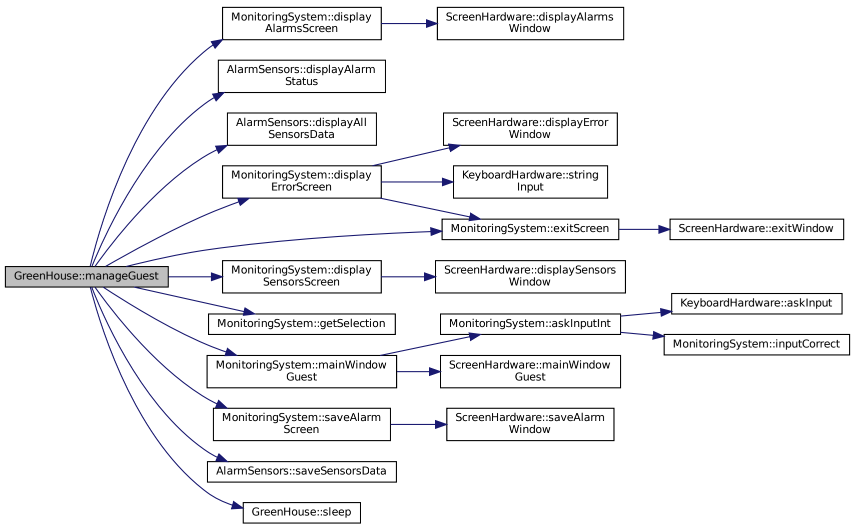
Definition at line 190 of file GreenHouse.cpp.

```
                          {
  bool exit = false;
  // Mostramos la ventana de guest
  do {
    sleep();
    ms_->mainWindowGuest();
    switch (ms_->getSelection()) {
    case 1:
      ms_->displaySensorsScreen();
      alarm_->displayAllSensorsData();
      break;
    case 2:
      ms_->displayAlarmsScreen();
      alarm_->displayAlarmStatus();
      break;
    case 3:
      ms_->saveAlarmScreen();
      alarm_->saveSensorsData();
      break;
    case 4:
      ms_->exitScreen();
      ms_->saveAlarmScreen();
      alarm_->saveSensorsData();
      exit = true;
      break;
    default:
      ms_->displayErrorScreen();
      break;
    }
  } while (!exit);
}
```

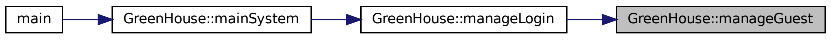
References alarm_, MonitoringSystem::displayAlarmsScreen(), AlarmSensors::displayAlarmStatus(), Alarm↩Sensors::displayAllSensorsData(), MonitoringSystem::displayErrorScreen(), MonitoringSystem::displaySensors↩Screen(), MonitoringSystem::exitScreen(), MonitoringSystem::getSelection(), MonitoringSystem::mainWindow↩Guest(), ms_, MonitoringSystem::saveAlarmScreen(), AlarmSensors::saveSensorsData(), and sleep().

Referenced by manageLogin().

Here is the call graph for this function:


GreenHouse::manageGuest
MonitoringSystem::display AlarmsScreen
ScreenHardware::displayAlarms Window
AlarmSensors::displayAlarm Status
AlarmSensors::displayAll SensorsData
ScreenHardware::displayError Window
MonitoringSystem::display ErrorScreen
KeyboardHardware::string Input
MonitoringSystem::exitScreen
ScreenHardware::exitWindow
MonitoringSystem::display SensorsScreen
ScreenHardware::displaySensors Window
KeyboardHardware::askInput
MonitoringSystem::getSelection
MonitoringSystem::askInputInt
MonitoringSystem::inputCorrect
MonitoringSystem::mainWindow Guest
ScreenHardware::mainWindow Guest
MonitoringSystem::saveAlarm Screen
ScreenHardware::saveAlarm Window
AlarmSensors::saveSensorsData
GreenHouse::sleep

Here is the caller graph for this function:



4.11.3.7 manageLogin()

```
void GreenHouse::manageLogin ( )  [private]
```

Manage the login.

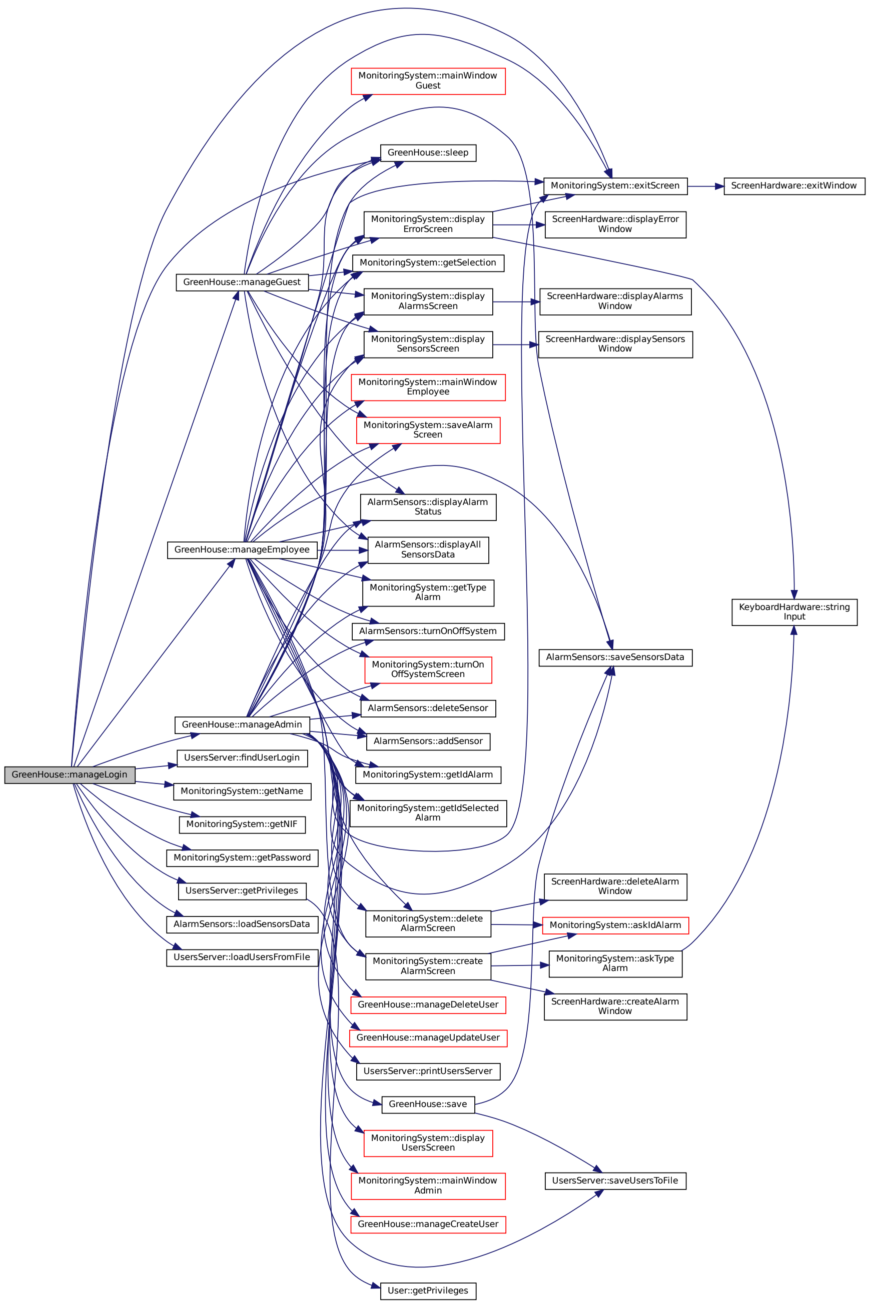
Definition at line 222 of file GreenHouse.cpp.

```
                              {

  // Cargamos los usuarios del archivo
  us_->loadUsersFromFile();
  // Cargamos los sensores del archivo
  alarm_->loadSensorsData();
  // Comprobamos si el usuario y la contraseña son correctos
  if (us_->findUserLogin(ms_->getName(), ms_->getPassword(), ms_->getNIF())) {
    printf("Usuario correcto\n");
    // ahora tengo qeu ver que tipo de usuario es
    // si es admin, employee o guest
    // si es admin
    if (us_->getPrivileges(ms_->getNIF()) == "ADMIN") {
      manageAdmin();

    } else if (us_->getPrivileges(ms_->getNIF()) == "EMPLOYEE") {
      manageEmployee();

    } else {
      manageGuest();
    }

  } else {
    printf("Usuario incorrecto\n");
    sleep();
    ms_->exitScreen();
  }
}
```

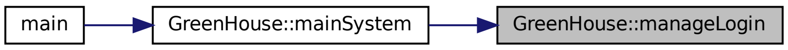
References alarm_, MonitoringSystem::exitScreen(), UsersServer::findUserLogin(), MonitoringSystem::getName(), MonitoringSystem::getNIF(), MonitoringSystem::getPassword(), UsersServer::getPrivileges(), AlarmSensors↩ ::loadSensorsData(), UsersServer::loadUsersFromFile(), manageAdmin(), manageEmployee(), manageGuest(), ms_, sleep(), and us_.

Referenced by mainSystem().

Here is the call graph for this function:


MonitoringSystem::mainWindow Guest
GreenHouse::sleep
MonitoringSystem::exitScreen
ScreenHardware::exitWindow
MonitoringSystem::display ErrorScreen
ScreenHardware::displayError Window
MonitoringSystem::getSelection
GreenHouse::manageGuest
MonitoringSystem::display AlarmsScreen
ScreenHardware::displayAlarms Window
MonitoringSystem::display SensorsScreen
ScreenHardware::displaySensors Window
MonitoringSystem::mainWindow Employee
MonitoringSystem::saveAlarm Screen
AlarmSensors::displayAlarm Status
GreenHouse::manageEmployee
AlarmSensors::displayAll SensorsData
MonitoringSystem::getType Alarm
KeyboardHardware::string Input
AlarmSensors::turnOnOffSystem
AlarmSensors::saveSensorsData
MonitoringSystem::turnOn OffSystemScreen
AlarmSensors::deleteSensor
GreenHouse::manageAdmin
AlarmSensors::addSensor
UsersServer::findUserLogin
GreenHouse::manageLogin
MonitoringSystem::getIdAlarm
MonitoringSystem::getName
MonitoringSystem::getNIF
MonitoringSystem::getIdSelected Alarm
MonitoringSystem::getPassword
ScreenHardware::deleteAlarm Window
UsersServer::getPrivileges
AlarmSensors::loadSensorsData
MonitoringSystem::delete AlarmScreen
MonitoringSystem::askIdAlarm
UsersServer::loadUsersFromFile
MonitoringSystem::create AlarmScreen
MonitoringSystem::askType Alarm
ScreenHardware::createAlarm Window
GreenHouse::manageDeleteUser
GreenHouse::manageUpdateUser
UsersServer::printUsersServer
GreenHouse::save
MonitoringSystem::display UsersScreen
UsersServer::saveUsersToFile
MonitoringSystem::mainWindow Admin
GreenHouse::manageCreateUser
User::getPrivileges

Here is the caller graph for this function:



4.11.3.8 manageUpdateUser()

```
void GreenHouse::manageUpdateUser ( )  [private]
```

Manage the update user.

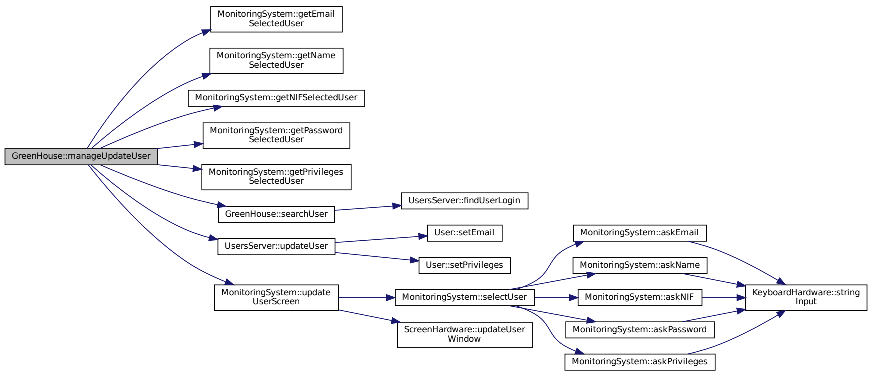
Definition at line 61 of file GreenHouse.cpp.

```
                                   {
  ms_->updateUserScreen();
  if (searchUser(ms_->getNameSelectedUser(), ms_->getPasswordSelectedUser(),
                 ms_->getNIFSelectedUser())) {
    us_->updateUser(ms_->getEmailSelectedUser(), ms_->getNIFSelectedUser(),
                    ms_->getPasswordSelectedUser(),
                    ms_->getPrivilegesSelectedUser(),
                    ms_->getEmailSelectedUser());
  } else {
    printf("Usuario no encontrado\n");
  }
}
```

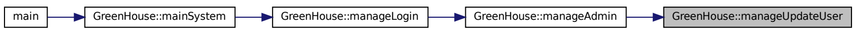
References MonitoringSystem::getEmailSelectedUser(), MonitoringSystem::getNameSelectedUser(), MonitoringSystem::getNIFSelectedUser(), MonitoringSystem::getPasswordSelectedUser(), MonitoringSystem::getPrivilegesSelectedUser(), ms_, searchUser(), UsersServer::updateUser(), MonitoringSystem::updateUserScreen(), and us_.

Referenced by manageAdmin().

Here is the call graph for this function:


GreenHouse::manageUpdateUser
MonitoringSystem::getEmail SelectedUser
MonitoringSystem::getName SelectedUser
MonitoringSystem::getNIFSelectedUser
MonitoringSystem::getPassword SelectedUser
MonitoringSystem::getPrivileges SelectedUser
GreenHouse::searchUser
UsersServer::findUserLogin
UsersServer::updateUser
User::setEmail
User::setPrivileges
MonitoringSystem::update UserScreen
MonitoringSystem::selectUser
ScreenHardware::updateUser Window
MonitoringSystem::askEmail
MonitoringSystem::askName
MonitoringSystem::askNIF
MonitoringSystem::askPassword
MonitoringSystem::askPrivileges
KeyboardHardware::string Input

Here is the caller graph for this function:



4.11.3.9 save()

```
void GreenHouse::save ( )  [private]
```

Load the system.

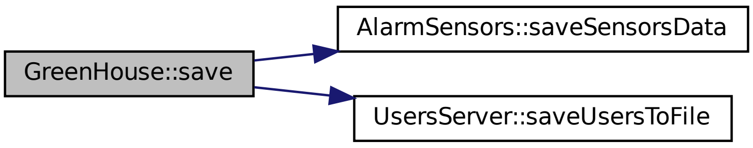
Definition at line 76 of file GreenHouse.cpp.

```
                     {
  us_->saveUsersToFile();
  alarm_->saveSensorsData();
}
```

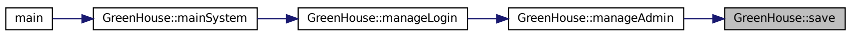
References alarm_, AlarmSensors::saveSensorsData(), UsersServer::saveUsersToFile(), and us_.

Referenced by manageAdmin().

Here is the call graph for this function:



Here is the caller graph for this function:



4.11.3.10 searchUser()

```
bool GreenHouse::searchUser (
             std::string name,
             std::string password,
             std::string nif )  [private]
```

Manage search user.

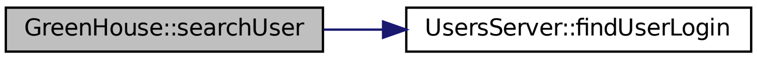
Definition at line 38 of file GreenHouse.cpp.

```
                                         {
  return us_->findUserLogin(name, password, nif);
}
```

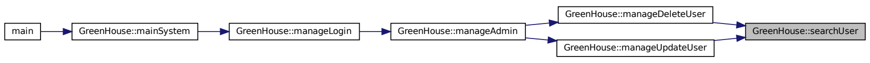
References UsersServer::findUserLogin(), and us_.

Referenced by manageDeleteUser(), and manageUpdateUser().

Here is the call graph for this function:



Here is the caller graph for this function:



4.11.3.11 sleep()

```
void GreenHouse::sleep ( )  [private]
```

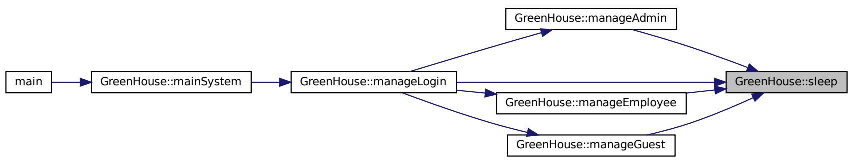
Save the system.

Definition at line 74 of file GreenHouse.cpp.

```
74 { system("sleep 5"); }
```

Referenced by manageAdmin(), manageEmployee(), manageGuest(), and manageLogin().

Here is the caller graph for this function:



4.11.3.12 startSystem()

```
void GreenHouse::startSystem ( )
```

Start the system.

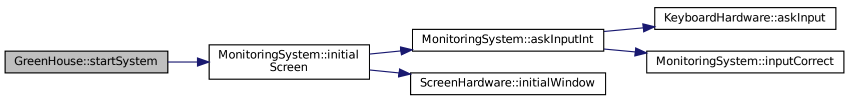
Definition at line 33 of file GreenHouse.cpp.

```
33                              {
34   // Mensaje de bienvenida del invernadero
35   ms_->initialScreen();
36 }
```

References MonitoringSystem::initialScreen(), and ms_.

Referenced by main().

Here is the call graph for this function:


GreenHouse::startSystem
MonitoringSystem::initial Screen
MonitoringSystem::askInputInt
ScreenHardware::initialWindow
KeyboardHardware::askInput
MonitoringSystem::inputCorrect

Here is the caller graph for this function:

main
GreenHouse::startSystem

4.11.4 Member Data Documentation

4.11.4.1 alarm_

```
AlarmSensors* GreenHouse::alarm_  [private]
```

This attribute is the AlarmSensors object (set of pointers to sensors).

Definition at line 98 of file GreenHouse.h.

Referenced by manageAdmin(), manageEmployee(), manageGuest(), manageLogin(), save(), and ∼Green↩ House().

4.11.4.2 ms_

`MonitoringSystem* GreenHouse::ms_ [private]`

This attribute is the MonitoringSystem object.

Definition at line 104 of file GreenHouse.h.

Referenced by mainSystem(), manageAdmin(), manageCreateUser(), manageDeleteUser(), manageEmployee(), manageGuest(), manageLogin(), manageUpdateUser(), startSystem(), and ~GreenHouse().

4.11.4.3 us_

`UsersServer* GreenHouse::us_ [private]`

This attribute is the UsersServer object (set of pointers to users).

Definition at line 110 of file GreenHouse.h.

Referenced by manageAdmin(), manageCreateUser(), manageDeleteUser(), manageLogin(), manageUpdate←User(), save(), searchUser(), and ~GreenHouse().

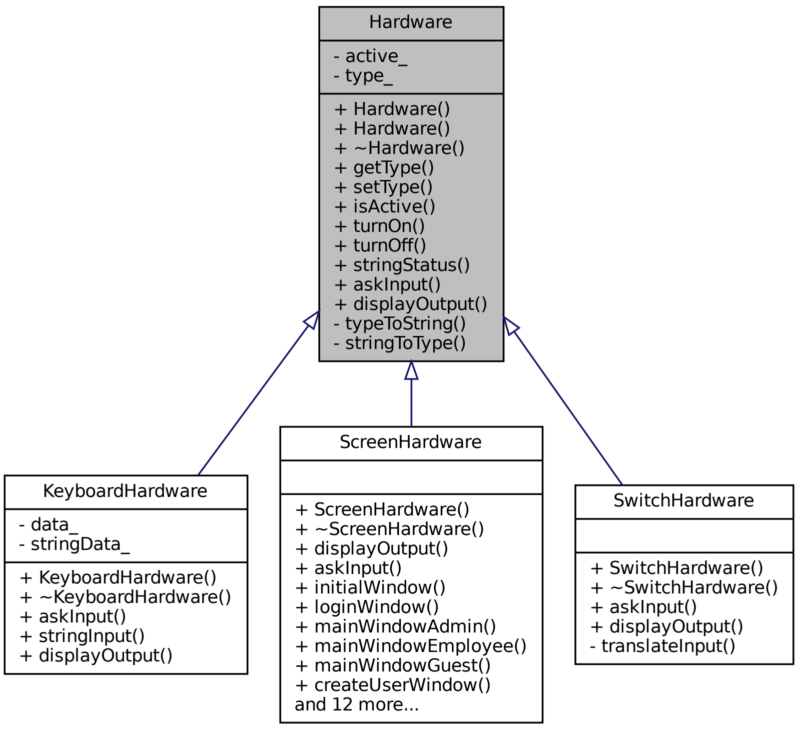
The documentation for this class was generated from the following files:

- src/GreenHouse.h
- src/GreenHouse.cpp

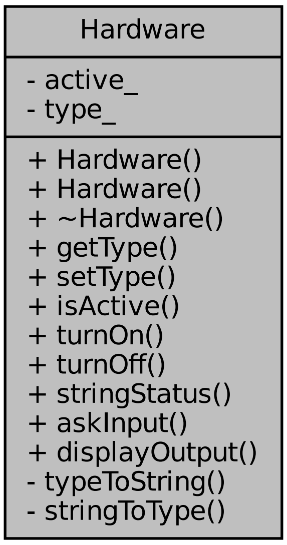
4.12 Hardware Class Reference

`#include <Hardware.h>`

Inheritance diagram for Hardware:



Collaboration diagram for Hardware:



Public Types

- enum Types_Hardware { NONE , SCREEN , KEYBOARD , SWITCH }

    Enum of the types of hardware.

Public Member Functions

- Hardware ()

    Construct a new Hardware object.
- Hardware (bool active, Types_Hardware type)

    Construct a new Hardware object.
- virtual ∼Hardware ()

    Destroy the Hardware object.
- std::string getType () const

    Get the Type object.
- void setType (std::string newtype)

    Set the Type object.
- bool isActive () const

    If the hardware is active.
- void turnOn ()

    Turn on the hardware.
- void turnOff ()

    Turn off the hardware.
- std::string stringStatus () const

    This method returns if the hardware is active or not in a string.
- virtual int askInput ()

    This method asks the user for an input.
- virtual void displayOutput () const

    This method displays the output of the hardware.

Private Member Functions

- std::string typeToString (Types_Hardware type) const

    This method converts the type of hardware to a string.
- Types_Hardware stringToType (std::string type) const

    This method converts the string to a type of hardware.

Private Attributes

- bool active_

    This atrribute is the status of the hardware.
- Types_Hardware type_

    This attribute is the type of the hardware.

4.12.1 Detailed Description

Definition at line 15 of file Hardware.h.

4.12.2 Member Enumeration Documentation

4.12.2.1 Types_Hardware

```
enum Hardware::Types_Hardware
```

Enum of the types of hardware.

Enumerator

NONE	
SCREEN	
KEYBOARD	
SWITCH	

Definition at line 21 of file Hardware.h.

```
21 { NONE, SCREEN, KEYBOARD, SWITCH };
```

4.12.3 Constructor & Destructor Documentation

4.12.3.1 Hardware() [1/2]

```
Hardware::Hardware ( )
```

Construct a new Hardware object.

Definition at line 7 of file Hardware.cpp.

```
: active_(false), type_(Types_Hardware::NONE) {}
```

4.12.3.2 Hardware() [2/2]

```
Hardware::Hardware (
            bool active,
            Types_Hardware type ) [explicit]
```

Construct a new Hardware object.

Parameters

active	
type	

Definition at line 8 of file Hardware.cpp.

```
    : active_(active), type_(type) {}
```

4.12.3.3 ∼Hardware()

```
Hardware::∼Hardware ( ) [virtual]
```

Destroy the Hardware object.

Definition at line 10 of file Hardware.cpp.

```
{}
```

4.12.4 Member Function Documentation

4.12.4.1 askInput()

```
int Hardware::askInput ( ) [virtual]
```

This method asks the user for an input.

Returns

int

Reimplemented in SwitchHardware, ScreenHardware, and KeyboardHardware.

Definition at line 34 of file Hardware.cpp.

```
                          {
  return 0;
  // esta funcion sera definida en clases hijas
  // pero la idea es qeu muestre un mensaje estilo Pantalla: (Aqui viene el
  // input del usuario)
}
```

4.12.4.2 displayOutput()

```
void Hardware::displayOutput ( ) const  [virtual]
```

This method displays the output of the hardware.

Reimplemented in SwitchHardware, ScreenHardware, and KeyboardHardware.

Definition at line 41 of file Hardware.cpp.

```
                                  {
  // Esta funcion sera definida en las clases hijas
  // Por ahora mostramos un mensaje generico
  cout « "Hardware cannot display output for this type" « endl;
}
```

4.12.4.3 getType()

```
std::string Hardware::getType ( ) const
```

Get the Type object.

Returns

std::string

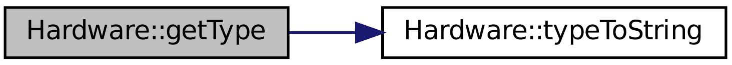
Definition at line 12 of file Hardware.cpp.

```
{ return typeToString(type_); }
```

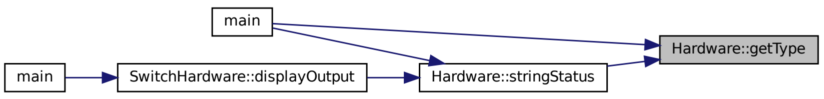
References type_, and typeToString().

Referenced by main(), and stringStatus().

Here is the call graph for this function:



Here is the caller graph for this function:



4.12.4.4 isActive()

```
bool Hardware::isActive ( ) const
```

If the hardware is active.

Returns

true if the hardware is active

false if the hardware is not active

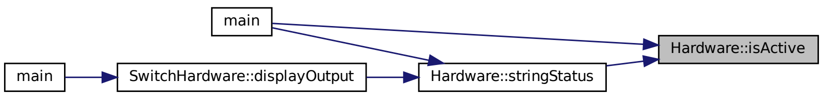
Definition at line 16 of file Hardware.cpp.

```
16 { return active_; }
```

References active_.

Referenced by main(), and stringStatus().

Here is the caller graph for this function:



4.12.4.5 setType()

```
void Hardware::setType (
             std::string newtype )
```

Set the Type object.

Parameters

newtype	

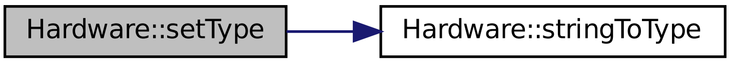
Definition at line 14 of file Hardware.cpp.

```
14 { type_ = stringToType(newtype); }
```

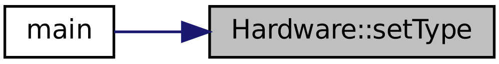
References stringToType(), and type_.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.12.4.6 stringStatus()

```
std::string Hardware::stringStatus ( ) const
```

This method returns if the hardware is active or not in a string.

Returns

std::string

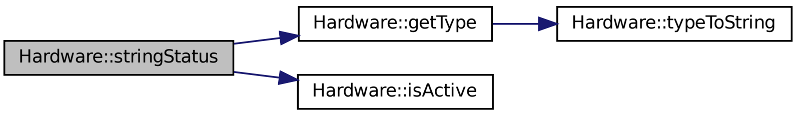
Definition at line 22 of file Hardware.cpp.

```
                                          {
  std::string status;
  if (isActive()) {
    status = "ON";
  } else {
    status = "OFF";
  }
  status += " - ";
  status += getType();
  return status;
}
```

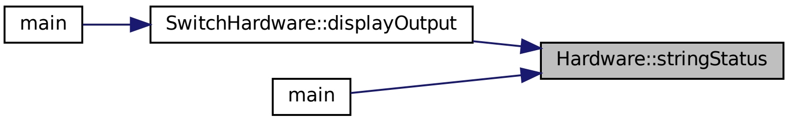
References getType(), and isActive().

Referenced by SwitchHardware::displayOutput(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.12.4.7 stringToType()

```
Hardware::Types_Hardware Hardware::stringToType (
             std::string type ) const  [private]
```

This method converts the string to a type of hardware.

Parameters

type	

Returns

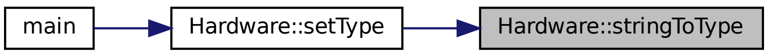
Types_Hardware

Definition at line 60 of file Hardware.cpp.

```
                                                                    {
    if (type == "SCREEN") {
      return Hardware::Types_Hardware::SCREEN;
    } else if (type == "KEYBOARD") {
      return Hardware::Types_Hardware::KEYBOARD;
    } else if (type == "SWITCH") {
      return Hardware::Types_Hardware::SWITCH;
    } else {
      return Hardware::Types_Hardware::NONE;
    }
  }
```

Referenced by setType().

Here is the caller graph for this function:



4.12.4.8 turnOff()

```
void Hardware::turnOff ( )
```

Turn off the hardware.

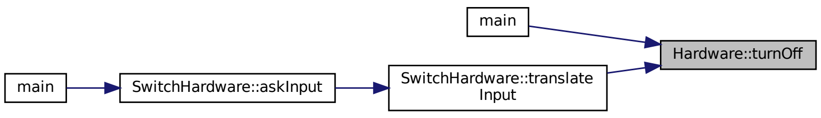
Definition at line 20 of file Hardware.cpp.

```
20 { active_ = false; }
```

References active_.

Referenced by main(), and SwitchHardware::translateInput().

Here is the caller graph for this function:



4.12.4.9 turnOn()

```
void Hardware::turnOn ( )
```

Turn on the hardware.

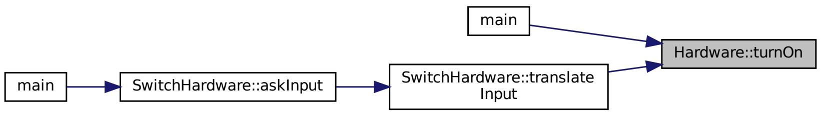
Definition at line 18 of file Hardware.cpp.

```
18 { active_ = true; }
```

References active_.

Referenced by main(), and SwitchHardware::translateInput().

Here is the caller graph for this function:



4.12.4.10 typeToString()

```
std::string Hardware::typeToString (
            Types_Hardware type ) const  [private]
```

This method converts the type of hardware to a string.

Parameters

type	

Returns

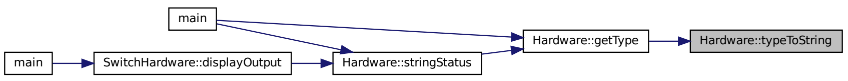
std::string

Definition at line 47 of file Hardware.cpp.

```
                                                        {
  switch (type) {
  case Types_Hardware::SCREEN:
    return "SCREEN";
  case Types_Hardware::KEYBOARD:
    return "KEYBOARD";
  case Types_Hardware::SWITCH:
    return "SWITCH";
  default:
    return "None";
  }
}
```

Referenced by getType().

Here is the caller graph for this function:



4.12.5 Member Data Documentation

4.12.5.1 active_

```
bool Hardware::active_  [private]
```

This atrribute is the status of the hardware.

Definition at line 95 of file Hardware.h.

Referenced by isActive(), turnOff(), and turnOn().

4.12.5.2 type_

```
Types_Hardware Hardware::type_  [private]
```

This attribute is the type of the hardware.

Definition at line 100 of file Hardware.h.

Referenced by getType(), and setType().

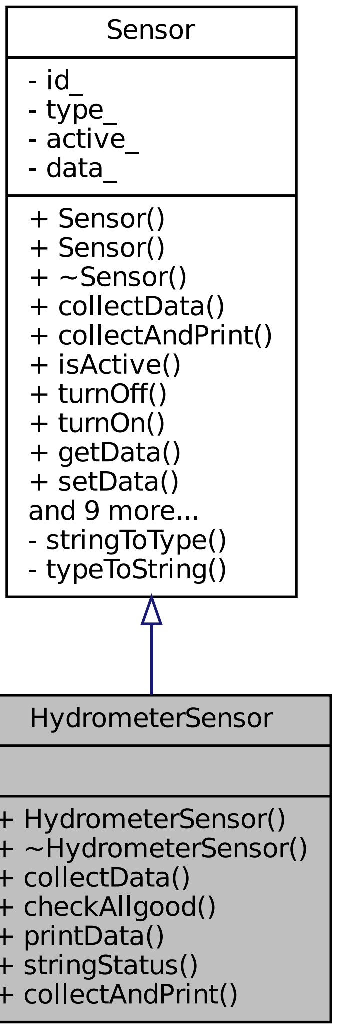
The documentation for this class was generated from the following files:

- src/Hardware.h
- src/Hardware.cpp

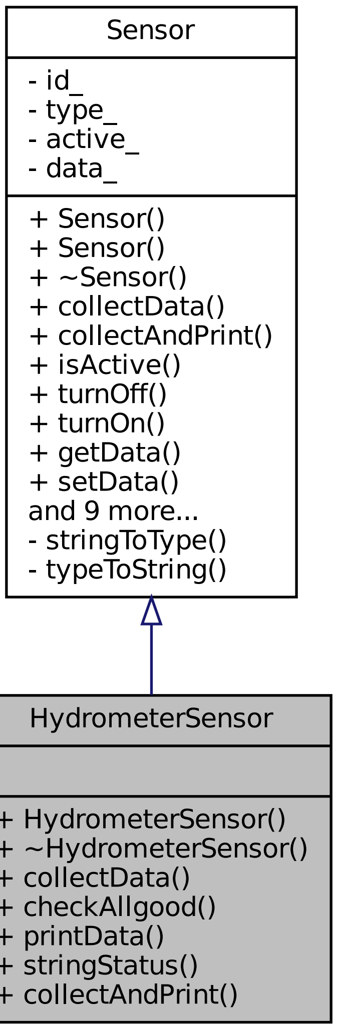
4.13 HydrometerSensor Class Reference

```
#include <HydrometerSensor.h>
```

Inheritance diagram for HydrometerSensor:



Collaboration diagram for HydrometerSensor:



Public Member Functions

- HydrometerSensor (int id, bool active)

  Construct a new Hydrometer Sensor object.
- ∼HydrometerSensor () override

  Destroy the Hydrometer Sensor object.
- void collectData () override

  Collect data of the Hydrometer Sensor.
- bool checkAllgood () const override

  Check if the Hydrometer Sensor is working properly.
- void printData () const override

  Print the data of the Hydrometer Sensor.
- std::string stringStatus () const

Collect and print the data of the Hydrometer Sensor.

- void collectAndPrint ()

  Collect and print the data of the Hydrometer Sensor.

Friends

- std::ostream & operator<< (std::ostream &os, const HydrometerSensor &sensor)

  Get the Data object.

Additional Inherited Members

4.13.1 Detailed Description

Definition at line 15 of file HydrometerSensor.h.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 HydrometerSensor()

```
HydrometerSensor::HydrometerSensor (
              int id,
              bool active )  [explicit]
```

Construct a new Hydrometer Sensor object.

Parameters

id	
active	

Returns

HydrometerSensor object

Definition at line 9 of file HydrometerSensor.cpp.

```
      : Sensor(id, Sensor::Types::HYDROMETER, active) {}
```

4.13.2.2 ∼HydrometerSensor()

```
HydrometerSensor::∼HydrometerSensor ( )  [override]
```

Destroy the Hydrometer Sensor object.

Definition at line 12 of file HydrometerSensor.cpp.

```
{}
```

4.13.3 Member Function Documentation

4.13.3.1 checkAllgood()

`bool HydrometerSensor::checkAllgood ( ) const [override], [virtual]`

Check if the Hydrometer Sensor is working properly.

Returns

true if the Hydrometer Sensor is working properly

false if the Hydrometer Sensor is not working properly

Reimplemented from Sensor.

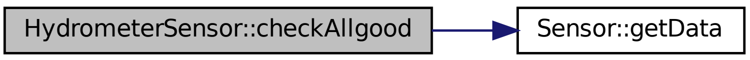
Definition at line 25 of file HydrometerSensor.cpp.

```
25                                {
26   float data = Sensor::getData();
27   // Reading between 55-85 is considered good
28   if (data >= 52 && data <= 88) {
29     return true;
30   } else {
31     return false;
32   }
33 }
```

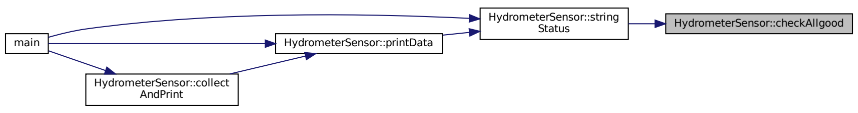
References Sensor::getData().

Referenced by stringStatus().

Here is the call graph for this function:



Here is the caller graph for this function:


main
HydrometerSensor::collectAndPrint
HydrometerSensor::printData
HydrometerSensor::stringStatus
HydrometerSensor::checkAllgood

4.13.3.2 collectAndPrint()

```
void HydrometerSensor::collectAndPrint ( )  [virtual]
```

Collect and print the data of the Hydrometer Sensor.

Reimplemented from Sensor.

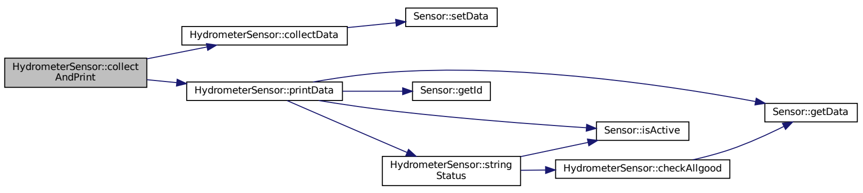
Definition at line 65 of file HydrometerSensor.cpp.

```
65                                            {
66   collectData();
67   printData();
68 }
```

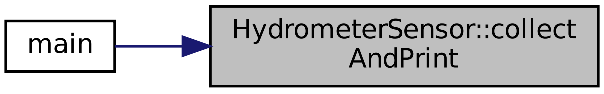
References collectData(), and printData().

Referenced by main().

Here is the call graph for this function:


Sensor::setData
HydrometerSensor::collectData
HydrometerSensor::collect AndPrint
HydrometerSensor::printData
Sensor::getId
Sensor::getData
Sensor::isActive
HydrometerSensor::string Status
HydrometerSensor::checkAllgood

Here is the caller graph for this function:



4.13.3.3 collectData()

```
void HydrometerSensor::collectData ( )  [override], [virtual]
```

Collect data of the Hydrometer Sensor.

This method collects the data of the Hydrometer Sensor and stores it in the data attribute.

Reimplemented from Sensor.

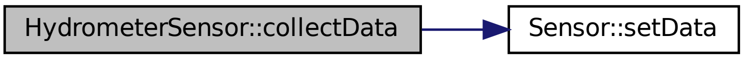
Definition at line 14 of file HydrometerSensor.cpp.

```
                                  {
    // Generate random hydrometer reading between 50-90
    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_int_distribution<> dis(50, 90);
    int reading = dis(gen);

    // Set data
    Sensor::setData(reading);
}
```

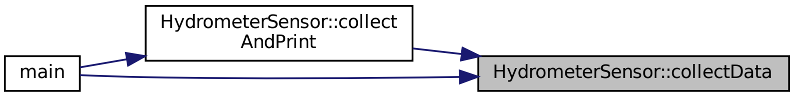
References Sensor::setData().

Referenced by collectAndPrint(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:


main
HydrometerSensor::collect AndPrint
HydrometerSensor::collectData

4.13.3.4 printData()

`void HydrometerSensor::printData ( ) const [override], [virtual]`

Print the data of the Hydrometer Sensor.

Reimplemented from Sensor.

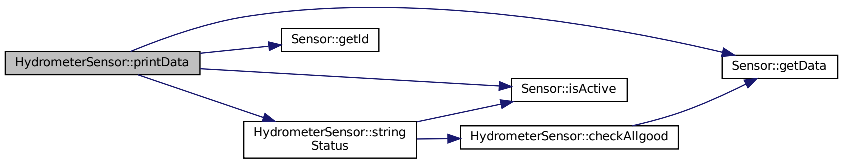
Definition at line 52 of file HydrometerSensor.cpp.

```
                                      {
    if (Sensor::isActive()) {
      std::cout « "Hydrometer Sensor with "
                « "ID: " « Sensor::getId() « " - Data: " « Sensor::getData()
                « " % "
                « "- Status: " « stringStatus() « endl;

    } else {
      cout « "Hydrometer Sensor ID: " « Sensor::getId() « " - INACTIVE"
           « endl;
    }
}
```

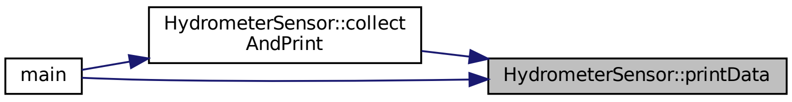
References Sensor::getData(), Sensor::getId(), Sensor::isActive(), and stringStatus().

Referenced by collectAndPrint(), and main().

Here is the call graph for this function:


HydrometerSensor::printData
Sensor::getId
Sensor::isActive
Sensor::getData
HydrometerSensor::string Status
HydrometerSensor::checkAllgood

Here is the caller graph for this function:


main
HydrometerSensor::collect AndPrint
HydrometerSensor::printData

4.13.3.5 stringStatus()

```
std::string HydrometerSensor::stringStatus ( ) const
```

Collect and print the data of the Hydrometer Sensor.

Returns

std::string of the status of the Hydrometer Sensor

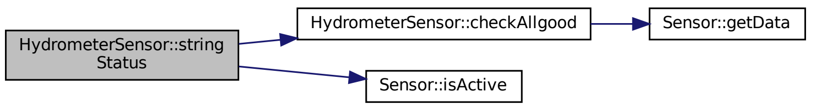
Definition at line 40 of file HydrometerSensor.cpp.

```
                                                      {
  if (Sensor::isActive()) {
    if (this->checkAllgood()) {
      return "ACTIVE - GOOD STATUS";
    } else {
      return "ACTIVE - BAD STATUS";
    }
  } else {
    return "INACTIVE";
  }
}
```

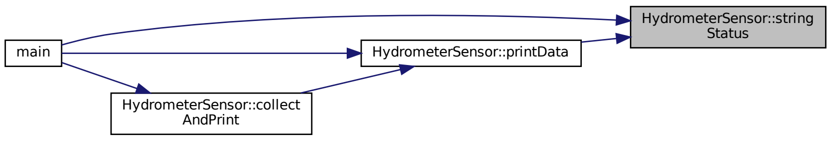
References checkAllgood(), and Sensor::isActive().

Referenced by main(), and printData().

Here is the call graph for this function:


HydrometerSensor::string Status
HydrometerSensor::checkAllgood
Sensor::getData
Sensor::isActive

Here is the caller graph for this function:


main
HydrometerSensor::printData
HydrometerSensor::string Status
HydrometerSensor::collect AndPrint

4.13.4 Friends And Related Function Documentation

4.13.4.1 operator<<

```
std::ostream& operator<< (
              std::ostream & os,
              const HydrometerSensor & sensor )  [friend]
```

Get the Data object.

Returns

double

Definition at line 35 of file HydrometerSensor.cpp.

```
                                                                          {
    sensor.printData();
    return os;
 }
```

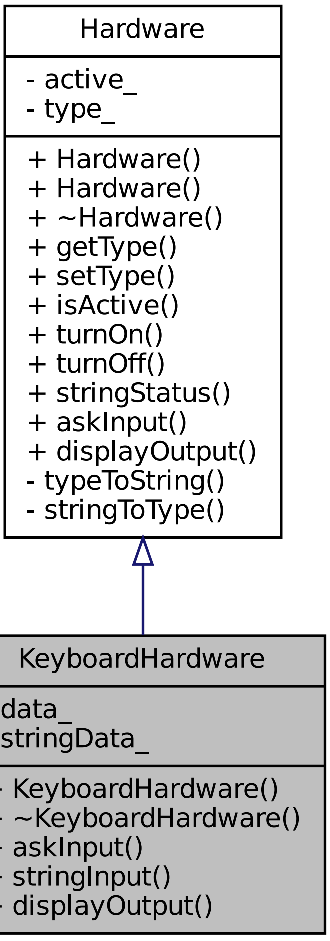
The documentation for this class was generated from the following files:

- src/HydrometerSensor.h
- src/HydrometerSensor.cpp

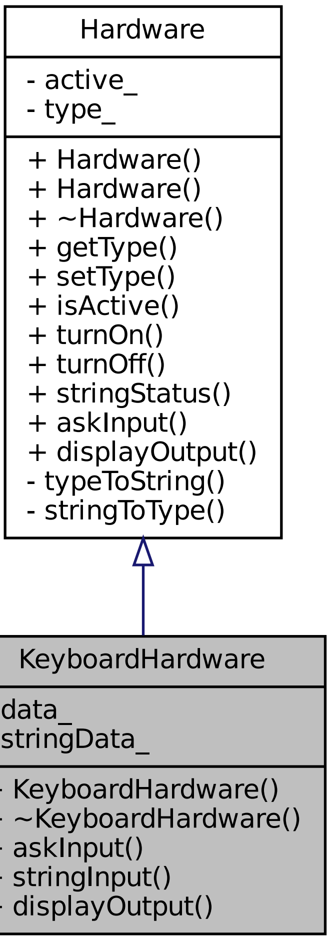
4.14 KeyboardHardware Class Reference

```
#include <KeyboardHardware.h>
```

Inheritance diagram for KeyboardHardware:



Collaboration diagram for KeyboardHardware:



Public Member Functions

- KeyboardHardware (bool active)

  Construct a new Keyboard Hardware object.
- ∼KeyboardHardware () override

  Destroy the Keyboard Hardware object.
- int askInput () override

  Ask for an input to the user.
- std::string stringInput ()

  Ask for a string input to the user.
- void displayOutput () const override

  Display the output of the Keyboard Hardware.

Private Attributes

- int data_

  The int data of the Keyboard Hardware.
- std::string stringData_

  The string data of the Keyboard Hardware.

Additional Inherited Members

4.14.1 Detailed Description

Definition at line 14 of file KeyboardHardware.h.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 KeyboardHardware()

```
KeyboardHardware::KeyboardHardware (
              bool active )  [explicit]
```

Construct a new Keyboard Hardware object.

Parameters

active	

Returns

KeyboardHardware object

Definition at line 9 of file KeyboardHardware.cpp.
```
    : Hardware(active, Hardware::Types_Hardware::KEYBOARD) {
  data_ = 0;
  stringData_ = "";
}
```

References data_, and stringData_.

4.14.2.2 ∼KeyboardHardware()

```
KeyboardHardware::∼KeyboardHardware ( )  [override]
```

Destroy the Keyboard Hardware object.

Definition at line 15 of file KeyboardHardware.cpp.
```
{}
```

4.14.3 Member Function Documentation

4.14.3.1 askInput()

```
int KeyboardHardware::askInput ( )  [override], [virtual]
```

Ask for an input to the user.

Returns

int

Reimplemented from Hardware.

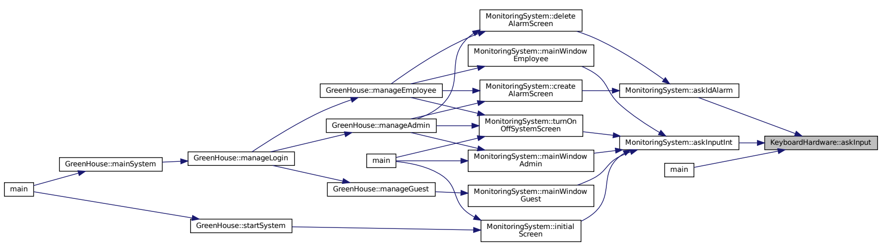
Definition at line 17 of file KeyboardHardware.cpp.

```
17                                 {
18   bool exit = false;
19   int input;
20
21   while (not exit) {
22     std::cout « "- Keyboard waiting for input (integer): ";
23     std::cin » input;
24
25     // Verificar si la entrada es un número
26     if (std::cin.fail()) {
27       std::cin.clear(); // Restablecer el estado de cin
28       std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
29                       '\n'); // Limpiar el buffer de entrada
30       std::cout « "Invalid input. Please enter an integer that corresponse to "
31                    "one of the options"
32                 « std::endl;
33     } else {
34       exit = true;
35       data_ = input; // Guardar el valor ingresado en data_
36     }
37   }
38
39   return data_;
40 }
```

References data_.

Referenced by MonitoringSystem::askIdAlarm(), MonitoringSystem::askInputInt(), and main().

Here is the caller graph for this function:


MonitoringSystem::deleteAlarmScreen
MonitoringSystem::mainWindowEmployee
MonitoringSystem::createAlarmScreen
MonitoringSystem::turnOnOffSystemScreen
MonitoringSystem::mainWindowAdmin
MonitoringSystem::mainWindowGuest
MonitoringSystem::initialScreen
MonitoringSystem::askIdAlarm
MonitoringSystem::askInputInt
KeyboardHardware::askInput
GreenHouse::manageEmployee
GreenHouse::manageAdmin
GreenHouse::manageLogin
GreenHouse::manageGuest
GreenHouse::mainSystem
GreenHouse::startSystem
main
main
main

4.14.3.2 displayOutput()

```
void KeyboardHardware::displayOutput ( ) const  [override], [virtual]
```

Display the output of the Keyboard Hardware.

Reimplemented from Hardware.

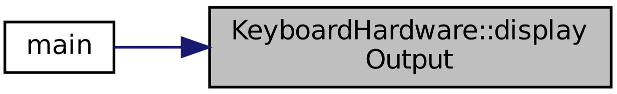
Definition at line 50 of file KeyboardHardware.cpp.

```
50                                              {
51   std::cout « "-Last intput(integer) of the keyboard: " « data_ « std::endl;
52   std::cout « "-Last intput(string) of the keyboard: " « stringData_
53             « std::endl;
54 }
```

References data_, and stringData_.

Referenced by main().

Here is the caller graph for this function:



4.14.3.3 stringInput()

```
std::string KeyboardHardware::stringInput ( )
```

Ask for a string input to the user.

Returns

std::string

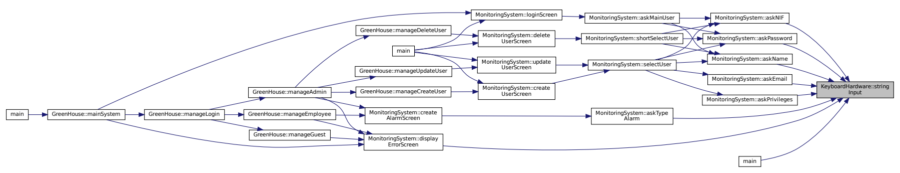
Definition at line 42 of file KeyboardHardware.cpp.

```
42                                           {
43   std::cout « "- Keyboard waiting for input (string): ";
44   std::string input;
45   std::cin » input;
46   stringData_ = input;
47   return stringData_;
48 }
```

References stringData_.

Referenced by MonitoringSystem::askEmail(), MonitoringSystem::askName(), MonitoringSystem::askNIF(), MonitoringSystem::askPassword(), MonitoringSystem::askPrivileges(), MonitoringSystem::askTypeAlarm(), MonitoringSystem::displayErrorScreen(), and main().

Here is the caller graph for this function:


main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageEmployee
GreenHouse::manageGuest
GreenHouse::manageDeleteUser
GreenHouse::manageUpdateUser
GreenHouse::manageCreateUser
main
MonitoringSystem::loginScreen
MonitoringSystem::delete UserScreen
MonitoringSystem::update UserScreen
MonitoringSystem::create UserScreen
MonitoringSystem::create AlarmScreen
MonitoringSystem::display ErrorScreen
MonitoringSystem::askMainUser
MonitoringSystem::shortSelectUser
MonitoringSystem::selectUser
MonitoringSystem::askType Alarm
MonitoringSystem::askNIF
MonitoringSystem::askPassword
MonitoringSystem::askName
MonitoringSystem::askEmail
MonitoringSystem::askPrivileges
main
KeyboardHardware::string Input

4.14.4 Member Data Documentation

4.14.4.1 data_

```
int KeyboardHardware::data_  [private]
```

The int data of the Keyboard Hardware.

Definition at line 52 of file KeyboardHardware.h.

Referenced by askInput(), displayOutput(), and KeyboardHardware().

4.14.4.2 stringData_

```
std::string KeyboardHardware::stringData_  [private]
```

The string data of the Keyboard Hardware.

Definition at line 57 of file KeyboardHardware.h.

Referenced by displayOutput(), KeyboardHardware(), and stringInput().

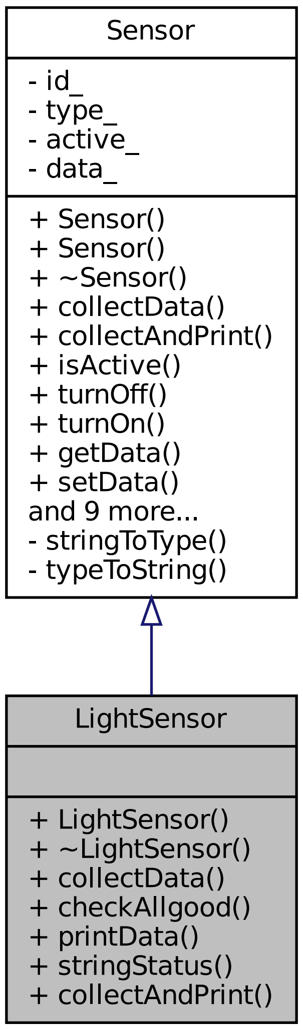
The documentation for this class was generated from the following files:

- src/KeyboardHardware.h
- src/KeyboardHardware.cpp

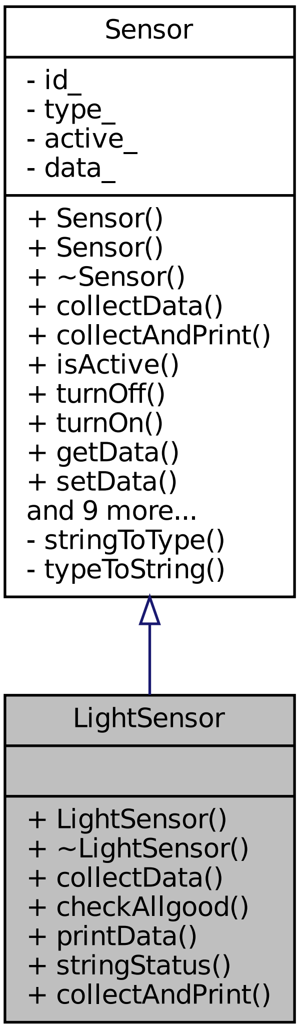
4.15 LightSensor Class Reference

```
#include <LightSensor.h>
```

Inheritance diagram for LightSensor:



Collaboration diagram for LightSensor:



Public Member Functions

- LightSensor (int id, bool active)

  Construct a new Light Sensor object.
- ∼LightSensor () override

  Destroy the Light Sensor object.
- void collectData () override

  Collect data of the Light Sensor.
- bool checkAllgood () const override

  Check if the Light Sensor is working properly.
- void printData () const override

  Print the data of the Light Sensor.
- std::string stringStatus () const

Collect and print the data of the Light Sensor.
- void collectAndPrint ()

  Collect and print the data of the Light Sensor.

Friends

- std::ostream & operator<< (std::ostream &os, const LightSensor &sensor)

  Get the Light object.

Additional Inherited Members

4.15.1 Detailed Description

Definition at line 15 of file LightSensor.h.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 LightSensor()

```
LightSensor::LightSensor (
              int id,
              bool active )   [explicit]
```

Construct a new Light Sensor object.

Parameters

id	
active	

Returns

LightSensor object

Definition at line 8 of file LightSensor.cpp.
```
9     : Sensor(id, Sensor::Types::LIGHT_SENSOR, active) {}
```

4.15.2.2 ~LightSensor()

```
LightSensor::~LightSensor ( )   [override]
```

Destroy the Light Sensor object.

Definition at line 11 of file LightSensor.cpp.
```
11 {}
```

4.15.3 Member Function Documentation

4.15.3.1 checkAllgood()

```
bool LightSensor::checkAllgood ( ) const  [override], [virtual]
```

Check if the Light Sensor is working properly.

Returns

true if the Light Sensor is working properly

false if the Light Sensor is not working properly

Reimplemented from Sensor.

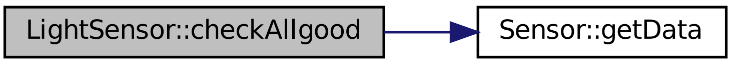
Definition at line 23 of file LightSensor.cpp.

```
                                         {
  float data = Sensor::getData();

  if (data >= 300 && data <= 3900) {
    return true;
  } else {
    return false;
  }
}
```

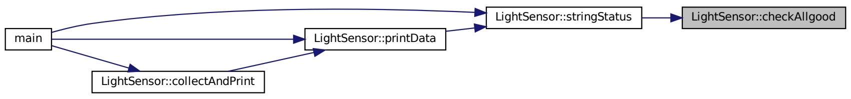
References Sensor::getData().

Referenced by stringStatus().

Here is the call graph for this function:



Here is the caller graph for this function:


main
LightSensor::stringStatus
LightSensor::checkAllgood
LightSensor::printData
LightSensor::collectAndPrint

4.15.3.2 collectAndPrint()

```
void LightSensor::collectAndPrint ( )  [virtual]
```

Collect and print the data of the Light Sensor.

Reimplemented from Sensor.

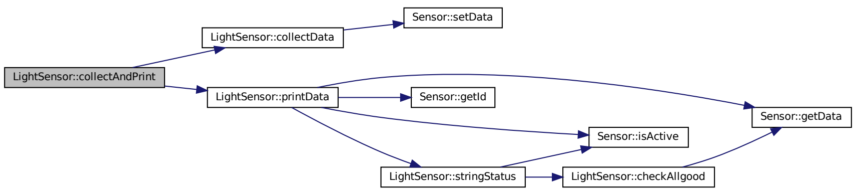
Definition at line 61 of file LightSensor.cpp.

```
61                                  {
62   collectData();
63   printData();
64 }
```

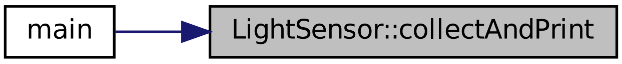
References collectData(), and printData().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.15.3.3 collectData()

```
void LightSensor::collectData ( )  [override], [virtual]
```

Collect data of the Light Sensor.

This method collects the data of the Light Sensor and stores it in the data attribute.

Reimplemented from Sensor.

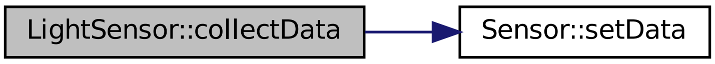
Definition at line 13 of file LightSensor.cpp.

```
13                              {
    // Medido en lux entre 200-4000 lux
    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_int_distribution<> dis(200, 4000);
    int reading = dis(gen);

    Sensor::setData(reading);
}
```

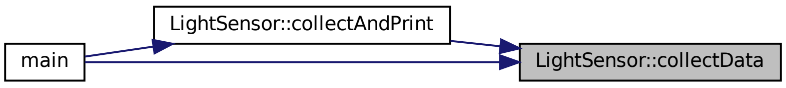
References Sensor::setData().

Referenced by collectAndPrint(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.15.3.4 printData()

```
void LightSensor::printData ( ) const  [override], [virtual]
```

Print the data of the Light Sensor.

Reimplemented from Sensor.

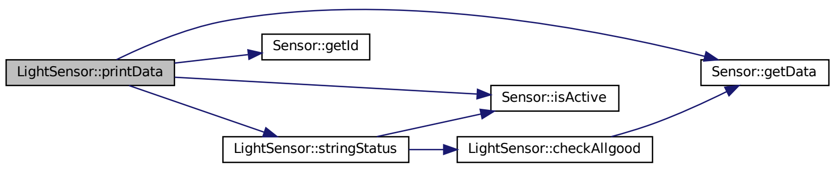
Definition at line 38 of file LightSensor.cpp.

```
                                   {
    if (Sensor::isActive()) {
      std::cout « "Light Sensor with "
                « "ID: " « Sensor::getId() « " - Data: " « Sensor::getData()
                « " lux - Status: " « stringStatus() « endl;
    } else {
      std::cout « "Light Sensor ID: " « Sensor::getId()
                « " - Status: " « stringStatus() « endl;
    }
}
```

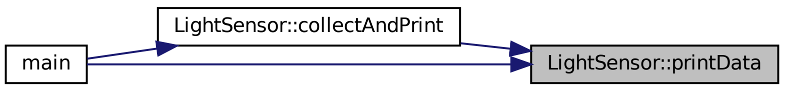
References Sensor::getData(), Sensor::getId(), Sensor::isActive(), and stringStatus().

Referenced by collectAndPrint(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.15.3.5 stringStatus()

```
std::string LightSensor::stringStatus ( ) const
```

Collect and print the data of the Light Sensor.

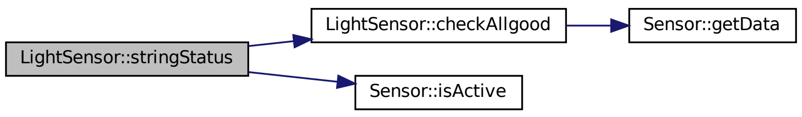
Definition at line 49 of file LightSensor.cpp.

```
                                             {
  if (Sensor::isActive()) {
    if (this->checkAllgood()) {
      return "ACTIVE - GOOD STATUS";
    } else {
      return "ACTIVE - BAD STATUS";
    }
  } else {
    return "INACTIVE";
  }
}
```

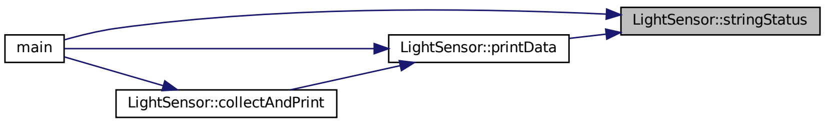
References checkAllgood(), and Sensor::isActive().

Referenced by main(), and printData().

Here is the call graph for this function:



Here is the caller graph for this function:



4.15.4 Friends And Related Function Documentation

4.15.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const LightSensor & sensor )  [friend]
```

Get the Light object.

Returns

int

Definition at line 33 of file LightSensor.cpp.

```
{
  sensor.printData();
  return os;
}
```

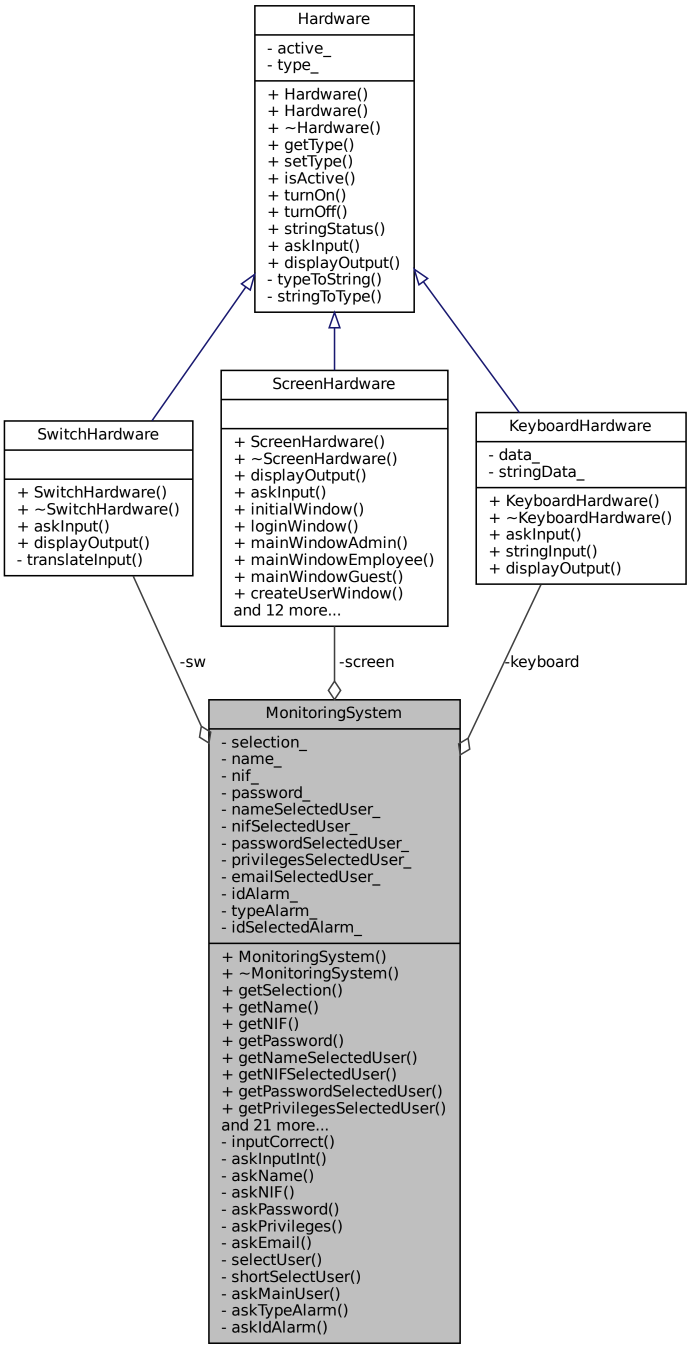
The documentation for this class was generated from the following files:

- src/LightSensor.h
- src/LightSensor.cpp

4.16 MonitoringSystem Class Reference

```
#include <MonitoringSystem.h>
```

Collaboration diagram for MonitoringSystem:



Public Member Functions

- MonitoringSystem (ScreenHardware ∗screen, KeyboardHardware ∗keyboard, SwitchHardware ∗sw)

    Construct a new Monitoring System object.
- ∼MonitoringSystem ()

    Destroy the Monitoring System object.
- int getSelection ()

Get the Selection object.
- std::string getName ()

  Get the Name object.
- std::string getNIF ()

  Get the NIF object.
- std::string getPassword ()

  Get the Password object.
- std::string getNameSelectedUser ()

  Get the Name Selected User object.
- std::string getNIFSelectedUser ()

  Get the NIF Selected User object.
- std::string getPasswordSelectedUser ()

  Get the Password Selected User object.
- std::string getPrivilegesSelectedUser ()

  Get the Privileges Selected User object.
- std::string getEmailSelectedUser ()

  Get the Email Selected User object.
- int getIdAlarm ()

  Get the Id Alarm object.
- std::string getTypeAlarm ()

  Get the Type Alarm object.
- int getIdSelectedAlarm ()

  Get the Id Selected Alarm object.
- void initialScreen ()

  This method initializes the screen and keyboard.
- void exitScreen ()

  This method exits the screen.
- void loginScreen ()

  This method shows the login screen.
- void mainWindowAdmin ()

  This method shows the main menu for the admins.
- void mainWindowEmployee ()

  This method shows the main menu for the employees.
- void mainWindowGuest ()

  This method shows the main menu for the guests.
- void createUserScreen ()

  This method shows the message to create a new user.
- void deleteUserScreen ()

  This method shows the message to delete a user.
- void updateUserScreen ()

  This method shows the message to update a user.
- void displayUsersScreen ()

  This method shows the message to show the users.
- void displaySensorsScreen ()

  This method shows the message to show the sensors.
- void displayAlarmsScreen ()

  This method shows the message to show the alarms.
- void turnOnOffSystemScreen ()

  This method shows the message to turn on or off the system.
- void displayErrorScreen ()

  This method shows the message if and error occurs.

- void createAlarmScreen ()

    This method shows the message to create a new alarm.
- void deleteAlarmScreen ()

    This method shows the message to delete an alarm.
- void saveAlarmScreen ()

    This method shows the message to save an alarm.

Private Member Functions

- bool inputCorrect (int input, int max)

    This method checks if the input is correct.
- int askInputInt (int max)

    This method asks the user for an int input.
- std::string askName ()

    This method asks the user for an input for the name.
- std::string askNIF ()

    This method asks the user for an input for the NIF.
- std::string askPassword ()

    This method asks the user for an input for the password.
- std::string askPrivileges ()

    This method asks the user for an input for the privileges.
- std::string askEmail ()

    This method asks the user for an input for the email.
- void selectUser ()

    This is the method to select a user.
- void shortSelectUser ()

    This is the method to select a short user.
- void askMainUser ()

    This is the method to ask the main user, the one that its going to loggin.
- std::string askTypeAlarm ()

    This is the method to ask the type of the alarm.
- int askIdAlarm ()

    This is the method to ask the id of the alarm.

Private Attributes

- ScreenHardware ∗ screen

    This is the pointer to the ScreenHardware object.
- KeyboardHardware ∗ keyboard

    This is the pointer to the KeyboardHardware object.
- SwitchHardware ∗ sw

    This is the pointer to the SwitchHardware object.
- int selection_

    This is the selection of the user.
- std::string name_

    This is the name of the user.
- std::string nif_

    This is the NIF of the user.
- std::string password_

This is the password of the user.
- std::string nameSelectedUser_

  This is the name of the selected user.
- std::string nifSelectedUser_

  This is the password of the selected user.
- std::string passwordSelectedUser_

  This is the privileges of the selected user.
- std::string privilegesSelectedUser_

  This is the privileges of the selected user.
- std::string emailSelectedUser_

  This is the email of the selected user.
- int idAlarm_

  This is attribute is the id of the alarm.
- std::string typeAlarm_

  This is attribute is the type of the alarm.
- int idSelectedAlarm_

  This is attribute is the id of the selected alarm.

4.16.1 Detailed Description

Definition at line 24 of file MonitoringSystem.h.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 MonitoringSystem()

```
MonitoringSystem::MonitoringSystem (
            ScreenHardware * screen,
            KeyboardHardware * keyboard,
            SwitchHardware * sw )  [explicit]
```

Construct a new Monitoring System object.

Parameters

screen	
keyboard	
sw	

Definition at line 15 of file MonitoringSystem.cpp.

```
18     : screen(screen), keyboard(keyboard), sw(sw) {}
```

4.16.2.2 ∼MonitoringSystem()

```
MonitoringSystem::∼MonitoringSystem ( )
```

Destroy the Monitoring System object.

Definition at line 20 of file MonitoringSystem.cpp.

```
                                     {
  delete screen;
  delete keyboard;
  delete sw;
}
```

References keyboard, screen, and sw.

4.16.3 Member Function Documentation

4.16.3.1 askEmail()

`std::string MonitoringSystem::askEmail ( )  [private]`

This method asks the user for an input for the email.

Returns

std::string

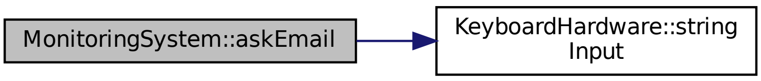
Definition at line 134 of file MonitoringSystem.cpp.

```
                                      {
  std::cout « "(EMAIL) ";
  return keyboard->stringInput();
}
```

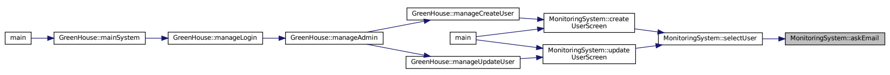
References keyboard, and KeyboardHardware::stringInput().

Referenced by selectUser().

Here is the call graph for this function:



Here is the caller graph for this function:


main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageCreateUser
GreenHouse::manageUpdateUser
main
MonitoringSystem::create UserScreen
MonitoringSystem::update UserScreen
MonitoringSystem::selectUser
MonitoringSystem::askEmail

4.16.3.2 askIdAlarm()

```
int MonitoringSystem::askIdAlarm ( )  [private]
```

This is the method to ask the id of the alarm.

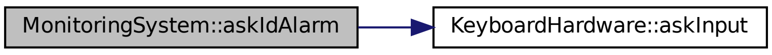
Definition at line 243 of file MonitoringSystem.cpp.

```
243                                 {
244   std::cout « "(ID ALARM) ";
245   return keyboard->askInput();
246 }
```

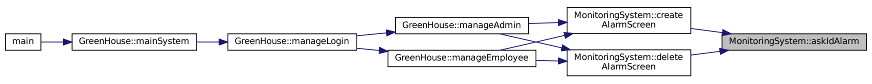
References KeyboardHardware::askInput(), and keyboard.

Referenced by createAlarmScreen(), and deleteAlarmScreen().

Here is the call graph for this function:



Here is the caller graph for this function:


main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageEmployee
MonitoringSystem::create AlarmScreen
MonitoringSystem::delete AlarmScreen
MonitoringSystem::askIdAlarm

4.16.3.3 askInputInt()

```
int MonitoringSystem::askInputInt (
             int max )  [private]
```

This method asks the user for an int input.

Parameters

max	

Returns

int

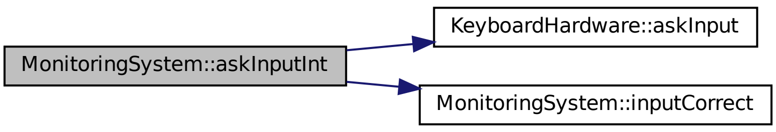
Definition at line 89 of file MonitoringSystem.cpp.

```
                                        {
  int input;

  do {
    input = keyboard->askInput();
  } while (!inputCorrect(input, max));

  return input;
}
```

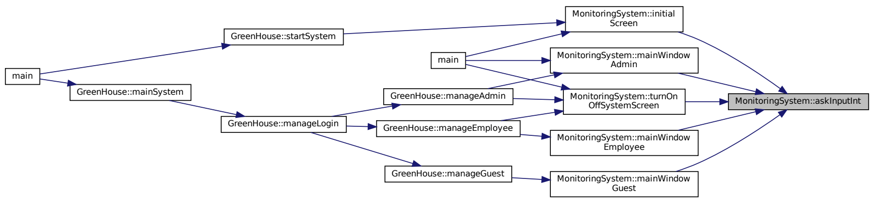
References KeyboardHardware::askInput(), inputCorrect(), and keyboard.

Referenced by initialScreen(), mainWindowAdmin(), mainWindowEmployee(), mainWindowGuest(), and turnOn↩
OffSystemScreen().

Here is the call graph for this function:


MonitoringSystem::askInputInt
KeyboardHardware::askInput
MonitoringSystem::inputCorrect

Here is the caller graph for this function:


main
GreenHouse::mainSystem
GreenHouse::startSystem
GreenHouse::manageLogin
main
GreenHouse::manageAdmin
GreenHouse::manageEmployee
GreenHouse::manageGuest
MonitoringSystem::initial Screen
MonitoringSystem::mainWindow Admin
MonitoringSystem::turnOn OffSystemScreen
MonitoringSystem::mainWindow Employee
MonitoringSystem::mainWindow Guest
MonitoringSystem::askInputInt

4.16.3.4 askMainUser()

`void MonitoringSystem::askMainUser ( )  [private]`

This is the method to ask the main user, the one that its going to loggin.

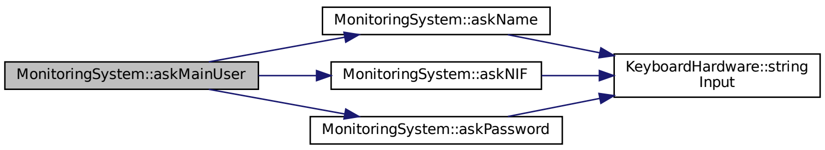
Definition at line 42 of file MonitoringSystem.cpp.

```
                                 {
  name_ = askName();
  password_ = askPassword();
  nif_ = askNIF();
}
```

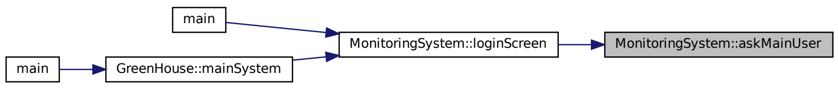
References askName(), askNIF(), askPassword(), name_, nif_, and password_.

Referenced by loginScreen().

Here is the call graph for this function:


MonitoringSystem::askMainUser
MonitoringSystem::askName
MonitoringSystem::askNIF
MonitoringSystem::askPassword
KeyboardHardware::string Input

Here is the caller graph for this function:



4.16.3.5 askName()

`std::string MonitoringSystem::askName ( )  [private]`

This method asks the user for an input for the name.

Returns

std::string

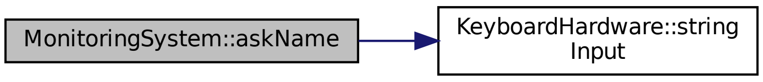
Definition at line 114 of file MonitoringSystem.cpp.

```
                                                {
    std::cout « "(NAME) ";
    return keyboard->stringInput();
}
```

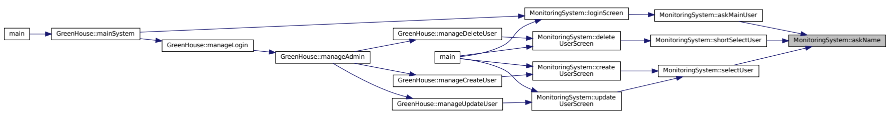
References keyboard, and KeyboardHardware::stringInput().

Referenced by askMainUser(), selectUser(), and shortSelectUser().

Here is the call graph for this function:



Here is the caller graph for this function:


main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageDeleteUser
main
GreenHouse::manageCreateUser
GreenHouse::manageUpdateUser
MonitoringSystem::loginScreen
MonitoringSystem::delete UserScreen
MonitoringSystem::create UserScreen
MonitoringSystem::update UserScreen
MonitoringSystem::askMainUser
MonitoringSystem::shortSelectUser
MonitoringSystem::selectUser
MonitoringSystem::askName

4.16.3.6 askNIF()

```
std::string MonitoringSystem::askNIF ( )  [private]
```

This method asks the user for an input for the NIF.

Returns

std::string

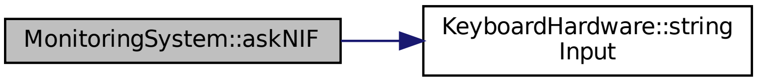
Definition at line 124 of file MonitoringSystem.cpp.

```
124                                  {
125   std::cout « "(NIF) ";
126   return keyboard->stringInput();
127 }
```

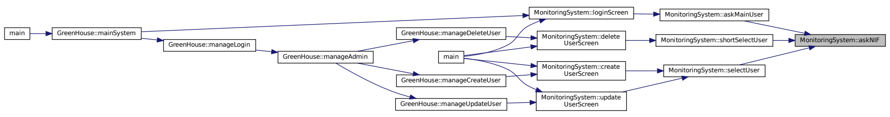
References keyboard, and KeyboardHardware::stringInput().

Referenced by askMainUser(), selectUser(), and shortSelectUser().

Here is the call graph for this function:



Here is the caller graph for this function:


main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageDeleteUser
main
GreenHouse::manageCreateUser
GreenHouse::manageUpdateUser
MonitoringSystem::loginScreen
MonitoringSystem::delete UserScreen
MonitoringSystem::create UserScreen
MonitoringSystem::update UserScreen
MonitoringSystem::askMainUser
MonitoringSystem::shortSelectUser
MonitoringSystem::selectUser
MonitoringSystem::askNIF

4.16.3.7 askPassword()

```
std::string MonitoringSystem::askPassword ( ) [private]
```

This method asks the user for an input for the password.

Returns

std::string

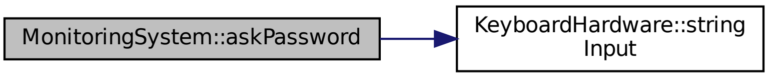
Definition at line 119 of file MonitoringSystem.cpp.

```
                                         {
    std::cout « "(PASSWORD) ";
    return keyboard->stringInput();
}
```

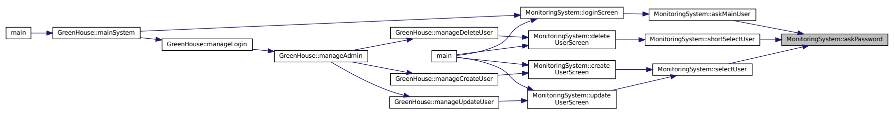
References keyboard, and KeyboardHardware::stringInput().

Referenced by askMainUser(), selectUser(), and shortSelectUser().

Here is the call graph for this function:



Here is the caller graph for this function:


main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageDeleteUser
GreenHouse::manageCreateUser
GreenHouse::manageUpdateUser
main
MonitoringSystem::loginScreen
MonitoringSystem::delete UserScreen
MonitoringSystem::create UserScreen
MonitoringSystem::update UserScreen
MonitoringSystem::askMainUser
MonitoringSystem::shortSelectUser
MonitoringSystem::selectUser
MonitoringSystem::askPassword

4.16.3.8 askPrivileges()

```
std::string MonitoringSystem::askPrivileges ( ) [private]
```

This method asks the user for an input for the privileges.

Returns

std::string

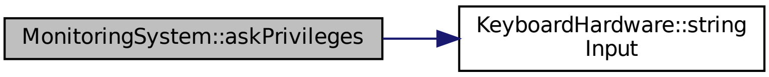
Definition at line 129 of file MonitoringSystem.cpp.

```
                                                {
  std::cout « "(PRIVILEGES) ";
  return keyboard->stringInput();
}
```

References keyboard, and KeyboardHardware::stringInput().

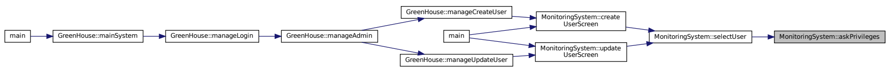
Referenced by selectUser().

Here is the call graph for this function:



MonitoringSystem::askPrivileges
KeyboardHardware::string Input

Here is the caller graph for this function:



main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageCreateUser
GreenHouse::manageUpdateUser
main
MonitoringSystem::create UserScreen
MonitoringSystem::update UserScreen
MonitoringSystem::selectUser
MonitoringSystem::askPrivileges

4.16.3.9 askTypeAlarm()

`std::string MonitoringSystem::askTypeAlarm ( )  [private]`

This is the method to ask the type of the alarm.

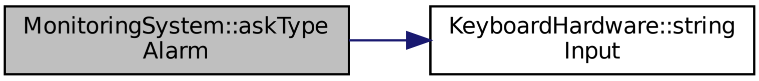
Definition at line 238 of file MonitoringSystem.cpp.

```
                                               {
  std::cout « "(TYPE ALARM) ";
  return keyboard->stringInput();
}
```

References keyboard, and KeyboardHardware::stringInput().

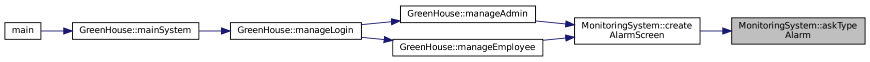
Referenced by createAlarmScreen().

Here is the call graph for this function:



MonitoringSystem::askType Alarm
KeyboardHardware::string Input

Here is the caller graph for this function:


main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageEmployee
MonitoringSystem::create AlarmScreen
MonitoringSystem::askType Alarm

4.16.3.10 createAlarmScreen()

```
void MonitoringSystem::createAlarmScreen ( )
```

This method shows the message to create a new alarm.

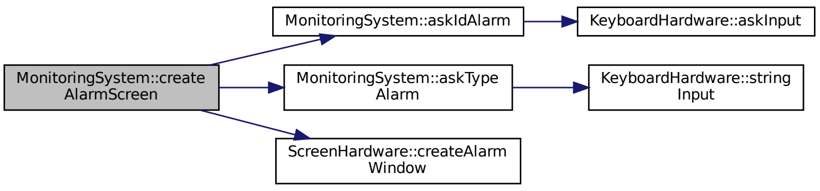
Definition at line 248 of file MonitoringSystem.cpp.

```
248                                          {
249   // Muestro de screen la createAlarmWindow
250   system("clear");
251   screen->createAlarmWindow();
252   typeAlarm_ = askTypeAlarm();
253   idAlarm_ = askIdAlarm();
254 }
```

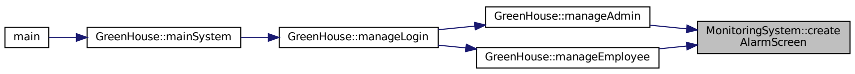
References askIdAlarm(), askTypeAlarm(), ScreenHardware::createAlarmWindow(), idAlarm_, screen, and type↩ Alarm_.

Referenced by GreenHouse::manageAdmin(), and GreenHouse::manageEmployee().

Here is the call graph for this function:


MonitoringSystem::create AlarmScreen
MonitoringSystem::askIdAlarm
KeyboardHardware::askInput
MonitoringSystem::askType Alarm
KeyboardHardware::string Input
ScreenHardware::createAlarm Window

Here is the caller graph for this function:



4.16.3.11 createUserScreen()

```
void MonitoringSystem::createUserScreen ( )
```

This method shows the message to create a new user.

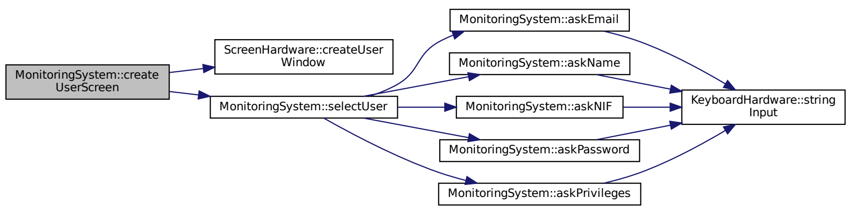
Definition at line 177 of file MonitoringSystem.cpp.

```
                                        {
    // Muestro de screen la createUserWindow, luego pido con el keyboard un input
    // hasta que este entre los valores correctos
    system("clear");
    screen->createUserWindow();
    selectUser();
}
```

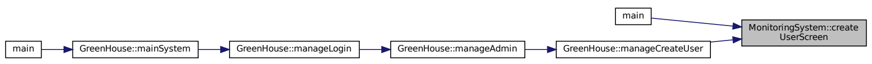
References ScreenHardware::createUserWindow(), screen, and selectUser().

Referenced by main(), and GreenHouse::manageCreateUser().

Here is the call graph for this function:


MonitoringSystem::create UserScreen
ScreenHardware::createUser Window
MonitoringSystem::selectUser
MonitoringSystem::askEmail
MonitoringSystem::askName
MonitoringSystem::askNIF
MonitoringSystem::askPassword
MonitoringSystem::askPrivileges
KeyboardHardware::string Input

Here is the caller graph for this function:



4.16.3.12 deleteAlarmScreen()

```
void MonitoringSystem::deleteAlarmScreen ( )
```

This method shows the message to delete an alarm.

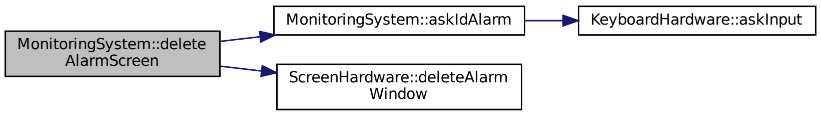
Definition at line 256 of file MonitoringSystem.cpp.

```
                                         {
    // Muestro de screen la deleteAlarmWindow
    system("clear");
    screen->deleteAlarmWindow();
    idSelectedAlarm_ = askIdAlarm();
}
```

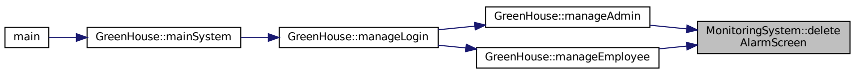
References askIdAlarm(), ScreenHardware::deleteAlarmWindow(), idSelectedAlarm_, and screen.

Referenced by GreenHouse::manageAdmin(), and GreenHouse::manageEmployee().

Here is the call graph for this function:


MonitoringSystem::delete AlarmScreen
MonitoringSystem::askIdAlarm
KeyboardHardware::askInput
ScreenHardware::deleteAlarm Window

Here is the caller graph for this function:



4.16.3.13 deleteUserScreen()

```
void MonitoringSystem::deleteUserScreen ( )
```

This method shows the message to delete a user.

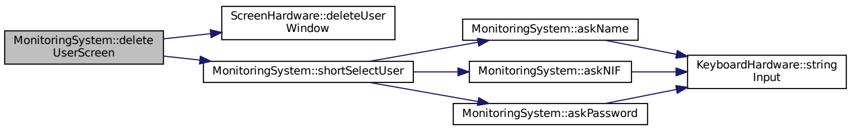
Definition at line 185 of file MonitoringSystem.cpp.

```
                                                {
    // Muestro de screen la deleteUserWindow, luego pido con el keyboard un input
    // hasta que este entre los valores correctos
    system("clear");
    screen->deleteUserWindow();
    shortSelectUser();
}
```

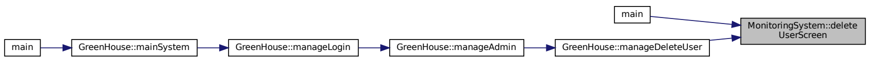
References ScreenHardware::deleteUserWindow(), screen, and shortSelectUser().

Referenced by main(), and GreenHouse::manageDeleteUser().

Here is the call graph for this function:


MonitoringSystem::delete UserScreen
ScreenHardware::deleteUser Window
MonitoringSystem::shortSelectUser
MonitoringSystem::askName
MonitoringSystem::askNIF
MonitoringSystem::askPassword
KeyboardHardware::string Input

Here is the caller graph for this function:



4.16.3.14 displayAlarmsScreen()

```
void MonitoringSystem::displayAlarmsScreen ( )
```

This method shows the message to show the alarms.

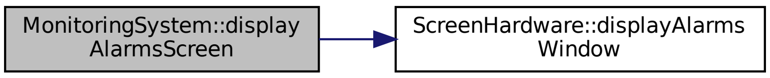
Definition at line 215 of file MonitoringSystem.cpp.

```
                                                {
  // Muestro de screen la displayAlarmsWindow
  system("clear");
  screen->displayAlarmsWindow();
  // keyboard->stringInput();
}
```

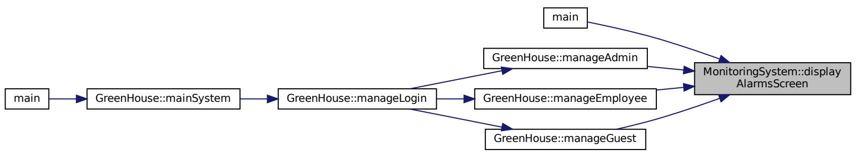
References ScreenHardware::displayAlarmsWindow(), and screen.

Referenced by main(), GreenHouse::manageAdmin(), GreenHouse::manageEmployee(), and GreenHouse↩
::manageGuest().

Here is the call graph for this function:


MonitoringSystem::display AlarmsScreen
ScreenHardware::displayAlarms Window

Here is the caller graph for this function:



4.16.3.15 displayErrorScreen()

```
void MonitoringSystem::displayErrorScreen ( )
```

This method shows the message if and error occurs.

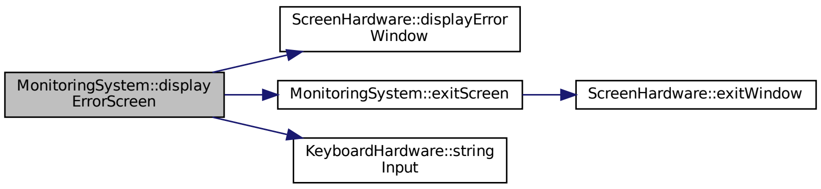
Definition at line 230 of file MonitoringSystem.cpp.

```
                                               {
  // Muestro de screen la displayErrorWindow
  system("clear");
  screen->displayErrorWindow();
  keyboard->stringInput();
  exitScreen();
}
```

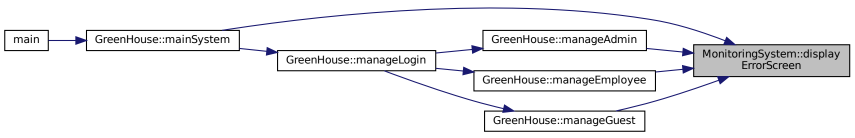
References ScreenHardware::displayErrorWindow(), exitScreen(), keyboard, screen, and KeyboardHardware↩ ::stringInput().

Referenced by GreenHouse::mainSystem(), GreenHouse::manageAdmin(), GreenHouse::manageEmployee(), and GreenHouse::manageGuest().

Here is the call graph for this function:


ScreenHardware::displayError Window
MonitoringSystem::display ErrorScreen
MonitoringSystem::exitScreen
ScreenHardware::exitWindow
KeyboardHardware::string Input

Here is the caller graph for this function:



4.16.3.16 displaySensorsScreen()

```
void MonitoringSystem::displaySensorsScreen ( )
```

This method shows the message to show the sensors.

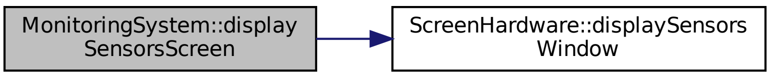
Definition at line 208 of file MonitoringSystem.cpp.

```
                                          {
  // Muestro de screen la displaySensorsWindow
  system("clear");
  screen->displaySensorsWindow();
  // keyboard->stringInput();
}
```

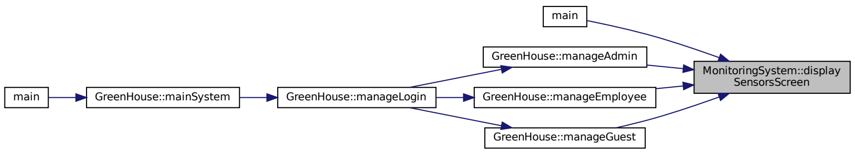
References ScreenHardware::displaySensorsWindow(), and screen.

Referenced by main(), GreenHouse::manageAdmin(), GreenHouse::manageEmployee(), and GreenHouse↩ ::manageGuest().

Here is the call graph for this function:


MonitoringSystem::display SensorsScreen
ScreenHardware::displaySensors Window

Here is the caller graph for this function:



4.16.3.17 displayUsersScreen()

```
void MonitoringSystem::displayUsersScreen ( )
```

This method shows the message to show the users.

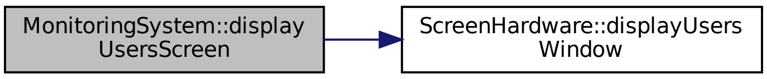
Definition at line 201 of file MonitoringSystem.cpp.

```
                                                    {
        // Muestro de screen la displayUsersWindow
        system("clear");
        screen->displayUsersWindow();
        // keyboard->stringInput();
}
```

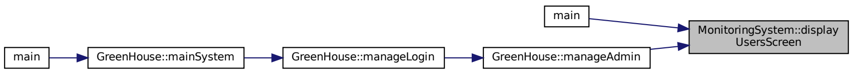
References ScreenHardware::displayUsersWindow(), and screen.

Referenced by main(), and GreenHouse::manageAdmin().

Here is the call graph for this function:


MonitoringSystem::display UsersScreen
ScreenHardware::displayUsers Window

Here is the caller graph for this function:



4.16.3.18 exitScreen()

```
void MonitoringSystem::exitScreen ( )
```

This method exits the screen.

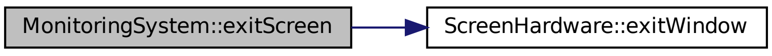
Definition at line 108 of file MonitoringSystem.cpp.

```
108                                  {
109   // Borrar terminal y mostrar el exitWindow
110   system("clear");
111   screen->exitWindow();
112 }
```

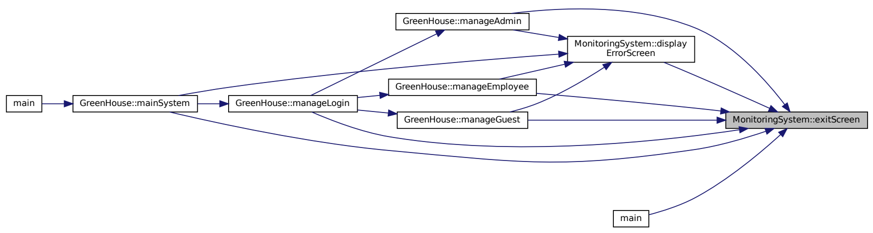
References ScreenHardware::exitWindow(), and screen.

Referenced by displayErrorScreen(), main(), GreenHouse::mainSystem(), GreenHouse::manageAdmin(), Green←
House::manageEmployee(), GreenHouse::manageGuest(), and GreenHouse::manageLogin().

Here is the call graph for this function:



Here is the caller graph for this function:



4.16.3.19 getEmailSelectedUser()

```
std::string MonitoringSystem::getEmailSelectedUser ( )
```

Get the Email Selected User object.

Returns

std::string

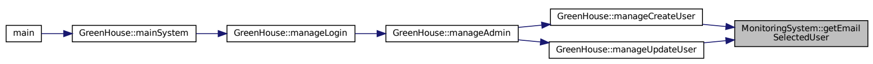
Definition at line 74 of file MonitoringSystem.cpp.

```
{
  return emailSelectedUser_;
}
```

References emailSelectedUser_.

Referenced by GreenHouse::manageCreateUser(), and GreenHouse::manageUpdateUser().

Here is the caller graph for this function:


main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageCreateUser
GreenHouse::manageUpdateUser
MonitoringSystem::getEmail SelectedUser

4.16.3.20 getIdAlarm()

```
int MonitoringSystem::getIdAlarm ( )
```

Get the Id Alarm object.

Returns

int

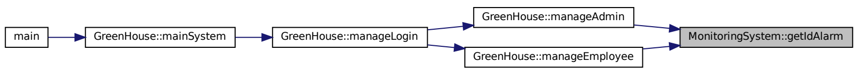
Definition at line 50 of file MonitoringSystem.cpp.

```
{ return idAlarm_; }
```

References idAlarm_.

Referenced by GreenHouse::manageAdmin(), and GreenHouse::manageEmployee().

Here is the caller graph for this function:



4.16.3.21 getIdSelectedAlarm()

```
int MonitoringSystem::getIdSelectedAlarm ( )
```

Get the Id Selected Alarm object.

Returns

int

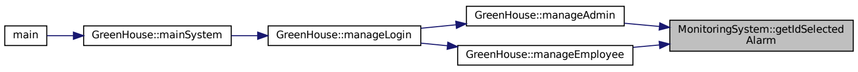
Definition at line 52 of file MonitoringSystem.cpp.

```
52 { return idSelectedAlarm_; }
```

References idSelectedAlarm_.

Referenced by GreenHouse::manageAdmin(), and GreenHouse::manageEmployee().

Here is the caller graph for this function:



4.16.3.22 getName()

```
std::string MonitoringSystem::getName ( )
```

Get the Name object.

Returns

std::string

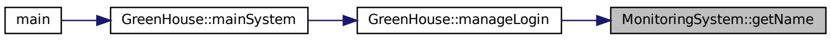
Definition at line 54 of file MonitoringSystem.cpp.

```
54 { return name_; }
```

References name_.

Referenced by GreenHouse::manageLogin().

Here is the caller graph for this function:



4.16.3.23 getNameSelectedUser()

```
std::string MonitoringSystem::getNameSelectedUser ( )
```

Get the Name Selected User object.

Returns

std::string

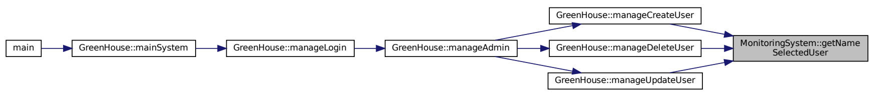
Definition at line 60 of file MonitoringSystem.cpp.

```
60                                              {
61   return nameSelectedUser_;
62 }
```

References nameSelectedUser_.

Referenced by GreenHouse::manageCreateUser(), GreenHouse::manageDeleteUser(), and GreenHouse↩
::manageUpdateUser().

Here is the caller graph for this function:


main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageCreateUser
GreenHouse::manageDeleteUser
GreenHouse::manageUpdateUser
MonitoringSystem::getName
SelectedUser

4.16.3.24 getNIF()

```
std::string MonitoringSystem::getNIF ( )
```

Get the NIF object.

Returns

std::string

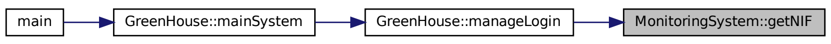
Definition at line 56 of file MonitoringSystem.cpp.

```
56 { return nif_; }
```

References nif_.

Referenced by GreenHouse::manageLogin().

Here is the caller graph for this function:



4.16.3.25 getNIFSelectedUser()

```
std::string MonitoringSystem::getNIFSelectedUser ( )
```

Get the NIF Selected User object.

Returns

std::string

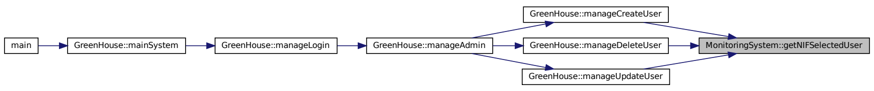
Definition at line 64 of file MonitoringSystem.cpp.

```
64 { return nifSelectedUser_; }
```

References nifSelectedUser_.

Referenced by GreenHouse::manageCreateUser(), GreenHouse::manageDeleteUser(), and GreenHouse↩
::manageUpdateUser().

Here is the caller graph for this function:


main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageCreateUser
GreenHouse::manageDeleteUser
GreenHouse::manageUpdateUser
MonitoringSystem::getNIFSelectedUser

4.16.3.26 getPassword()

```
std::string MonitoringSystem::getPassword ( )
```

Get the Password object.

Returns

std::string

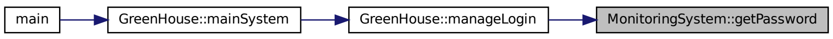
Definition at line 58 of file MonitoringSystem.cpp.

```
58 { return password_; }
```

References password_.

Referenced by GreenHouse::manageLogin().

Here is the caller graph for this function:



4.16.3.27 getPasswordSelectedUser()

```
std::string MonitoringSystem::getPasswordSelectedUser ( )
```

Get the Password Selected User object.

Returns

std::string

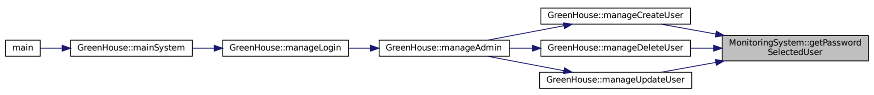
Definition at line 66 of file MonitoringSystem.cpp.

```
66                                                {
67   return passwordSelectedUser_;
68 }
```

References passwordSelectedUser_.

Referenced by GreenHouse::manageCreateUser(), GreenHouse::manageDeleteUser(), and GreenHouse↩ ::manageUpdateUser().

Here is the caller graph for this function:


main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageCreateUser
GreenHouse::manageDeleteUser
GreenHouse::manageUpdateUser
MonitoringSystem::getPassword SelectedUser

4.16.3.28 getPrivilegesSelectedUser()

```
std::string MonitoringSystem::getPrivilegesSelectedUser ( )
```

Get the Privileges Selected User object.

Returns

std::string

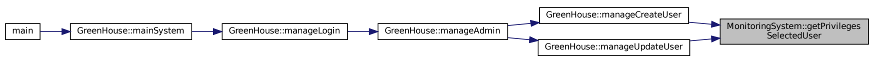
Definition at line 70 of file MonitoringSystem.cpp.

```
70                                                  {
71   return privilegesSelectedUser_;
72 }
```

References privilegesSelectedUser_.

Referenced by GreenHouse::manageCreateUser(), and GreenHouse::manageUpdateUser().

Here is the caller graph for this function:


main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageCreateUser
GreenHouse::manageUpdateUser
MonitoringSystem::getPrivileges SelectedUser

4.16.3.29 getSelection()

```
int MonitoringSystem::getSelection ( )
```

Get the Selection object.

Returns

int

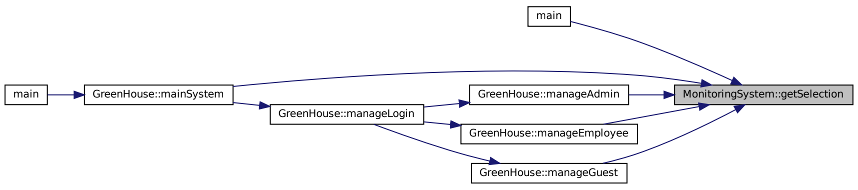
Definition at line 26 of file MonitoringSystem.cpp.

```
26 { return selection_; }
```

References selection_.

Referenced by main(), GreenHouse::mainSystem(), GreenHouse::manageAdmin(), GreenHouse::manage←Employee(), and GreenHouse::manageGuest().

Here is the caller graph for this function:



4.16.3.30 getTypeAlarm()

```
std::string MonitoringSystem::getTypeAlarm ( )
```

Get the Type Alarm object.

Returns

std::string

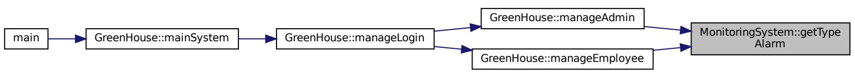
Definition at line 48 of file MonitoringSystem.cpp.

```
48 { return typeAlarm_; }
```

References typeAlarm_.

Referenced by GreenHouse::manageAdmin(), and GreenHouse::manageEmployee().

Here is the caller graph for this function:



4.16.3.31 initialScreen()

```
void MonitoringSystem::initialScreen ( )
```

This method initializes the screen and keyboard.

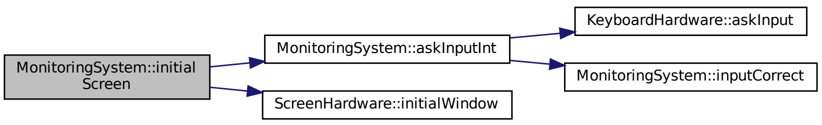
Definition at line 99 of file MonitoringSystem.cpp.

```
                                        {
    system("clear");
    int options = MAIN_MENU_OPTIONS;
    // Muestro de screen la initialWindow, luego pido con el keyboard un input
    // hasta que este entre los valores correctos
    screen->initialWindow();
    selection_ = askInputInt(options);
}
```

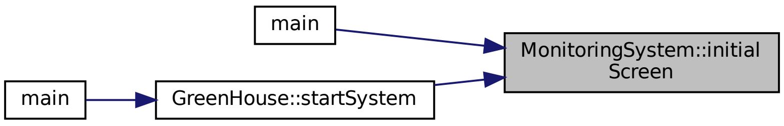
References askInputInt(), ScreenHardware::initialWindow(), MAIN_MENU_OPTIONS, screen, and selection_.

Referenced by main(), and GreenHouse::startSystem().

Here is the call graph for this function:


MonitoringSystem::initial Screen
MonitoringSystem::askInputInt
ScreenHardware::initialWindow
KeyboardHardware::askInput
MonitoringSystem::inputCorrect

Here is the caller graph for this function:



4.16.3.32 inputCorrect()

```
bool MonitoringSystem::inputCorrect (
                int input,
                int max )  [private]
```

This method checks if the input is correct.

Parameters

input	
max	

Returns

true

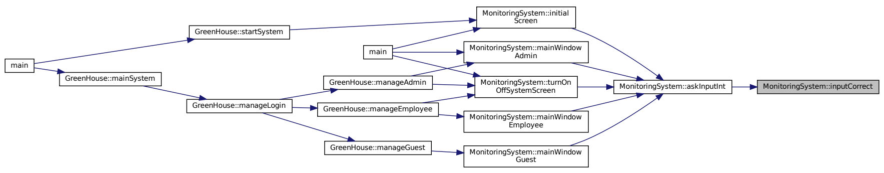
false

Definition at line 78 of file MonitoringSystem.cpp.

```
78                                                          {
79    bool correct = input >= 1 && input <= max;
80    if (!correct) {
81      cout « "Invalid input. Please enter an integer that corresponds to one of "
82              "the options"
83           « endl;
84    }
85    // Input debe de estar entre 1 y max
86    return correct;
87 }
```

Referenced by askInputInt().

Here is the caller graph for this function:


main
GreenHouse::startSystem
GreenHouse::mainSystem
GreenHouse::manageLogin
main
GreenHouse::manageAdmin
GreenHouse::manageEmployee
GreenHouse::manageGuest
MonitoringSystem::initial Screen
MonitoringSystem::mainWindow Admin
MonitoringSystem::turnOn OffSystemScreen
MonitoringSystem::mainWindow Employee
MonitoringSystem::mainWindow Guest
MonitoringSystem::askInputInt
MonitoringSystem::inputCorrect

4.16.3.33 loginScreen()

```
void MonitoringSystem::loginScreen ( )
```

This method shows the login screen.

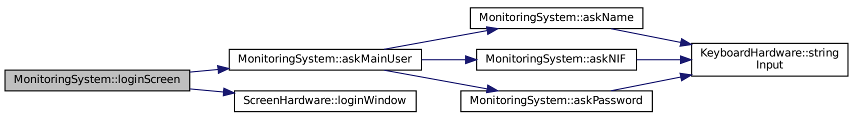
Definition at line 139 of file MonitoringSystem.cpp.

```
139                                  {
140    // Muestro de screen la loginWindow, luego pido con el keyboard un input hasta
141    // que este entre los valores correctos
142    system("clear");
143    screen->loginWindow();
144    askMainUser();
145    // std::cout « name_ « " " « password_ « " " « nif_ « endl;
146 }
```

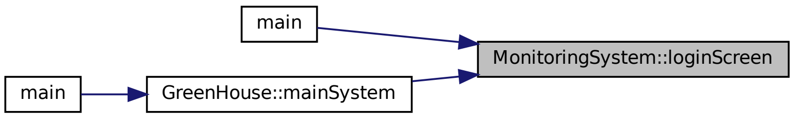
References askMainUser(), ScreenHardware::loginWindow(), and screen.

Referenced by main(), and GreenHouse::mainSystem().

Here is the call graph for this function:


MonitoringSystem::loginScreen
MonitoringSystem::askMainUser
ScreenHardware::loginWindow
MonitoringSystem::askName
MonitoringSystem::askNIF
MonitoringSystem::askPassword
KeyboardHardware::string Input

Here is the caller graph for this function:



4.16.3.34 mainWindowAdmin()

```
void MonitoringSystem::mainWindowAdmin ( )
```

This method shows the main menu for the admins.

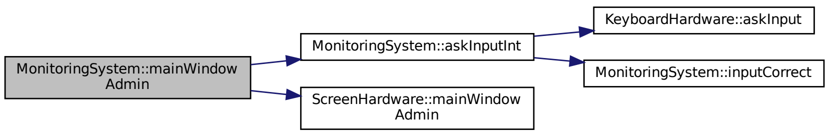
Definition at line 148 of file MonitoringSystem.cpp.

```
148                                          {
149   // Muestro de screen la mainWindowAdmin, luego pido con el keyboard un input
150   // hasta que este entre los valores correctos
151   system("clear");
152   int options = ADMIN_MENU_OPTIONS;
153   screen->mainWindowAdmin();
154   selection_ = askInputInt(options);
155 }
```

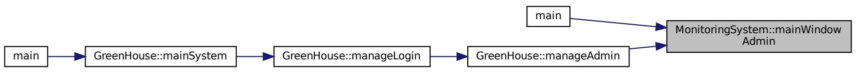
References ADMIN_MENU_OPTIONS, askInputInt(), ScreenHardware::mainWindowAdmin(), screen, and selection_.

Referenced by main(), and GreenHouse::manageAdmin().

Here is the call graph for this function:


MonitoringSystem::mainWindow Admin
MonitoringSystem::askInputInt
KeyboardHardware::askInput
MonitoringSystem::inputCorrect
ScreenHardware::mainWindow Admin

Here is the caller graph for this function:



4.16.3.35 mainWindowEmployee()

```
void MonitoringSystem::mainWindowEmployee ( )
```

This method shows the main menu for the employees.

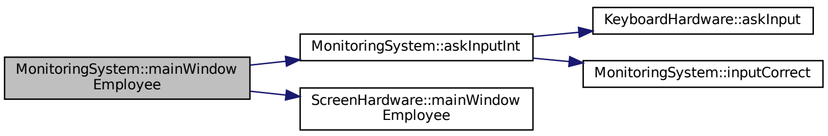
Definition at line 157 of file MonitoringSystem.cpp.

```
                                              {
    // Muestro de screen la mainWindowEmployee, luego pido con el keyboard un
    // input hasta que este entre los valores correctos
    system("clear");
    int options = EMPLOYEE_MENU_OPTIONS;
    screen->mainWindowEmployee();
    selection_ = askInputInt(options);
}
```

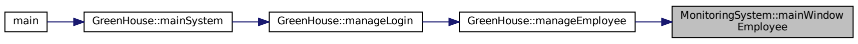
References askInputInt(), EMPLOYEE_MENU_OPTIONS, ScreenHardware::mainWindowEmployee(), screen, and selection_.

Referenced by GreenHouse::manageEmployee().

Here is the call graph for this function:


MonitoringSystem::mainWindow Employee
MonitoringSystem::askInputInt
ScreenHardware::mainWindow Employee
KeyboardHardware::askInput
MonitoringSystem::inputCorrect

Here is the caller graph for this function:


main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageEmployee
MonitoringSystem::mainWindow Employee

4.16.3.36 mainWindowGuest()

```
void MonitoringSystem::mainWindowGuest ( )
```

This method shows the main menu for the guests.

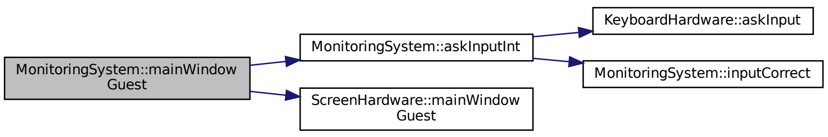
Definition at line 166 of file MonitoringSystem.cpp.

```
                                           {
    // Muestro de screen la mainWindowGuest, luego pido con el keyboard un input
    // hasta que este entre los valores correctos
    system("clear");
    int options = GUEST_MENU_OPTIONS;
    screen->mainWindowGuest();
    selection_ = askInputInt(options);
}
```

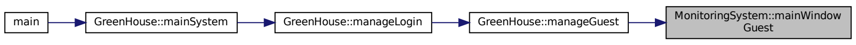
References askInputInt(), GUEST_MENU_OPTIONS, ScreenHardware::mainWindowGuest(), screen, and selection_.

Referenced by GreenHouse::manageGuest().

Here is the call graph for this function:


MonitoringSystem::mainWindow Guest
MonitoringSystem::askInputInt
KeyboardHardware::askInput
MonitoringSystem::inputCorrect
ScreenHardware::mainWindow Guest

Here is the caller graph for this function:



4.16.3.37 saveAlarmScreen()

```
void MonitoringSystem::saveAlarmScreen ( )
```

This method shows the message to save an alarm.

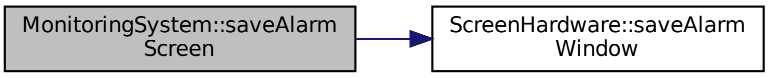
Definition at line 263 of file MonitoringSystem.cpp.

```
                                                        {
   // Muestro de screen la saveAlarmWindow
   system("clear");
   screen->saveAlarmWindow();
}
```

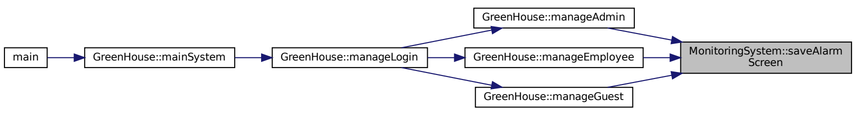
References ScreenHardware::saveAlarmWindow(), and screen.

Referenced by GreenHouse::manageAdmin(), GreenHouse::manageEmployee(), and GreenHouse::manageGuest().

Here is the call graph for this function:


MonitoringSystem::saveAlarm Screen
ScreenHardware::saveAlarm Window

Here is the caller graph for this function:



4.16.3.38 selectUser()

```
void MonitoringSystem::selectUser ( )  [private]
```

This is the method to select a user.

In this selection you have to introduce all the parameters of the user.

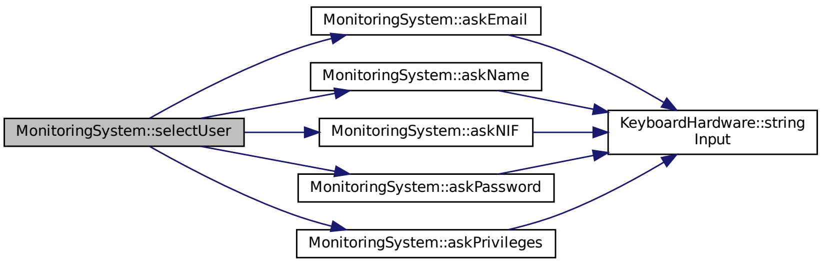
Definition at line 28 of file MonitoringSystem.cpp.

```
                                    {
  nameSelectedUser_ = askName();
  passwordSelectedUser_ = askPassword();
  nifSelectedUser_ = askNIF();
  privilegesSelectedUser_ = askPrivileges();
  emailSelectedUser_ = askEmail();
}
```

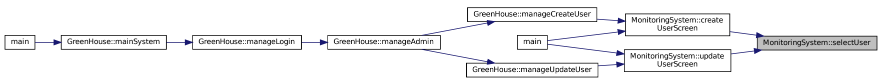
References askEmail(), askName(), askNIF(), askPassword(), askPrivileges(), emailSelectedUser_, name↩ SelectedUser_, nifSelectedUser_, passwordSelectedUser_, and privilegesSelectedUser_.

Referenced by createUserScreen(), and updateUserScreen().

Here is the call graph for this function:


MonitoringSystem::selectUser
MonitoringSystem::askEmail
MonitoringSystem::askName
MonitoringSystem::askNIF
MonitoringSystem::askPassword
MonitoringSystem::askPrivileges
KeyboardHardware::string Input

Here is the caller graph for this function:


main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageCreateUser
main
GreenHouse::manageUpdateUser
MonitoringSystem::create UserScreen
MonitoringSystem::update UserScreen
MonitoringSystem::selectUser

4.16.3.39 shortSelectUser()

```
void MonitoringSystem::shortSelectUser ( )  [private]
```

This is the method to select a short user.

In this selection you have to introduce the name, the password and the NIF of the user.

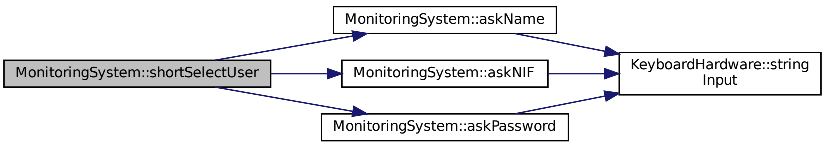
Definition at line 36 of file MonitoringSystem.cpp.

```
36                                        {
37   nameSelectedUser_ = askName();
38   passwordSelectedUser_ = askPassword();
39   nifSelectedUser_ = askNIF();
40 }
```

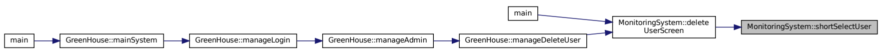
References askName(), askNIF(), askPassword(), nameSelectedUser_, nifSelectedUser_, and password↩
SelectedUser_.

Referenced by deleteUserScreen().

Here is the call graph for this function:


MonitoringSystem::shortSelectUser
MonitoringSystem::askName
MonitoringSystem::askNIF
MonitoringSystem::askPassword
KeyboardHardware::string Input

Here is the caller graph for this function:


main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageDeleteUser
main
MonitoringSystem::delete UserScreen
MonitoringSystem::shortSelectUser

4.16.3.40 turnOnOffSystemScreen()

```
void MonitoringSystem::turnOnOffSystemScreen ( )
```

This method shows the message to turn on or off the system.

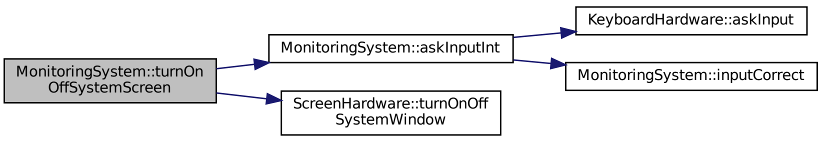
Definition at line 222 of file MonitoringSystem.cpp.

```
222                                               {
223   // Muestro de screen la turnOnOffSystemWindow
224   system("clear");
225   int options = 2;
226   screen->turnOnOffSystemWindow();
227   selection_ = askInputInt(options);
228 }
```

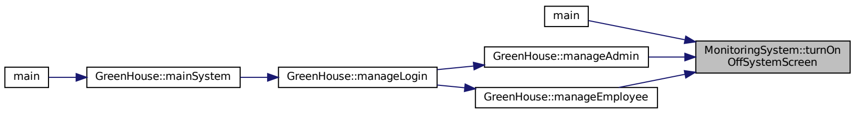
References askInputInt(), screen, selection_, and ScreenHardware::turnOnOffSystemWindow().

Referenced by main(), GreenHouse::manageAdmin(), and GreenHouse::manageEmployee().

Here is the call graph for this function:


MonitoringSystem::turnOn OffSystemScreen
MonitoringSystem::askInputInt
ScreenHardware::turnOnOff SystemWindow
KeyboardHardware::askInput
MonitoringSystem::inputCorrect

Here is the caller graph for this function:



4.16.3.41 updateUserScreen()

```
void MonitoringSystem::updateUserScreen ( )
```

This method shows the message to update a user.

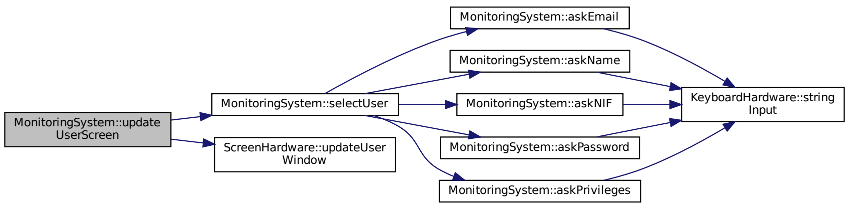
Definition at line 193 of file MonitoringSystem.cpp.

```
                                          {
  // Muestro de screen la updateUserWindow, luego pido con el keyboard un input
  // hasta que este entre los valores correctos
  system("clear");
  screen->updateUserWindow();
  selectUser();
}
```

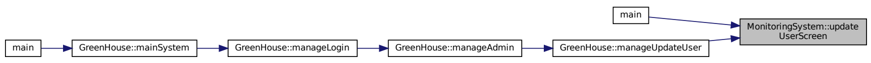
References screen, selectUser(), and ScreenHardware::updateUserWindow().

Referenced by main(), and GreenHouse::manageUpdateUser().

Here is the call graph for this function:


MonitoringSystem::updateUserScreen
MonitoringSystem::selectUser
ScreenHardware::updateUserWindow
MonitoringSystem::askEmail
MonitoringSystem::askName
MonitoringSystem::askNIF
MonitoringSystem::askPassword
MonitoringSystem::askPrivileges
KeyboardHardware::stringInput

Here is the caller graph for this function:



4.16.4 Member Data Documentation

4.16.4.1 emailSelectedUser_

`std::string MonitoringSystem::emailSelectedUser_ [private]`

This is the email of the selected user.

Definition at line 353 of file MonitoringSystem.h.

Referenced by getEmailSelectedUser(), and selectUser().

4.16.4.2 idAlarm_

`int MonitoringSystem::idAlarm_ [private]`

This is attribute is the id of the alarm.

Definition at line 369 of file MonitoringSystem.h.

Referenced by createAlarmScreen(), and getIdAlarm().

4.16.4.3 idSelectedAlarm_

`int MonitoringSystem::idSelectedAlarm_ [private]`

This is attribute is the id of the selected alarm.

Definition at line 377 of file MonitoringSystem.h.

Referenced by deleteAlarmScreen(), and getIdSelectedAlarm().

4.16.4.4 keyboard

`KeyboardHardware* MonitoringSystem::keyboard [private]`

This is the pointer to the KeyboardHardware object.

Definition at line 226 of file MonitoringSystem.h.

Referenced by askEmail(), askIdAlarm(), askInputInt(), askName(), askNIF(), askPassword(), askPrivileges(), ask↩
TypeAlarm(), displayErrorScreen(), and ∼MonitoringSystem().

4.16.4.5 name_

```
std::string MonitoringSystem::name_  [private]
```

This is the name of the user.

Definition at line 291 of file MonitoringSystem.h.

Referenced by askMainUser(), and getName().

4.16.4.6 nameSelectedUser_

```
std::string MonitoringSystem::nameSelectedUser_  [private]
```

This is the name of the selected user.

This is the NIF of the selected user.

Definition at line 333 of file MonitoringSystem.h.

Referenced by getNameSelectedUser(), selectUser(), and shortSelectUser().

4.16.4.7 nif_

```
std::string MonitoringSystem::nif_  [private]
```

This is the NIF of the user.

Definition at line 296 of file MonitoringSystem.h.

Referenced by askMainUser(), and getNIF().

4.16.4.8 nifSelectedUser_

```
std::string MonitoringSystem::nifSelectedUser_  [private]
```

This is the password of the selected user.

Definition at line 338 of file MonitoringSystem.h.

Referenced by getNIFSelectedUser(), selectUser(), and shortSelectUser().

4.16.4.9 password_

```
std::string MonitoringSystem::password_  [private]
```

This is the password of the user.

Definition at line 301 of file MonitoringSystem.h.

Referenced by askMainUser(), and getPassword().

4.16.4.10 passwordSelectedUser_

```
std::string MonitoringSystem::passwordSelectedUser_  [private]
```

This is the privileges of the selected user.

Definition at line 343 of file MonitoringSystem.h.

Referenced by getPasswordSelectedUser(), selectUser(), and shortSelectUser().

4.16.4.11 privilegesSelectedUser_

```
std::string MonitoringSystem::privilegesSelectedUser_  [private]
```

This is the privileges of the selected user.

Definition at line 348 of file MonitoringSystem.h.

Referenced by getPrivilegesSelectedUser(), and selectUser().

4.16.4.12 screen

```
ScreenHardware* MonitoringSystem::screen  [private]
```

This is the pointer to the ScreenHardware object.

Definition at line 221 of file MonitoringSystem.h.

Referenced by createAlarmScreen(), createUserScreen(), deleteAlarmScreen(), deleteUserScreen(), display↩ AlarmsScreen(), displayErrorScreen(), displaySensorsScreen(), displayUsersScreen(), exitScreen(), initialScreen(), loginScreen(), mainWindowAdmin(), mainWindowEmployee(), mainWindowGuest(), saveAlarmScreen(), turnOn↩ OffSystemScreen(), updateUserScreen(), and ∼MonitoringSystem().

4.16.4.13 selection_

```
int MonitoringSystem::selection_  [private]
```

This is the selection of the user.

Definition at line 286 of file MonitoringSystem.h.

Referenced by getSelection(), initialScreen(), mainWindowAdmin(), mainWindowEmployee(), mainWindowGuest(), and turnOnOffSystemScreen().

4.16.4.14 sw

```
SwitchHardware* MonitoringSystem::sw  [private]
```

This is the pointer to the SwitchHardware object.

Definition at line 231 of file MonitoringSystem.h.

Referenced by ∼MonitoringSystem().

4.16.4.15 typeAlarm_

```
std::string MonitoringSystem::typeAlarm_  [private]
```

This is attribute is the type of the alarm.

Definition at line 373 of file MonitoringSystem.h.

Referenced by createAlarmScreen(), and getTypeAlarm().

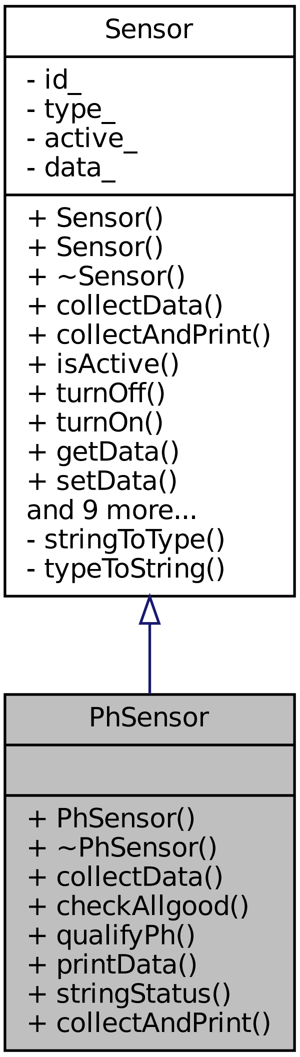
The documentation for this class was generated from the following files:

- src/MonitoringSystem.h
- src/MonitoringSystem.cpp

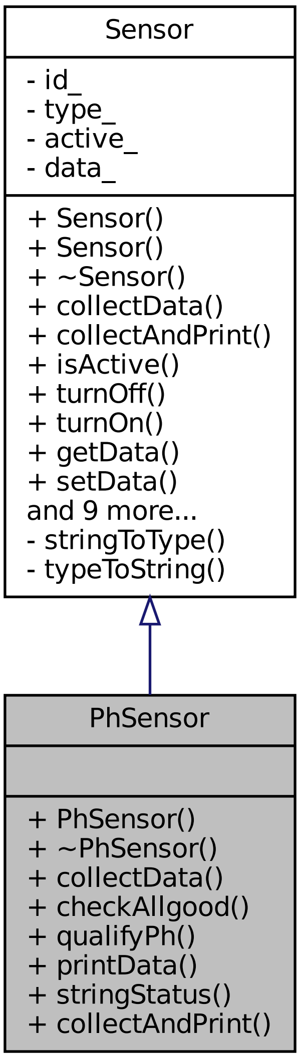
4.17 PhSensor Class Reference

```
#include <PhSensor.h>
```

Inheritance diagram for PhSensor:



Collaboration diagram for PhSensor:



Public Member Functions

- PhSensor (int id, bool active)

  Construct a new Ph Sensor object.
- ~PhSensor () override

  Destroy the Ph Sensor object.
- void collectData () override

  Collect data of the Ph Sensor.
- bool checkAllgood () const override

  Check if the Ph Sensor is working properly.
- std::string qualifyPh () const

  This qualifies the Ph Sensor into Acidic, Neutral or Alkaline.

- void printData () const override
  This method prints the data of the Ph Sensor.
- std::string stringStatus () const
  This method returns the status in a string.
- void collectAndPrint ()
  Collect and print the data of the Ph Sensor.

Friends

- std::ostream & operator<< (std::ostream &os, const PhSensor &sensor)
  Operator << overload.

Additional Inherited Members

4.17.1 Detailed Description

Definition at line 15 of file PhSensor.h.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 PhSensor()

```
PhSensor::PhSensor (
            int id,
            bool active )  [explicit]
```

Construct a new Ph Sensor object.

Parameters

id	
active	

Returns

PhSensor object

Definition at line 9 of file PhSensor.cpp.

```
      : Sensor(id, Sensor::Types::PH_SENSOR, active) {}
```

4.17.2.2 ∼PhSensor()

```
PhSensor::∼PhSensor ( )  [override]
```

Destroy the Ph Sensor object.

Definition at line 11 of file PhSensor.cpp.

```
11 {}
```

4.17.3 Member Function Documentation

4.17.3.1 checkAllgood()

```
bool PhSensor::checkAllgood ( ) const  [override], [virtual]
```

Check if the Ph Sensor is working properly.

Returns

true if the Ph Sensor is working properly

false if the Ph Sensor is not working properly

Reimplemented from Sensor.

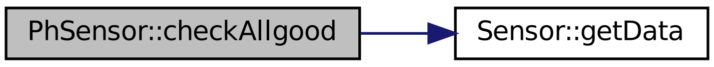
Definition at line 22 of file PhSensor.cpp.

```
22                                    {
23   float data = Sensor::getData();
24
25   if (data >= 6.2 && data <= 7.8) {
26     return true;
27   } else {
28     return false;
29   }
30 }
```

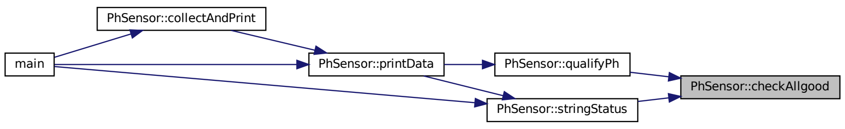
References Sensor::getData().

Referenced by qualifyPh(), and stringStatus().

Here is the call graph for this function:



Here is the caller graph for this function:



4.17.3.2 collectAndPrint()

```
void PhSensor::collectAndPrint ( )  [virtual]
```

Collect and print the data of the Ph Sensor.

Reimplemented from Sensor.

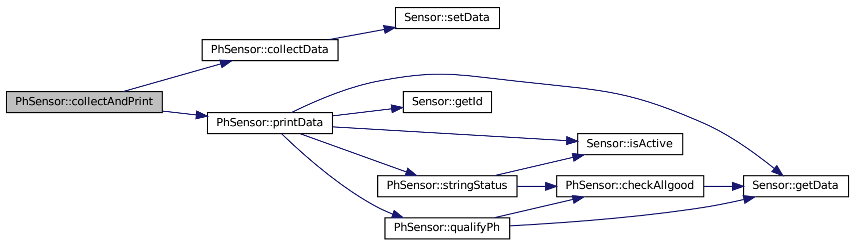
Definition at line 72 of file PhSensor.cpp.

```
                               {
  collectData();
  printData();
}
```

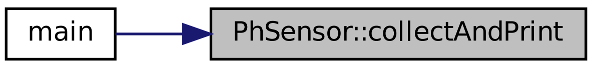
References collectData(), and printData().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.17.3.3 collectData()

```
void PhSensor::collectData ( )  [override], [virtual]
```

Collect data of the Ph Sensor.

This method collects the data of the Ph Sensor and stores it in the data attribute.

Reimplemented from Sensor.

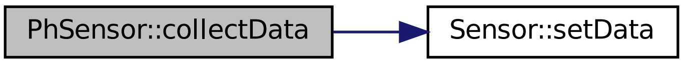
Definition at line 13 of file PhSensor.cpp.

```
                          {
  std::random_device rd;
  std::mt19937 gen(rd());
  std::uniform_real_distribution<> dis(6, 8);
  float reading = dis(gen);

  Sensor::setData(reading);
}
```

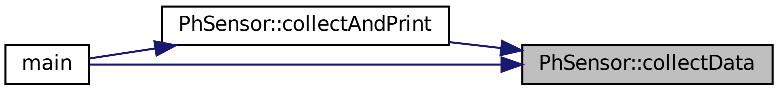
References Sensor::setData().

Referenced by collectAndPrint(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.17.3.4 printData()

```
void PhSensor::printData ( ) const  [override], [virtual]
```

This method prints the data of the Ph Sensor.

Reimplemented from Sensor.

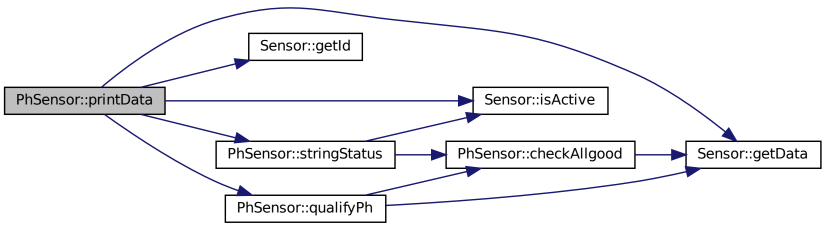
Definition at line 43 of file PhSensor.cpp.

```
                              {
  if (Sensor::isActive()) {
    std::cout « "Ph Sensor with "
              « "ID: " « Sensor::getId() « " - Data: " « Sensor::getData()
              « " - Status: " « stringStatus()
              « " - Qualification: " « qualifyPh() « endl;
  } else {
    std::cout « "Ph Sensor ID: " « Sensor::getId()
              « " - Status: " « stringStatus() « endl;
  }
}
```

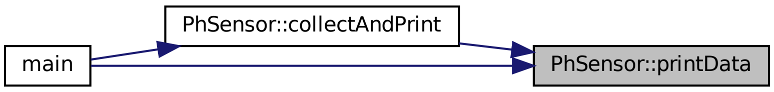
References Sensor::getData(), Sensor::getId(), Sensor::isActive(), qualifyPh(), and stringStatus().

Referenced by collectAndPrint(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.17.3.5 qualifyPh()

```
std::string PhSensor::qualifyPh ( ) const
```

This qualifies the Ph Sensor into Acidic, Neutral or Alkaline.

Returns

std::string of the qualification of the Ph Sensor

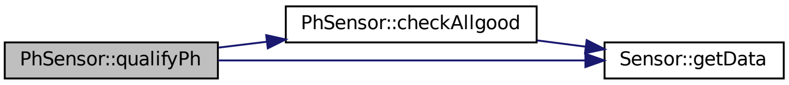
Definition at line 32 of file PhSensor.cpp.

```
                                       {
  float data = Sensor::getData();
  if (this->checkAllgood()) {
    return "Ideal";
  } else if (data < 6.5f) {
    return "Acidic";
  } else {
    return "Alkaline";
  }
}
```

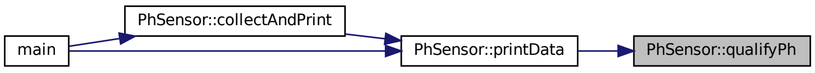
References checkAllgood(), and Sensor::getData().

Referenced by printData().

Here is the call graph for this function:



Here is the caller graph for this function:



4.17.3.6 stringStatus()

```
std::string PhSensor::stringStatus ( ) const
```

This method returns the status in a string.

Returns

std::string of the status of the Ph Sensor

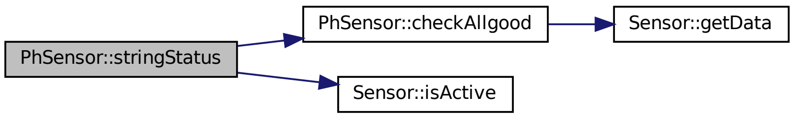
Definition at line 60 of file PhSensor.cpp.

```
                                        {
  if (Sensor::isActive()) {
    if (this->checkAllgood()) {
      return "ACTIVE - GOOD STATUS";
    } else {
      return "ACTIVE - BAD STATUS";
    }
  } else {
    return "INACTIVE";
  }
}
```

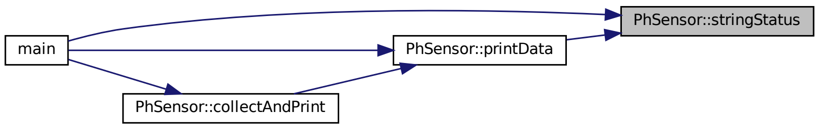
References checkAllgood(), and Sensor::isActive().

Referenced by main(), and printData().

Here is the call graph for this function:



Here is the caller graph for this function:



4.17.4 Friends And Related Function Documentation

4.17.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const PhSensor & sensor )   [friend]
```

Operator << overload.

Definition at line 55 of file PhSensor.cpp.

```
                                                                       {
    sensor.printData();
    return os;
}
```

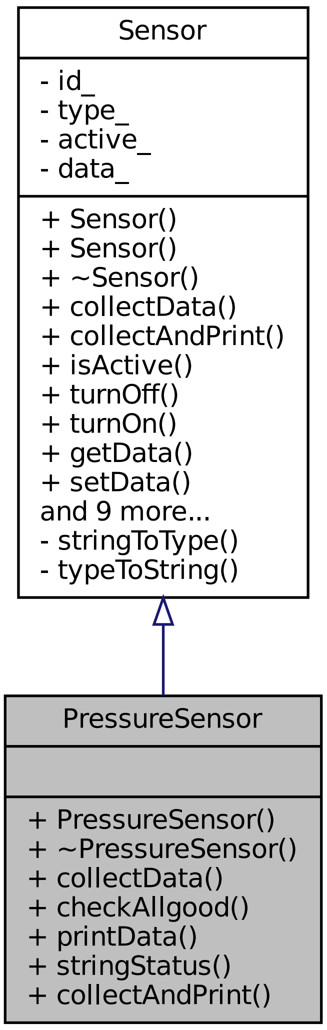
The documentation for this class was generated from the following files:

- src/PhSensor.h
- src/PhSensor.cpp

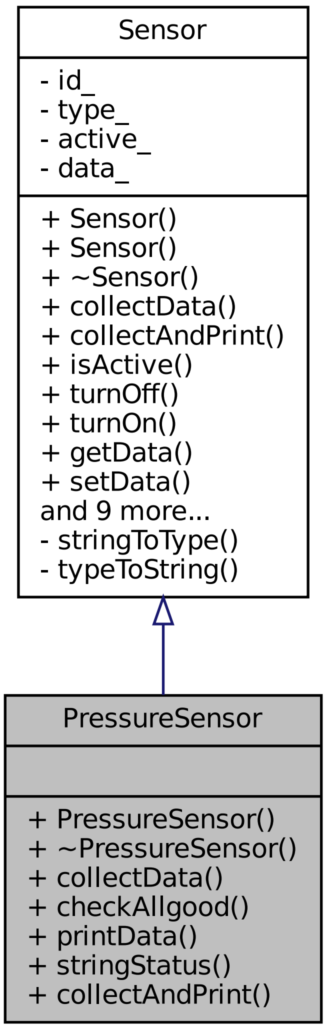
4.18 PressureSensor Class Reference

```
#include <PressureSensor.h>
```

Inheritance diagram for PressureSensor:



Collaboration diagram for PressureSensor:



Public Member Functions

- PressureSensor (int id, bool active)

  Construct a new Pressure Sensor object.
- ∼PressureSensor () override

  Destroy the Pressure Sensor object.
- void collectData () override

  Collect data of the Pressure Sensor.
- bool checkAllgood () const override

  Check if the Pressure Sensor is working properly.
- void printData () const override

  Print the data of the Pressure Sensor.
- std::string stringStatus () const

This method returns the status in a string.
- void collectAndPrint ()

  Collect and print the data of the Pressure Sensor.

Friends

- std::ostream & operator<< (std::ostream &os, const PressureSensor &sensor)

  Operator << overload.

Additional Inherited Members

4.18.1 Detailed Description

Definition at line 15 of file PressureSensor.h.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 PressureSensor()

```
PressureSensor::PressureSensor (
            int id,
            bool active )  [explicit]
```

Construct a new Pressure Sensor object.

Parameters

id	
active	

Returns

PressureSensor object

Definition at line 9 of file PressureSensor.cpp.

```
10     : Sensor(id, Sensor::Types::PRESSURE, active) {}
```

4.18.2.2 ∼PressureSensor()

```
PressureSensor::∼PressureSensor ( )  [override]
```

Destroy the Pressure Sensor object.

Definition at line 12 of file PressureSensor.cpp.

```
12 {}
```

4.18.3 Member Function Documentation

4.18.3.1 checkAllgood()

`bool PressureSensor::checkAllgood ( ) const [override], [virtual]`

Check if the Pressure Sensor is working properly.

Returns

true if the Pressure Sensor is working properly

false if the Pressure Sensor is not working properly

Reimplemented from Sensor.

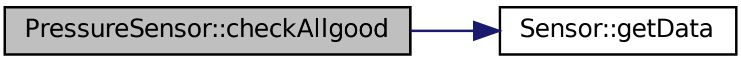
Definition at line 23 of file PressureSensor.cpp.

```
23                                          {
24   float data = Sensor::getData();
25   if (data >= 0.91f && data <= 1.09f) {
26     return true;
27   } else {
28     return false;
29   }
30 }
```

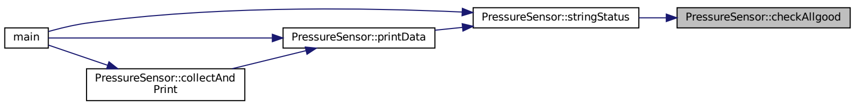
References Sensor::getData().

Referenced by stringStatus().

Here is the call graph for this function:



Here is the caller graph for this function:


main
PressureSensor::stringStatus
PressureSensor::printData
PressureSensor::collectAnd Print
PressureSensor::checkAllgood

4.18.3.2 collectAndPrint()

```
void PressureSensor::collectAndPrint ( )  [virtual]
```

Collect and print the data of the Pressure Sensor.

Reimplemented from Sensor.

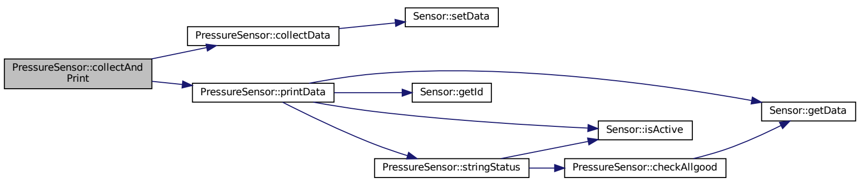
Definition at line 60 of file PressureSensor.cpp.

```
60                                    {
61   collectData();
62   printData();
63 }
```

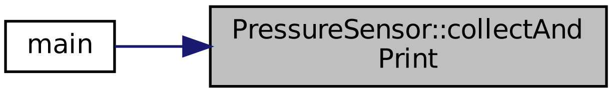
References collectData(), and printData().

Referenced by main().

Here is the call graph for this function:


PressureSensor::collectAnd Print
PressureSensor::collectData
Sensor::setData
PressureSensor::printData
Sensor::getId
Sensor::getData
Sensor::isActive
PressureSensor::stringStatus
PressureSensor::checkAllgood

Here is the caller graph for this function:



4.18.3.3 collectData()

```
void PressureSensor::collectData ( )  [override], [virtual]
```

Collect data of the Pressure Sensor.

This method collects the data of the Pressure Sensor and stores it in the data attribute.

Reimplemented from Sensor.

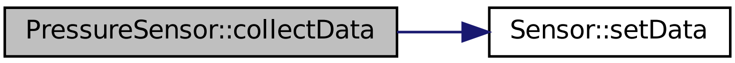
Definition at line 14 of file PressureSensor.cpp.

```
                                    {
    // Numero random entre 0.90 y 1.10 bares
    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_real_distribution<> dis(0.9, 1.1);
    float pressure = dis(gen);
    Sensor::setData(pressure);
}
```

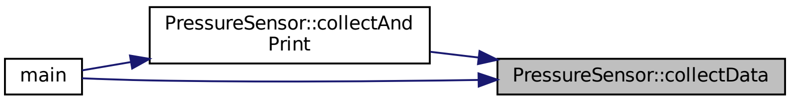
References Sensor::setData().

Referenced by collectAndPrint(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.18.3.4 printData()

`void PressureSensor::printData ( ) const [override], [virtual]`

Print the data of the Pressure Sensor.

Reimplemented from Sensor.

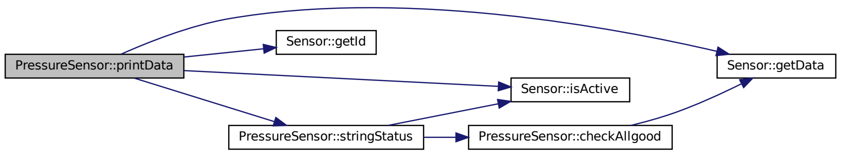
Definition at line 37 of file PressureSensor.cpp.

```
                                        {
    if (Sensor::isActive()) {
      std::cout « "Pressure Sensor with "
                « "ID: " « Sensor::getId() « " - Data: " « Sensor::getData()
                « " bar - Status: " « stringStatus() « endl;
    } else {
      std::cout « "Pressure Sensor ID: " « Sensor::getId()
                « " - Status: " « stringStatus() « endl;
    }
}
```

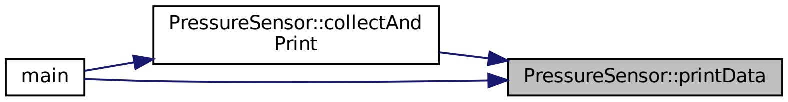
References Sensor::getData(), Sensor::getId(), Sensor::isActive(), and stringStatus().

Referenced by collectAndPrint(), and main().

Here is the call graph for this function:


PressureSensor::printData
Sensor::getId
Sensor::isActive
Sensor::getData
PressureSensor::stringStatus
PressureSensor::checkAllgood

Here is the caller graph for this function:



4.18.3.5 stringStatus()

```
std::string PressureSensor::stringStatus ( ) const
```

This method returns the status in a string.

Returns

std::string of the status of the Pressure Sensor

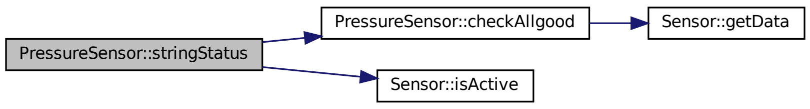
Definition at line 48 of file PressureSensor.cpp.

```
                                                {
  if (Sensor::isActive()) {
    if (this->checkAllgood()) {
      return "ACTIVE - GOOD STATUS";
    } else {
      return "ACTIVE - BAD STATUS";
    }
  } else {
    return "INACTIVE";
  }
}
```

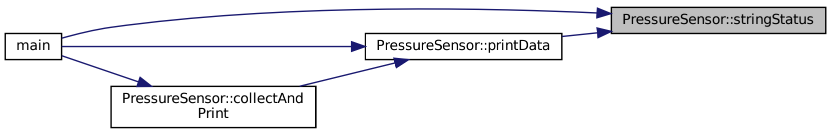
References checkAllgood(), and Sensor::isActive().

Referenced by main(), and printData().

Here is the call graph for this function:



Here is the caller graph for this function:


main
PressureSensor::stringStatus
PressureSensor::printData
PressureSensor::collectAnd Print

4.18.4 Friends And Related Function Documentation

4.18.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const PressureSensor & sensor )  [friend]
```

Operator << overload.

Definition at line 32 of file PressureSensor.cpp.

```
                                                                        {
    sensor.printData();
    return os;
}
```

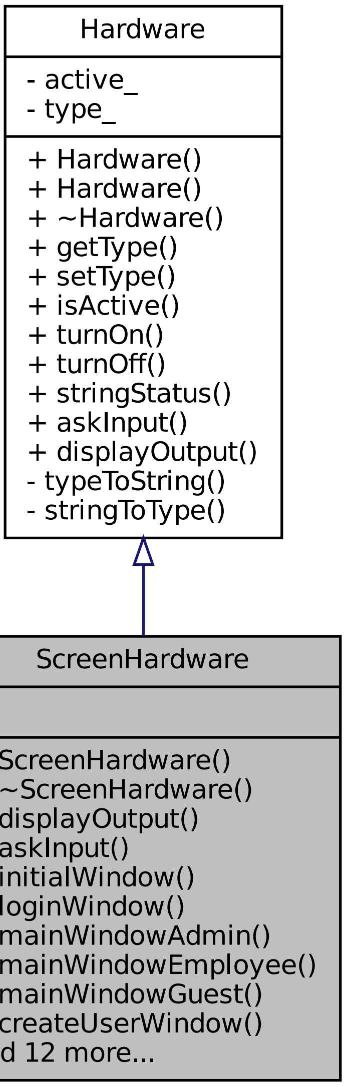
The documentation for this class was generated from the following files:

- src/PressureSensor.h
- src/PressureSensor.cpp

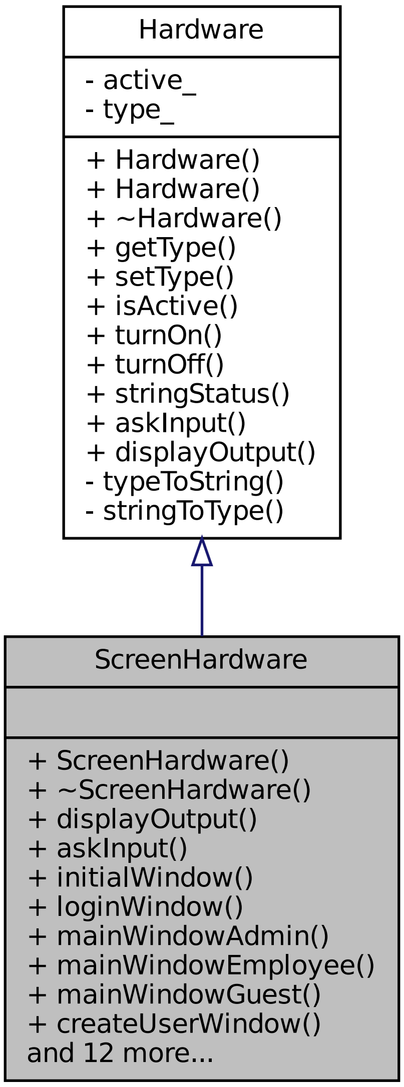
4.19 ScreenHardware Class Reference

```
#include <ScreenHardware.h>
```

Inheritance diagram for ScreenHardware:



Collaboration diagram for ScreenHardware:



Public Member Functions

- ScreenHardware (bool active)

  Construct a new Screen Hardware object.
- ∼ScreenHardware () override

  Destroy the Screen Hardware object.
- void displayOutput () const override

  This method displays the output of the system.
- int askInput () override

  Ask for an input to the user (this method in reality is not used, we use the keyboard to ask for an input)
- void initialWindow ()

This method displays the initial window of the system.
- void loginWindow ()

  This method displays the login window of the system.
- void mainWindowAdmin ()

  This method displays the main window of the system for the admin.
- void mainWindowEmployee ()

  This method displays the main window of the system for the employee.
- void mainWindowGuest ()

  This method displays the main window of the system for the guest.
- void createUserWindow ()

  Create a User object window.
- void deleteUserWindow ()

  Delete a User object window.
- void updateUserWindow ()

  Update a User object window.
- void displayUsersWindow ()

  Display all Users window.
- void createAlarmWindow ()

  Create a Alarm Window object.
- void deleteAlarmWindow ()

  Delete a Alarm Window object.
- void displaySensorsWindow ()

  Display all Sensors window.
- void displayAlarmsWindow ()

  Display all Alarms window.
- void saveAlarmWindow ()

  Save Alarm Window object.
- void turnOnOffSystemWindow ()

  Turn on or off the system window.
- void displayErrorWindow ()

  Display the error window.
- void cleanScreen ()

  Clean the screen.
- void exitWindow ()

  Exit the window.

Additional Inherited Members

4.19.1 Detailed Description

Definition at line 15 of file ScreenHardware.h.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 ScreenHardware()

```
ScreenHardware::ScreenHardware (
              bool active )  [explicit]
```

Construct a new Screen Hardware object.

Parameters

active	

Returns

ScreenHardware object

Definition at line 15 of file ScreenHardware.cpp.

```
    : Hardware(active, Hardware::Types_Hardware::SCREEN) {}
```

4.19.2.2 ∼ScreenHardware()

```
ScreenHardware::∼ScreenHardware ( )  [override]
```

Destroy the Screen Hardware object.

Definition at line 18 of file ScreenHardware.cpp.

```
{}
```

4.19.3 Member Function Documentation

4.19.3.1 askInput()

```
int ScreenHardware::askInput ( )  [override], [virtual]
```

Ask for an input to the user (this method in reality is not used, we use the keyboard to ask for an input)

Returns

int

Reimplemented from Hardware.

Definition at line 24 of file ScreenHardware.cpp.

```
                              {
    std::cout « "Screen wating a input..." « std::endl;
    return 0;
}
```

4.19.3.2 cleanScreen()

`void ScreenHardware::cleanScreen ( )`

Clean the screen.

Definition at line 155 of file ScreenHardware.cpp.

```
155                                       {
156   // Aqui tengo que limpiar la pantalla
157   system("clear");
158 }
```

4.19.3.3 createAlarmWindow()

`void ScreenHardware::createAlarmWindow ( )`

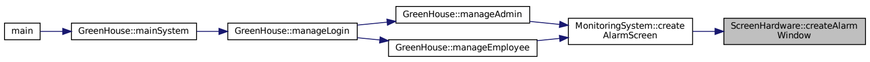
Create a Alarm Window object.

Definition at line 122 of file ScreenHardware.cpp.

```
122                                             {
123   // Aqui tengo que mostrar un menu para crear una alarma
124   std::cout « "---_Create Alarm Window_---" « std::endl;
125   std::cout « "First the type(intro), then the id(intro)" « std::endl;
126 }
```

Referenced by MonitoringSystem::createAlarmScreen().

Here is the caller graph for this function:


main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageEmployee
MonitoringSystem::create AlarmScreen
ScreenHardware::createAlarm Window

4.19.3.4 createUserWindow()

`void ScreenHardware::createUserWindow ( )`

Create a User object window.

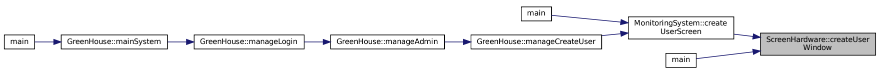
Definition at line 92 of file ScreenHardware.cpp.

```
92                                          {
93   // Aqui tengo que mostrar un menu para crear un usuario
94   std::cout « "---_Create User Window_---" « std::endl;
95   std::cout « ASK_DATA « std::endl;
96   std::cout « USER_PROMPT « std::endl;
97 }
```

References ASK_DATA, and USER_PROMPT.

Referenced by MonitoringSystem::createUserScreen(), and main().

Here is the caller graph for this function:


main
main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageCreateUser
MonitoringSystem::create UserScreen
main
ScreenHardware::createUser Window

4.19.3.5 deleteAlarmWindow()

```
void ScreenHardware::deleteAlarmWindow ( )
```

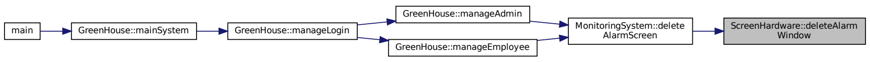
Delete a Alarm Window object.

Definition at line 127 of file ScreenHardware.cpp.

```
127                                        {
128   // Aqui tengo que mostrar un menu para borrar una alarma
129   std::cout « "---_Delete Alarm Window_---" « std::endl;
130   std::cout « "Enter the id of the alarm you want to delete" « std::endl;
131 }
```

Referenced by MonitoringSystem::deleteAlarmScreen().

Here is the caller graph for this function:


main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageEmployee
MonitoringSystem::delete AlarmScreen
ScreenHardware::deleteAlarm Window

4.19.3.6 deleteUserWindow()

```
void ScreenHardware::deleteUserWindow ( )
```

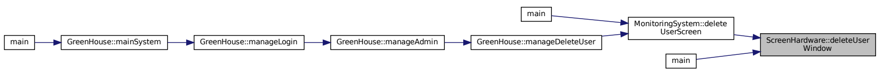
Delete a User object window.

Definition at line 99 of file ScreenHardware.cpp.

```
99                                      {
100   // Aqui tengo que mostrar un menu para borrar un usuario
101   std::cout « "---_Delete User Window_---" « std::endl;
102   std::cout « "First the name(intro), then the password(intro), then the "
103                "nif(intro)"
104             « std::endl;
105 }
```

Referenced by MonitoringSystem::deleteUserScreen(), and main().

Here is the caller graph for this function:


main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageDeleteUser
main
MonitoringSystem::delete UserScreen
main
ScreenHardware::deleteUser Window

4.19.3.7 displayAlarmsWindow()

```
void ScreenHardware::displayAlarmsWindow ( )
```

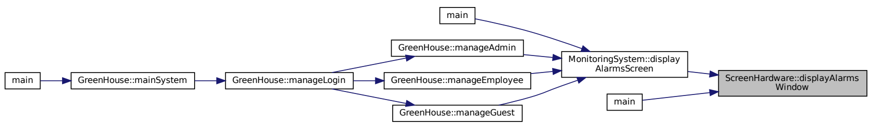
Display all Alarms window.

Definition at line 138 of file ScreenHardware.cpp.

```
138                                          {
139   // Aqui tengo que mostrar un menu para ver todas las alarmas
140   std::cout « "---_Display Alarms Window_---" « std::endl;
141 }
```

Referenced by MonitoringSystem::displayAlarmsScreen(), and main().

Here is the caller graph for this function:



4.19.3.8 displayErrorWindow()

```
void ScreenHardware::displayErrorWindow ( )
```

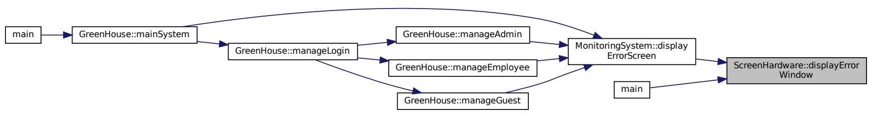
Display the error window.

Definition at line 149 of file ScreenHardware.cpp.

```
149                                         {
150   // Aqui tengo que mostrar un menu de error
151   std::cout « "---_Display Error Window_---" « std::endl;
152   std::cout « "And error happend" « std::endl;
153 }
```

Referenced by MonitoringSystem::displayErrorScreen(), and main().

Here is the caller graph for this function:



4.19.3.9 displayOutput()

```
void ScreenHardware::displayOutput ( ) const  [override], [virtual]
```

This method displays the output of the system.

Reimplemented from Hardware.

Definition at line 20 of file ScreenHardware.cpp.

```
20                                            {
21   std::cout « "Displaying output..." « std::endl;
22 }
```

4.19.3.10 displaySensorsWindow()

```
void ScreenHardware::displaySensorsWindow ( )
```

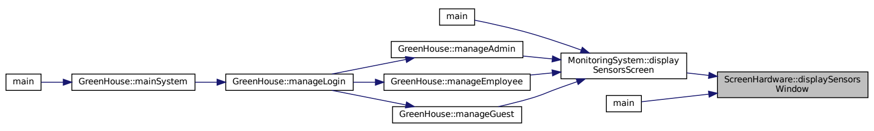
Display all Sensors window.

Definition at line 133 of file ScreenHardware.cpp.

```
133                                           {
134   // Aqui tengo que mostrar un menu para ver todos los sensores
135   std::cout « "---_Display Sensors Window_---" « std::endl;
136 }
```

Referenced by MonitoringSystem::displaySensorsScreen(), and main().

Here is the caller graph for this function:



4.19.3.11 displayUsersWindow()

```
void ScreenHardware::displayUsersWindow ( )
```

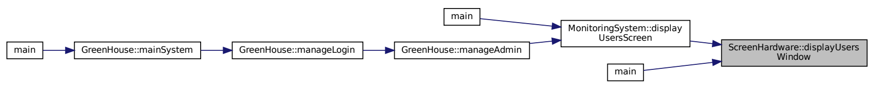
Display all Users window.

Definition at line 117 of file ScreenHardware.cpp.

```
117                                         {
118   // Aqui tengo que mostrar un menu para ver todos los usuarios
119   std::cout « "---_Display Users Window_---" « std::endl;
120 }
```

Referenced by MonitoringSystem::displayUsersScreen(), and main().

Here is the caller graph for this function:



4.19.3.12 exitWindow()

```
void ScreenHardware::exitWindow ( )
```

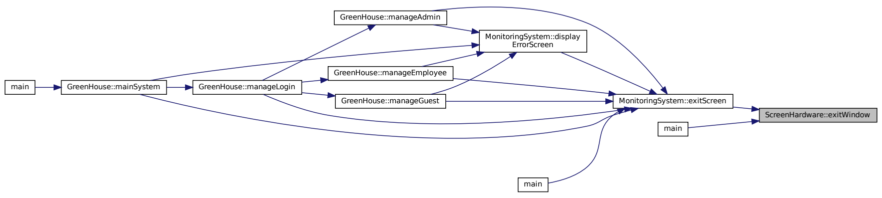
Exit the window.

Definition at line 36 of file ScreenHardware.cpp.

```
36                              {
37   // Aqui es el menu de salida
38   std::cout « "---_Exit Window_---" « std::endl;
39   std::cout « "Thanks for using our system" « std::endl;
40 }
```

Referenced by MonitoringSystem::exitScreen(), and main().

Here is the caller graph for this function:


GreenHouse::manageAdmin
MonitoringSystem::display ErrorScreen
GreenHouse::manageEmployee
main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageGuest
MonitoringSystem::exitScreen
ScreenHardware::exitWindow
main
main

4.19.3.13 initialWindow()

```
void ScreenHardware::initialWindow ( )
```

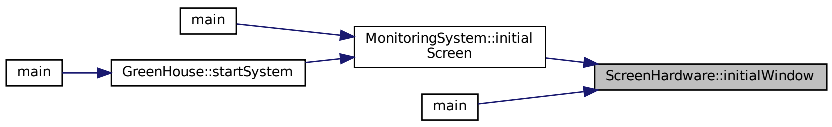
This method displays the initial window of the system.

Definition at line 29 of file ScreenHardware.cpp.

```
29                                 {
30   // Aqui tengo que mostrar un menu principal donde se de la bienvenida
31   std::cout « "---_Initial Window_---" « std::endl;
32   std::cout « "1. Login" « std::endl;
33   std::cout « "2. Exit" « std::endl;
34 }
```

Referenced by MonitoringSystem::initialScreen(), and main().

Here is the caller graph for this function:


main
MonitoringSystem::initial Screen
main
GreenHouse::startSystem
ScreenHardware::initialWindow
main

4.19.3.14 loginWindow()

```
void ScreenHardware::loginWindow ( )
```

This method displays the login window of the system.

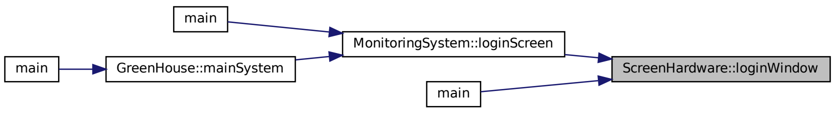
Definition at line 42 of file ScreenHardware.cpp.

```
42                                  {
43    // Este es el menu de login
44    std::cout « "---_Login Window_---" « std::endl;
45    std::cout « ASK_DATA « std::endl;
46    std::cout
47        « "First the name(intro), then the password(intro), then the nif(intro)"
48        « std::endl;
49 }
```

References ASK_DATA.

Referenced by MonitoringSystem::loginScreen(), and main().

Here is the caller graph for this function:



4.19.3.15 mainWindowAdmin()

```
void ScreenHardware::mainWindowAdmin ( )
```

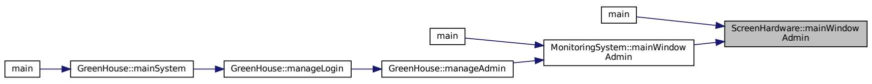
This method displays the main window of the system for the admin.

Definition at line 51 of file ScreenHardware.cpp.

```
51                                      {
52    // Aqui tengo que mostrar un menu principal donde se muestran todas las
53    // opciones para admins
54    std::cout « "---_Main Window Admin_---" « std::endl;
55    std::cout « "1. Create User" « std::endl;
56    std::cout « "2. Delete User" « std::endl;
57    std::cout « "3. Update User" « std::endl;
58    std::cout « "4. Display Users" « std::endl;
59    std::cout « "5. Create Sensor" « std::endl;
60    std::cout « "6. Delete Sensor" « std::endl;
61    std::cout « "7. Display Sensors" « std::endl;
62    std::cout « "8. Display Alarms" « std::endl;
63    std::cout « "9. Turn On/Off System" « std::endl;
64    std::cout « "10. Save Users" « std::endl;
65    std::cout « "11. Save Sensors" « std::endl;
66    std::cout « "12. Exit & Save all" « std::endl;
67 }
```

Referenced by main(), and MonitoringSystem::mainWindowAdmin().

Here is the caller graph for this function:



4.19.3.16 mainWindowEmployee()

```
void ScreenHardware::mainWindowEmployee ( )
```

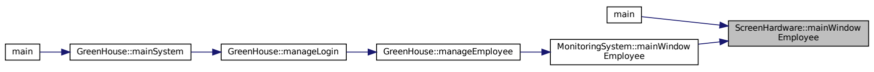
This method displays the main window of the system for the employee.

Definition at line 69 of file ScreenHardware.cpp.

```
                                      {
    // Aqui tengo que mostrar un menu principal donde se muestran todas las
    // opciones para employees
    std::cout « "---_Main Window Employee_---" « std::endl;
    std::cout « "1. Create Sensor" « std::endl;
    std::cout « "2. Delete Sensor" « std::endl;
    std::cout « "3. Display Sensors" « std::endl;
    std::cout « "4. Display Alarms" « std::endl;
    std::cout « "5. Turn On/Off System" « std::endl;
    std::cout « "6. Save Sensors" « std::endl;
    std::cout « "7. Exit and Save Sensors" « std::endl;
}
```

Referenced by main(), and MonitoringSystem::mainWindowEmployee().

Here is the caller graph for this function:


main
main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageEmployee
MonitoringSystem::mainWindow Employee
ScreenHardware::mainWindow Employee

4.19.3.17 mainWindowGuest()

```
void ScreenHardware::mainWindowGuest ( )
```

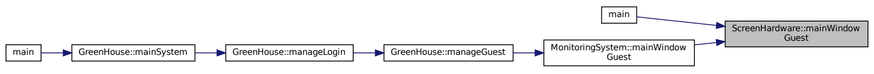
This method displays the main window of the system for the guest.

Definition at line 82 of file ScreenHardware.cpp.

```
                                   {
    // Aqui tengo que mostrar un menu principal donde se muestran todas las
    // opciones para guests
    std::cout « "---_Main Window Guest_---" « std::endl;
    std::cout « "1. Display Sensors" « std::endl;
    std::cout « "2. Display Alarms" « std::endl;
    std::cout « "3. Save Sensors" « std::endl;
    std::cout « "4. Exit and Save Sensors" « std::endl;
}
```

Referenced by main(), and MonitoringSystem::mainWindowGuest().

Here is the caller graph for this function:



4.19.3.18 saveAlarmWindow()

```
void ScreenHardware::saveAlarmWindow ( )
```

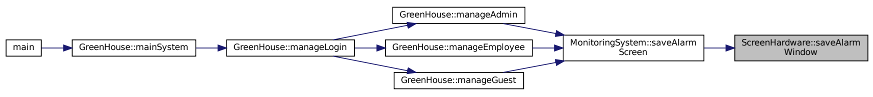
Save Alarm Window object.

Definition at line 160 of file ScreenHardware.cpp.

```
160                                      {
161   // Aqui tengo que mostrar un menu para guardar una alarma
162   std::cout « "---_Save Alarm Window_---" « std::endl;
163   std::cout « "Saving all the sensors(txt/dat)..." « std::endl;
164 }
```

Referenced by MonitoringSystem::saveAlarmScreen().

Here is the caller graph for this function:



4.19.3.19 turnOnOffSystemWindow()

```
void ScreenHardware::turnOnOffSystemWindow ( )
```

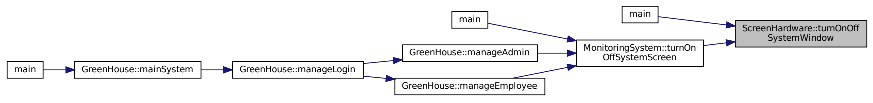
Turn on or off the system window.

Definition at line 143 of file ScreenHardware.cpp.

```
143                                            {
144   // Aqui tengo que mostrar un menu para encender y apagar el sistema
145   std::cout « "---_Turn On/Off System Window_---" « std::endl;
146   std::cout « "1. ON \n2. OFF" « std::endl;
147 }
```

Referenced by main(), and MonitoringSystem::turnOnOffSystemScreen().

Here is the caller graph for this function:


main
main
main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageEmployee
MonitoringSystem::turnOn OffSystemScreen
ScreenHardware::turnOnOff SystemWindow

4.19.3.20 updateUserWindow()

```
void ScreenHardware::updateUserWindow ( )
```

Update a User object window.

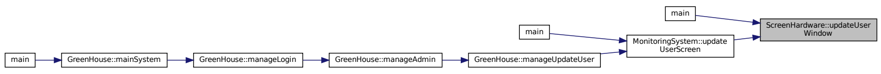
Definition at line 107 of file ScreenHardware.cpp.

```
107                                     {
108   // Aqui tengo que mostrar un menu para actualizar un usuario
109   std::cout « "---_Update User Window_---" « std::endl;
110   std::cout « ASK_DATA « std::endl;
111   std::cout « "You can change the role / the email / if you want to change "
112                "password delete the user and create a new one"
113             « std::endl;
114   std::cout « USER_PROMPT « std::endl;
115 }
```

References ASK_DATA, and USER_PROMPT.

Referenced by main(), and MonitoringSystem::updateUserScreen().

Here is the caller graph for this function:


main
GreenHouse::mainSystem
GreenHouse::manageLogin
GreenHouse::manageAdmin
GreenHouse::manageUpdateUser
main
MonitoringSystem::update UserScreen
main
ScreenHardware::updateUser Window

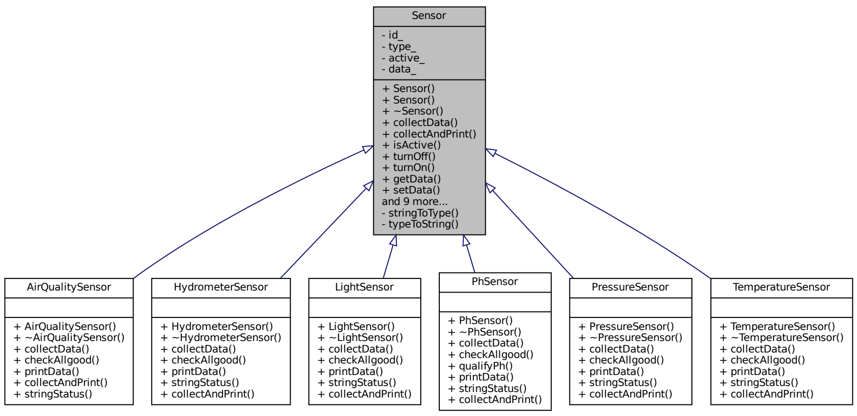
The documentation for this class was generated from the following files:

- src/ScreenHardware.h
- src/ScreenHardware.cpp

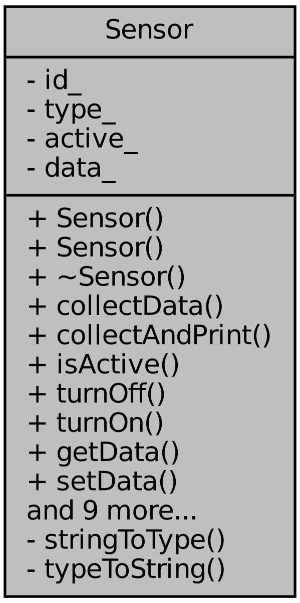
4.20 Sensor Class Reference

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

Inheritance diagram for Sensor:



Collaboration diagram for Sensor:



Public Types

- enum Types {
  NONE , TEMPERATURE , AIR_QUALITY , HYDROMETER ,
  PRESSURE , LIGHT_SENSOR , PH_SENSOR }

  This is the enum Types. It contains the types of the sensors.

Public Member Functions

- Sensor ()

  Construct a new Sensor object.
- Sensor (int id, Types type, bool active)

  Construct a new Sensor object.
- virtual ~Sensor ()

  Destroy the Sensor object.
- virtual void collectData ()

  Collect data of the Sensor.
- virtual void collectAndPrint ()

  Collect and print the data of the Sensor.
- bool isActive () const

  Return if is active or not the sensor.
- void turnOff ()

  Turn off the sensor.
- void turnOn ()

Turn on the sensor.
- float getData () const

  Get the Data object.
- void setData (float data)

  Set the Data object.
- int getId () const

  Get the Id object.
- void setId (int newid)

  Set the Id object.
- std::string getType () const

  Get the Type object.
- void setType (std::string newtype)

  Set the Type object.
- virtual bool checkAllgood () const

  Check if the Sensor is working properly.
- bool operator< (const Sensor &Sensor) const

  Operator < overload.
- bool operator> (const Sensor &Sensor) const

  Operator > overload.
- bool operator== (const Sensor &Sensor) const

  Operator == overload.
- virtual void printData () const

  Print the data of the Sensor.

Private Member Functions

- Types stringToType (const std::string &type) const

  Convert the string to the type.
- std::string typeToString (Types type) const

  Convert the type to the string.

Private Attributes

- int id_

  The id of the sensor.
- Types type_

  The type of the sensor.
- bool active_

  The state of the sensor.
- float data_

  The data of the sensor.

Friends

- std::ostream & operator<< (std::ostream &os, const Sensor &Sensor)

  Operator << overload.
- std::istream & operator>> (std::istream &is, Sensor &Sensor)

  Operator >> overload.

4.20.1 Detailed Description

Definition at line 13 of file Sensor.h.

4.20.2 Member Enumeration Documentation

4.20.2.1 Types

`enum Sensor::Types`

This is the enum Types. It contains the types of the sensors.

Enumerator

NONE	
TEMPERATURE	
AIR_QUALITY	
HYDROMETER	
PRESSURE	
LIGHT_SENSOR	
PH_SENSOR	

Definition at line 19 of file Sensor.h.

```
               {
    NONE,
    TEMPERATURE,
    AIR_QUALITY,
    HYDROMETER,
    PRESSURE,
    LIGHT_SENSOR,
    PH_SENSOR,
  };
```

4.20.3 Constructor & Destructor Documentation

4.20.3.1 Sensor() [1/2]

`Sensor::Sensor ( )`

Construct a new Sensor object.

Creates a new Sensor object with the default values (id, type, active).

Returns

Sensor object

Definition at line 6 of file Sensor.cpp.
```
                 {
  id_ = -1;
  type_ = Types::NONE;
  active_ = false;
   data_ = -1;
}
```

4.20.3.2 Sensor() [2/2]

```
Sensor::Sensor (
              int id,
              Types type,
              bool active )  [explicit]
```

Construct a new Sensor object.

Creates a new Sensor object with the values passed as parameters.

Parameters

id	of the sensor
type	of the sensor
active	of the sensor

Returns

Sensor object

Definition at line 13 of file Sensor.cpp.
```
                                               {
   id_ = id;
   type_ = type;
   active_ = active;
   data_ = -1;
}
```

4.20.3.3 ∼Sensor()

```
Sensor::∼Sensor ( )  [virtual]
```

Destroy the Sensor object.

Definition at line 20 of file Sensor.cpp.
```
{}
```

4.20.4 Member Function Documentation

4.20.4.1 checkAllgood()

```
bool Sensor::checkAllgood ( ) const  [virtual]
```

Check if the Sensor is working properly.

Returns

true if the Sensor is working properly

false if the Sensor is not working properly

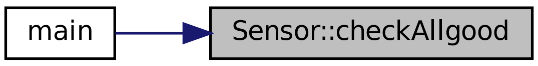
Reimplemented in TemperatureSensor, PressureSensor, PhSensor, LightSensor, HydrometerSensor, and AirQualitySensor.

Definition at line 128 of file Sensor.cpp.

```
{ return true; }
```

Referenced by main().

Here is the caller graph for this function:



4.20.4.2 collectAndPrint()

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

Collect and print the data of the Sensor.

Reimplemented in TemperatureSensor, PressureSensor, PhSensor, LightSensor, HydrometerSensor, and AirQualitySensor.

Definition at line 31 of file Sensor.cpp.

```
                              {
  collectData();
  printData();
}
```

4.20.4.3 collectData()

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

Collect data of the Sensor.

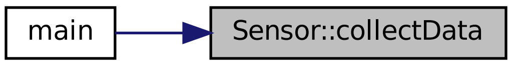
Reimplemented in TemperatureSensor, PressureSensor, PhSensor, LightSensor, HydrometerSensor, and AirQualitySensor.

Definition at line 22 of file Sensor.cpp.

```
                             {
    cout « "Collecting data from sensor id" « id_ « " wich is: " « getType()
         « endl;
    setData(-10000);
    // This function will be implemented in the derived classes
    // en la clase derivada se implementara la funcion, generaremos de manera
    // aleatoria el valor y despues lo asignaremos al atributo data_ con setData()
}
```

Referenced by main().

Here is the caller graph for this function:



4.20.4.4 getData()

```
float Sensor::getData ( ) const
```

Get the Data object.

Returns

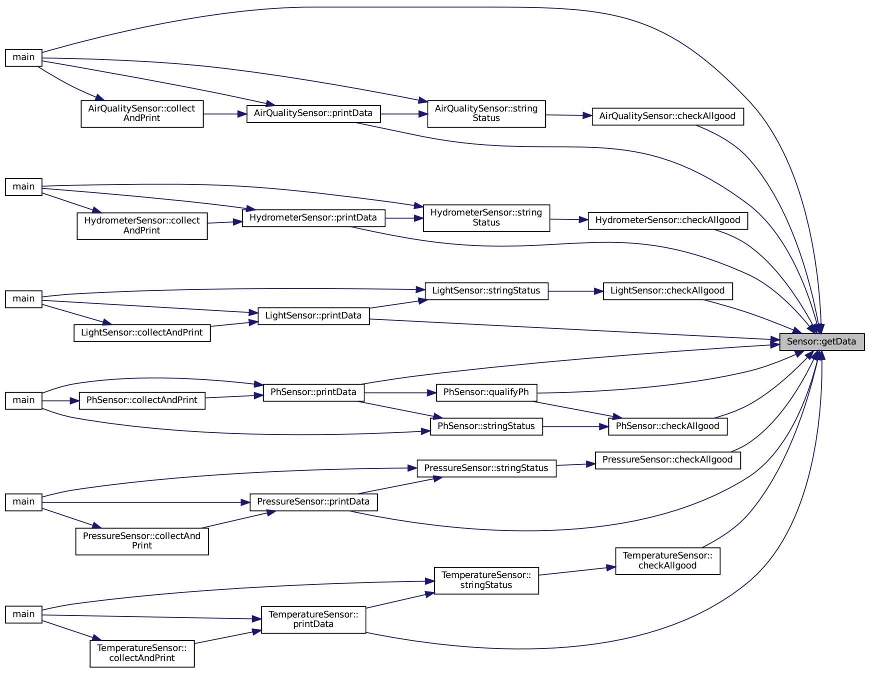
float

Definition at line 42 of file Sensor.cpp.

```
{ return data_; }
```

Referenced by AirQualitySensor::checkAllgood(), HydrometerSensor::checkAllgood(), LightSensor::checkAllgood(), PhSensor::checkAllgood(), PressureSensor::checkAllgood(), TemperatureSensor::checkAllgood(), main(), Air←QualitySensor::printData(), HydrometerSensor::printData(), LightSensor::printData(), PhSensor::printData(), PressureSensor::printData(), TemperatureSensor::printData(), and PhSensor::qualifyPh().

Here is the caller graph for this function:


main
AirQualitySensor::collect AndPrint
AirQualitySensor::printData
AirQualitySensor::string Status
AirQualitySensor::checkAllgood
main
HydrometerSensor::collect AndPrint
HydrometerSensor::printData
HydrometerSensor::string Status
HydrometerSensor::checkAllgood
main
LightSensor::collectAndPrint
LightSensor::printData
LightSensor::stringStatus
LightSensor::checkAllgood
Sensor::getData
main
PhSensor::collectAndPrint
PhSensor::printData
PhSensor::qualifyPh
PhSensor::stringStatus
PhSensor::checkAllgood
main
PressureSensor::collectAnd Print
PressureSensor::printData
PressureSensor::stringStatus
PressureSensor::checkAllgood
main
TemperatureSensor:: collectAndPrint
TemperatureSensor:: printData
TemperatureSensor:: stringStatus
TemperatureSensor:: checkAllgood

4.20.4.5 getId()

```
int Sensor::getId ( ) const
```

Get the Id object.

Returns

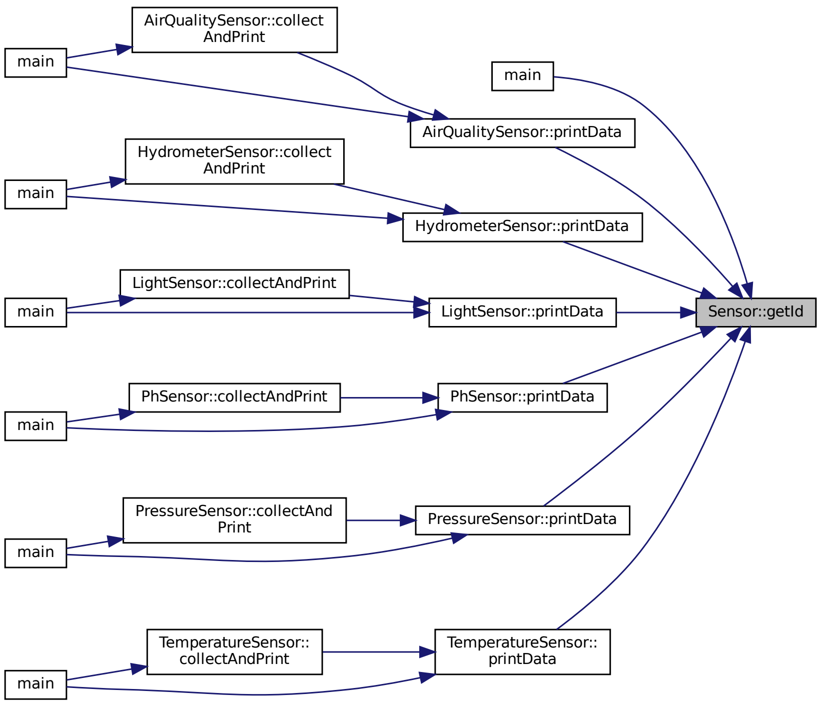
int

Definition at line 46 of file Sensor.cpp.

```
46 { return id_; }
```

Referenced by main(), AirQualitySensor::printData(), HydrometerSensor::printData(), LightSensor::printData(), PhSensor::printData(), PressureSensor::printData(), and TemperatureSensor::printData().

Here is the caller graph for this function:


main
AirQualitySensor::collect AndPrint
main
AirQualitySensor::printData
main
HydrometerSensor::collect AndPrint
HydrometerSensor::printData
main
LightSensor::collectAndPrint
LightSensor::printData
Sensor::getId
main
PhSensor::collectAndPrint
PhSensor::printData
main
PressureSensor::collectAnd Print
PressureSensor::printData
main
TemperatureSensor:: collectAndPrint
TemperatureSensor:: printData
main

4.20.4.6 getType()

`std::string Sensor::getType ( ) const`

Get the Type object.

Returns

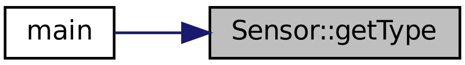
std::string

Definition at line 50 of file Sensor.cpp.

```
                                {
  std::string type = typeToString(type_);
  return type;
}
```

Referenced by main().

Here is the caller graph for this function:



4.20.4.7 isActive()

```
bool Sensor::isActive ( ) const
```

Return if is active or not the sensor.

Returns

true

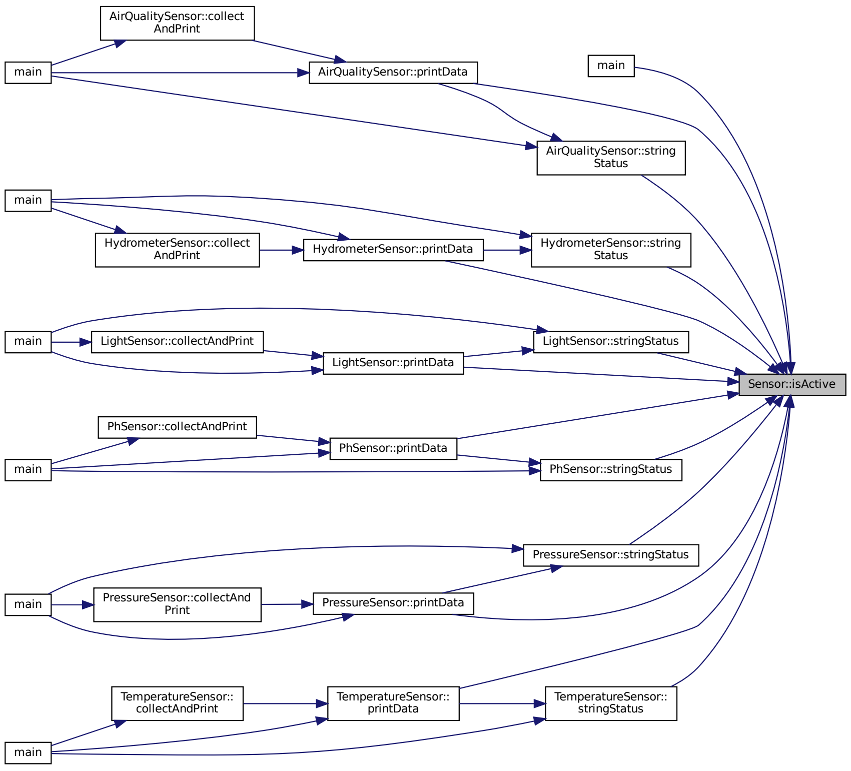
false

Definition at line 36 of file Sensor.cpp.

```
36 { return active_; }
```

Referenced by main(), AirQualitySensor::printData(), HydrometerSensor::printData(), LightSensor::printData(), PhSensor::printData(), PressureSensor::printData(), TemperatureSensor::printData(), AirQualitySensor::string↩ Status(), HydrometerSensor::stringStatus(), LightSensor::stringStatus(), PhSensor::stringStatus(), Pressure↩ Sensor::stringStatus(), and TemperatureSensor::stringStatus().

Here is the caller graph for this function:


AirQualitySensor::collect AndPrint
main
AirQualitySensor::printData
main
AirQualitySensor::string Status
main
HydrometerSensor::collect AndPrint
HydrometerSensor::printData
HydrometerSensor::string Status
main
LightSensor::collectAndPrint
LightSensor::stringStatus
LightSensor::printData
Sensor::isActive
PhSensor::collectAndPrint
PhSensor::printData
main
PhSensor::stringStatus
PressureSensor::stringStatus
main
PressureSensor::collectAnd Print
PressureSensor::printData
TemperatureSensor:: collectAndPrint
TemperatureSensor:: printData
TemperatureSensor:: stringStatus
main

4.20.4.8 operator<()

```
bool Sensor::operator< (
            const Sensor & Sensor ) const
```

Operator < overload.

Parameters

Sensor	

Returns

true

false

Definition at line 94 of file Sensor.cpp.

```
{ return id_ < Sensor.id_; }
```

References id_.

4.20.4.9 operator==()

```
bool Sensor::operator== (
            const Sensor & Sensor ) const
```

Operator == overload.

Parameters

Sensor	

Returns

true

false

Definition at line 98 of file Sensor.cpp.

```
                                            {
    return id_ == Sensor.id_;
}
```

References id_.

4.20.4.10 operator>()

```
bool Sensor::operator> (
            const Sensor & Sensor ) const
```

Operator > overload.

Parameters

Sensor	

Returns

true

false

Definition at line 96 of file Sensor.cpp.

```
{ return id_ > Sensor.id_; }
```

References id_.

4.20.4.11 printData()

```
void Sensor::printData ( ) const  [virtual]
```

Print the data of the Sensor.

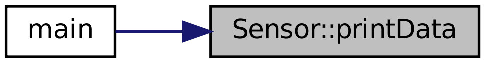
Reimplemented in TemperatureSensor, PressureSensor, PhSensor, LightSensor, HydrometerSensor, and AirQualitySensor.

Definition at line 122 of file Sensor.cpp.

```
                              {
  cout « "This prints the data of sensor " « getType() « " with id" « id_
       « " please use the correct function to print the data" « endl;
  // This function will be implemented in the derived classes
}
```

Referenced by main().

Here is the caller graph for this function:



4.20.4.12 setData()

```
void Sensor::setData (
              float data )
```

Set the Data object.

Parameters

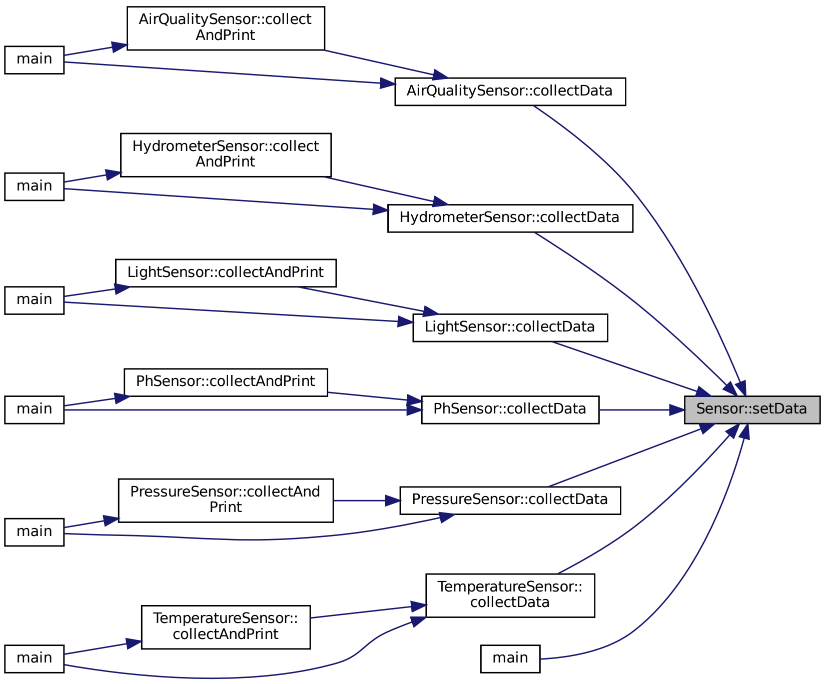
data	

Definition at line 44 of file Sensor.cpp.

```
44 { data_ = data; }
```

Referenced by AirQualitySensor::collectData(), HydrometerSensor::collectData(), LightSensor::collectData(), Ph↩
Sensor::collectData(), PressureSensor::collectData(), TemperatureSensor::collectData(), and main().

Here is the caller graph for this function:


main
AirQualitySensor::collect AndPrint
AirQualitySensor::collectData
main
HydrometerSensor::collect AndPrint
HydrometerSensor::collectData
main
LightSensor::collectAndPrint
LightSensor::collectData
main
PhSensor::collectAndPrint
PhSensor::collectData
Sensor::setData
main
PressureSensor::collectAnd Print
PressureSensor::collectData
main
TemperatureSensor:: collectAndPrint
TemperatureSensor:: collectData
main

4.20.4.13 setId()

```
void Sensor::setId (
            int newid )
```

Set the Id object.

Parameters

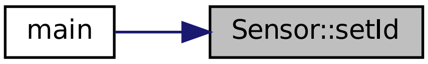
newid	

Definition at line 48 of file Sensor.cpp.

```
48 { id_ = newid; }
```

Referenced by main().

Here is the caller graph for this function:



4.20.4.14 setType()

```
void Sensor::setType (
            std::string newtype )
```

Set the Type object.

Parameters

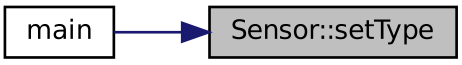
newtype	

Definition at line 55 of file Sensor.cpp.

```
55 { type_ = stringToType(newtype); }
```

Referenced by main().

Here is the caller graph for this function:



4.20.4.15 stringToType()

```
Sensor::Types Sensor::stringToType (
            const std::string & type ) const  [private]
```

Convert the string to the type.

Parameters

type	

Returns

Types

Definition at line 57 of file Sensor.cpp.

```
                                                              {
  if (type == "TEMPERATURE") {
    return Types::TEMPERATURE;
  } else if (type == "AIR_QUALITY") {
    return Types::AIR_QUALITY;
  } else if (type == "HYDROMETER") {
    return Types::HYDROMETER;
  } else if (type == "PRESSURE") {
    return Types::PRESSURE;
  } else if (type == "LIGHT_SENSOR") {
    return Types::LIGHT_SENSOR;
  } else if (type == "PH_SENSOR") {
    return Types::PH_SENSOR;
  } else {
    return Types::NONE;
  }
}
```

4.20.4.16 turnOff()

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

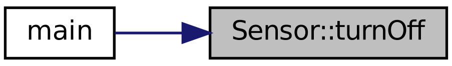
Turn off the sensor.

Definition at line 38 of file Sensor.cpp.

```
{ active_ = false; }
```

Referenced by main().

Here is the caller graph for this function:



4.20.4.17 turnOn()

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

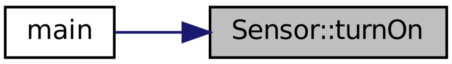
Turn on the sensor.

Definition at line 40 of file Sensor.cpp.

```
{ active_ = true; }
```

Referenced by main().

Here is the caller graph for this function:



4.20.4.18 typeToString()

```
std::string Sensor::typeToString (
            Types type ) const  [private]
```

Convert the type to the string.

Parameters

type	

Returns

std::string

Definition at line 75 of file Sensor.cpp.

```
                                                {
  switch (type) {
  case Types::TEMPERATURE:
    return "TEMPERATURE";
  case Types::AIR_QUALITY:
    return "AIR_QUALITY";
  case Types::HYDROMETER:
    return "HYDROMETER";
  case Types::PRESSURE:
    return "PRESSURE";
  case Types::LIGHT_SENSOR:
    return "LIGHT_SENSOR";
  case Types::PH_SENSOR:
    return "PH_SENSOR";
  default:
    return "NONE";
  }
}
```

4.20.5 Friends And Related Function Documentation

4.20.5.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const Sensor & Sensor )  [friend]
```

Operator << overload.

Parameters

os	
Sensor	

Returns

std::ostream&

Definition at line 102 of file Sensor.cpp.

```
                                                              {
  os « "ID: " « Sensor.getId() « " Type: " « Sensor.getType()
     « " Active: " « Sensor.isActive() « " Data: " « Sensor.getData()
     « std::endl;
  return os;
}
```

4.20.5.2 operator>>

```
std::istream& operator>> (
            std::istream & is,
            Sensor & Sensor )  [friend]
```

Operator >> overload.

Parameters

is	
Sensor	

Returns

std::istream&

Definition at line 109 of file Sensor.cpp.

```
                                                        {
  cout « "Enter sensor ID: ";
  is » sensor.id_;
  cout « "Enter the type: ";
```

```
    std::string type;
    is » type;
    sensor.setType(type);
    cout « "Enter sensor active: ";
    is » sensor.active_;

    return is;
}
```

4.20.6 Member Data Documentation

4.20.6.1 active_

`bool Sensor::active_ [private]`

The state of the sensor.

Definition at line 194 of file Sensor.h.

4.20.6.2 data_

`float Sensor::data_ [private]`

The data of the sensor.

Definition at line 199 of file Sensor.h.

4.20.6.3 id_

`int Sensor::id_ [private]`

The id of the sensor.

Definition at line 184 of file Sensor.h.

Referenced by operator<(), operator==(), and operator>().

4.20.6.4 type_

`Types Sensor::type_ [private]`

The type of the sensor.

Definition at line 189 of file Sensor.h.

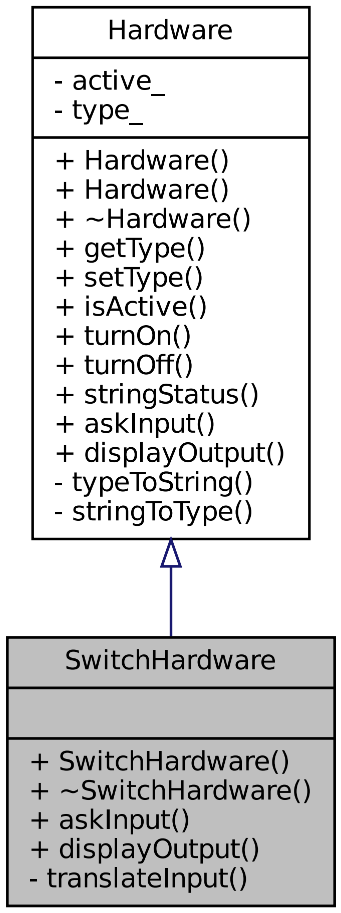
The documentation for this class was generated from the following files:

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

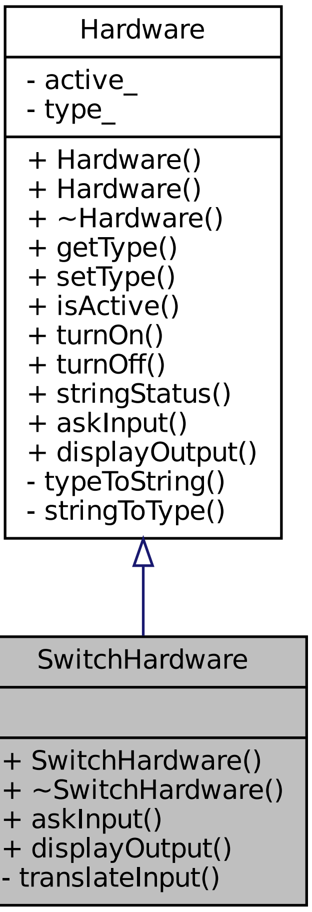
4.21 SwitchHardware Class Reference

```
#include <SwitchHardware.h>
```

Inheritance diagram for SwitchHardware:



Collaboration diagram for SwitchHardware:



Public Member Functions

- SwitchHardware (bool active)

  Construct a new Switch Hardware object.
- ∼SwitchHardware () override

  Destroy the Switch Hardware object.
- int askInput () override

  Ask for an input.
- void displayOutput () const override

  Display the output of the Switch Hardware.

Private Member Functions

- void translateInput (int input)

  The data of the Switch Hardware.

Additional Inherited Members

4.21.1 Detailed Description

Definition at line 14 of file SwitchHardware.h.

4.21.2 Constructor & Destructor Documentation

4.21.2.1 SwitchHardware()

```
SwitchHardware::SwitchHardware (
             bool active )  [explicit]
```

Construct a new Switch Hardware object.

Parameters

active	

Returns

SwitchHardware object

Definition at line 7 of file SwitchHardware.cpp.

```
8     : Hardware(active, Hardware::Types_Hardware::SWITCH) {}
```

4.21.2.2 ∼SwitchHardware()

```
SwitchHardware::∼SwitchHardware ( )  [override]
```

Destroy the Switch Hardware object.

Definition at line 10 of file SwitchHardware.cpp.

```
10 {}
```

4.21.3 Member Function Documentation

4.21.3.1 askInput()

```
int SwitchHardware::askInput ( )  [override], [virtual]
```

Ask for an input.

Returns

int

Reimplemented from Hardware.

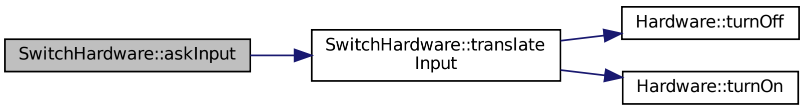
Definition at line 20 of file SwitchHardware.cpp.

```
                              {
  int input;
  std::string input_string;
  // Preguntamos al usuario si ON of OFF y luego el input se lo pasamos a
  // translateInputToBool
  std::cout « "Switch wating a input(ON/OFF)..." « std::endl;
  std::cin » input_string;
  if (input_string == "ON") {
    input = 1;
  }
  if (input_string == "OFF") {
    input = 0;
  }

  translateInput(input);
  return input;
  // El valor de active_ sera true o false dependiendo de si se activa o se
  // desactiva el switch
}
```

References translateInput().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

main
SwitchHardware::askInput

4.21.3.2 displayOutput()

```
void SwitchHardware::displayOutput ( ) const  [override], [virtual]
```

Display the output of the Switch Hardware.

Reimplemented from Hardware.

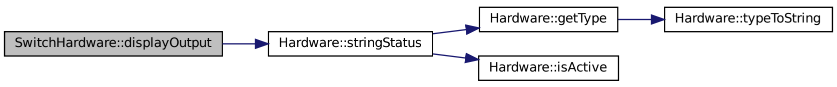
Definition at line 40 of file SwitchHardware.cpp.

```
                                              {
  std::cout « stringStatus() « std::endl;
}
```

References Hardware::stringStatus().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

main
SwitchHardware::displayOutput

4.21.3.3 translateInput()

```
void SwitchHardware::translateInput (
               int input )  [private]
```

The data of the Switch Hardware.

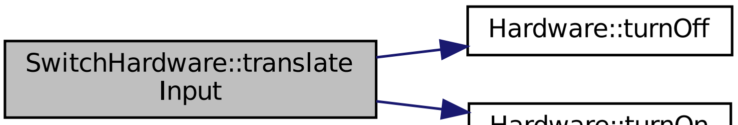
Definition at line 12 of file SwitchHardware.cpp.

```
                                                  {
  if (input) {
    turnOn();
  } else {
    turnOff();
  }
}
```

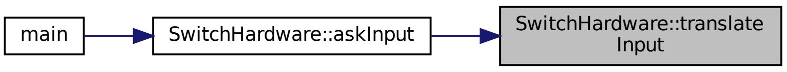
References Hardware::turnOff(), and Hardware::turnOn().

Referenced by askInput().

Here is the call graph for this function:



Here is the caller graph for this function:



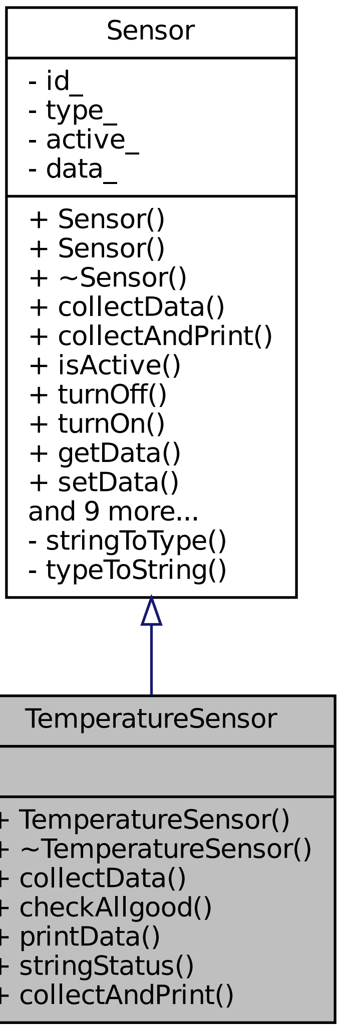
The documentation for this class was generated from the following files:

- src/SwitchHardware.h
- src/SwitchHardware.cpp

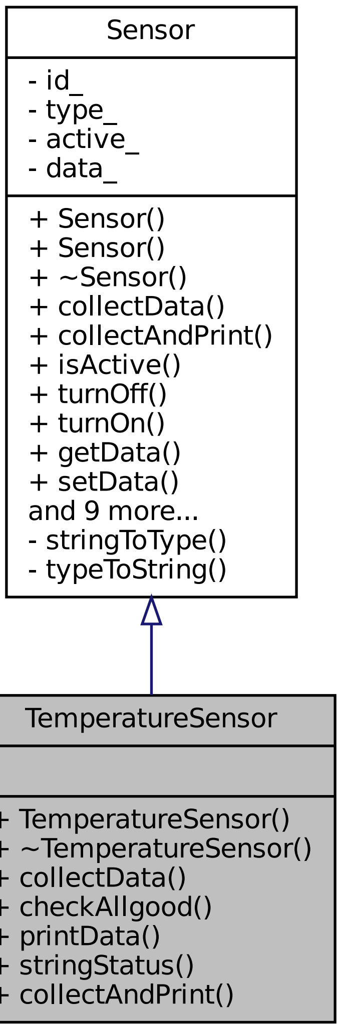
4.22 TemperatureSensor Class Reference

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

Inheritance diagram for TemperatureSensor:



Collaboration diagram for TemperatureSensor:



Public Member Functions

- TemperatureSensor (int id, bool active)

  Construct a new Temperature Sensor object.
- ∼TemperatureSensor () override

  Destroy the Temperature Sensor object.
- void collectData () override

  Collect data of the Temperature Sensor.
- bool checkAllgood () const override

  Check if the Temperature Sensor is working properly.
- void printData () const override

  Print the data of the Temperature Sensor.
- std::string stringStatus () const

String status of the Temperature Sensor.

- void collectAndPrint ()

  Collect and print the data of the Temperature Sensor.

Friends

- std::ostream & operator<< (std::ostream &os, const TemperatureSensor &sensor)

  Overloaded operator<<.

Additional Inherited Members

4.22.1 Detailed Description

Definition at line 15 of file TemperatureSensor.h.

4.22.2 Constructor & Destructor Documentation

4.22.2.1 TemperatureSensor()

```
TemperatureSensor::TemperatureSensor (
              int id,
              bool active )  [explicit]
```

Construct a new Temperature Sensor object.

Parameters

id	
active	

Returns

TemperatureSensor object

Definition at line 10 of file TemperatureSensor.cpp.

```
    : Sensor(id, Sensor::Types::TEMPERATURE, active) {}
```

4.22.2.2 ∼TemperatureSensor()

```
TemperatureSensor::∼TemperatureSensor ( )  [override]
```

Destroy the Temperature Sensor object.

Definition at line 13 of file TemperatureSensor.cpp.

```
{}
```

4.22.3 Member Function Documentation

4.22.3.1 checkAllgood()

```
bool TemperatureSensor::checkAllgood ( ) const  [override], [virtual]
```

Check if the Temperature Sensor is working properly.

Returns

true if the Temperature Sensor is working properly

false if the Temperature Sensor is not working properly

Reimplemented from Sensor.

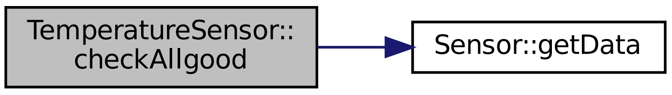
Definition at line 25 of file TemperatureSensor.cpp.

```
                                              {
  float data = Sensor::getData();
  // Entre 20 y 30 estara bien la temperatura, en el resto de los casos no
  // estara bien
  if (data >= 20.0f && data <= 30.0f) {
    return true;
  } else {
    return false;
  }
}
```

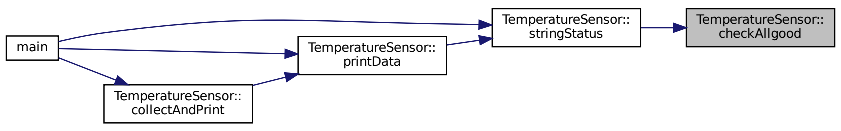
References Sensor::getData().

Referenced by stringStatus().

Here is the call graph for this function:



Here is the caller graph for this function:



4.22.3.2 collectAndPrint()

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

Collect and print the data of the Temperature Sensor.

Reimplemented from Sensor.

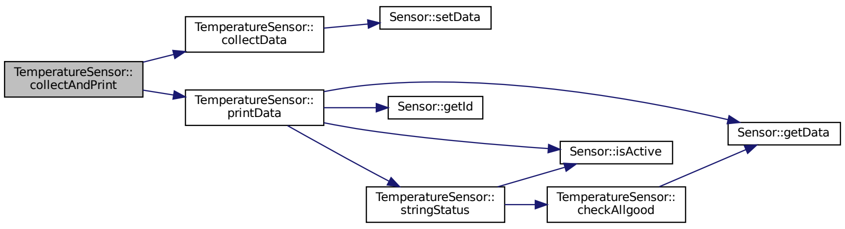
Definition at line 66 of file TemperatureSensor.cpp.

```
                                              {
    collectData();
    printData();
 }
```

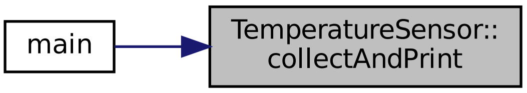
References collectData(), and printData().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.22.3.3 collectData()

```
void TemperatureSensor::collectData ( )  [override], [virtual]
```

Collect data of the Temperature Sensor.

This method collects the data of the Temperature Sensor and stores it in the data attribute.

Reimplemented from Sensor.

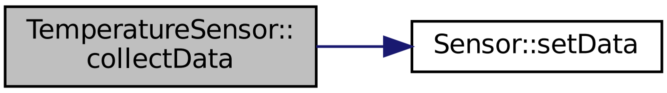
Definition at line 15 of file TemperatureSensor.cpp.

```
                                     {
    // Generamos una temperatura aleatoria entre 19 y 31 grados Centigrados

    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_real_distribution<> dis(19.0, 31.0);
    float temperature = dis(gen);
    Sensor::setData(temperature);
}
```

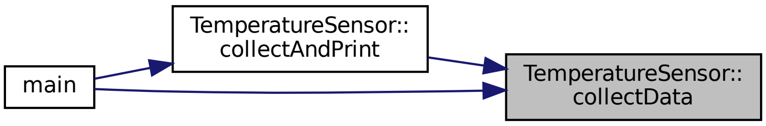
References Sensor::setData().

Referenced by collectAndPrint(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.22.3.4 printData()

```
void TemperatureSensor::printData ( ) const  [override], [virtual]
```

Print the data of the Temperature Sensor.

Reimplemented from Sensor.

Definition at line 53 of file TemperatureSensor.cpp.

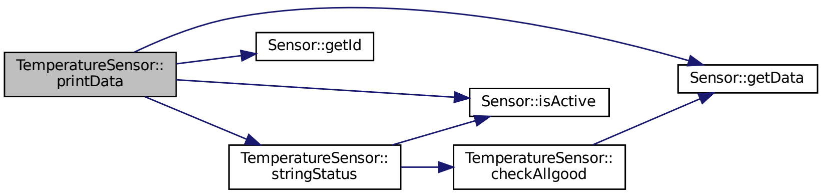
```
                                         {
    // Imprimimos la temperatura actual del sensor, el id que tiene el sensor, y
    // si todo esta bien o no con (True/False)
    if (Sensor::isActive()) {
      std::cout « "Temperature Sensor with "
                « "ID: " « Sensor::getId() « " - Data: " « Sensor::getData()
                « " C° - Status: " « stringStatus() « endl;
    } else {
```

```
    std::cout « "Temperature Sensor with "
              « "ID: " « Sensor::getId() « " - INACTIVE" « endl;
  }
}
```

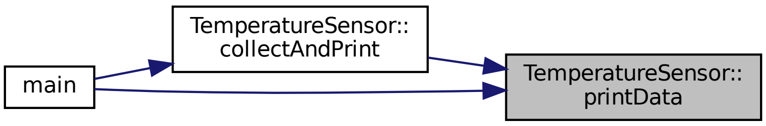
References Sensor::getData(), Sensor::getId(), Sensor::isActive(), and stringStatus().

Referenced by collectAndPrint(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.22.3.5 stringStatus()

```
std::string TemperatureSensor::stringStatus ( ) const
```

String status of the Temperature Sensor.

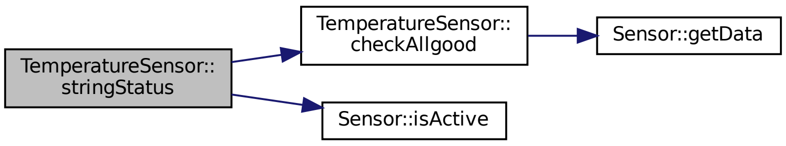
Definition at line 41 of file TemperatureSensor.cpp.

```
                                                   {
  if (Sensor::isActive()) {
    if (this->checkAllgood()) {
      return "ACTIVE - GOOD STATUS";
    } else {
      return "ACTIVE - BAD STATUS";
    }
  } else {
    return "INACTIVE";
  }
}
```

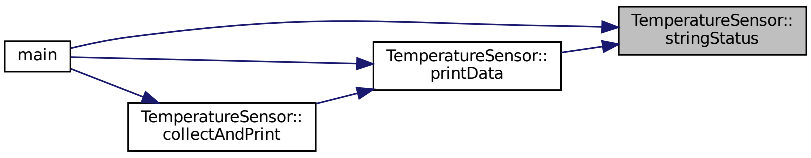
References checkAllgood(), and Sensor::isActive().

Referenced by main(), and printData().

Here is the call graph for this function:


TemperatureSensor:: stringStatus
TemperatureSensor:: checkAllgood
Sensor::getData
Sensor::isActive

Here is the caller graph for this function:


main
TemperatureSensor:: stringStatus
TemperatureSensor:: printData
TemperatureSensor:: collectAndPrint

4.22.4 Friends And Related Function Documentation

4.22.4.1 operator<<

```
std::ostream& operator<< (
              std::ostream & os,
              const TemperatureSensor & sensor )  [friend]
```

Overloaded operator<<.

Definition at line 36 of file TemperatureSensor.cpp.

```
                                                                     {
    sensor.printData();
    return os;
}
```

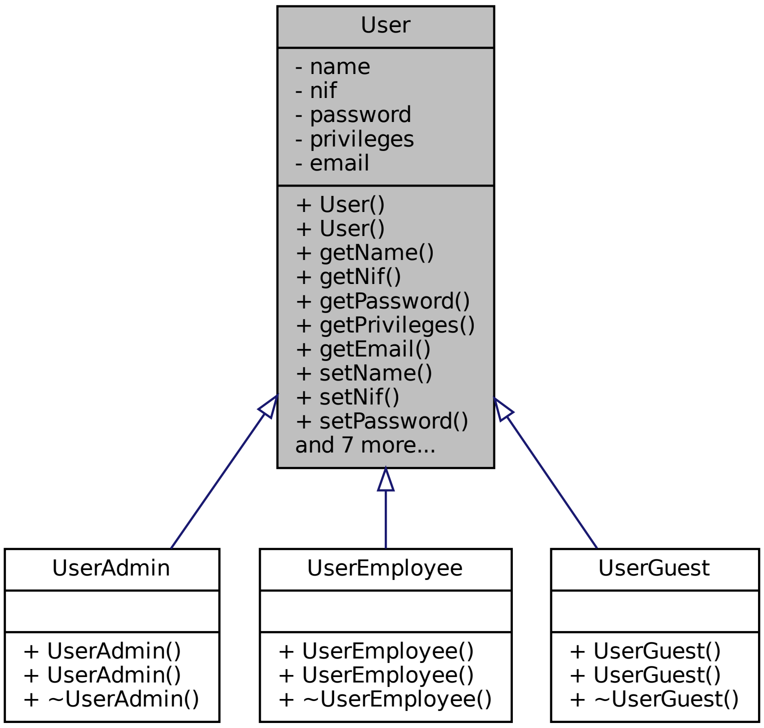
The documentation for this class was generated from the following files:

- src/TemperatureSensor.h
- src/TemperatureSensor.cpp

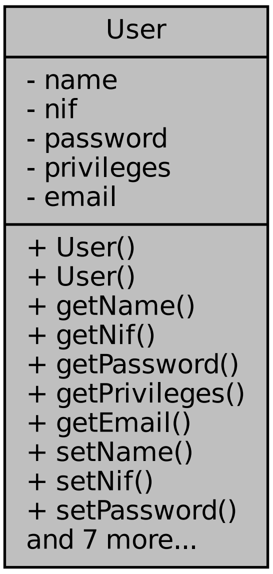
4.23 User Class Reference

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

Inheritance diagram for User:



Collaboration diagram for User:



Public Member Functions

- User ()

  Construct a new User object.
- User (const std::string name, const std::string nif, std::string password, std::string privileges, std::string email)

  Construct a new User object.
- std::string getName () const

  Get the Name object.
- std::string getNif () const

  Get the Nif object.
- std::string getPassword () const

  Get the Password object.
- std::string getPrivileges () const

  Get the Privileges object.
- std::string getEmail () const

  Get the Email object.
- void setName (const std::string name)

  Set the Name object.
- void setNif (const std::string nif)

  Set the Nif object.
- void setPassword (const std::string password)

  Set the Password object.
- virtual void setPrivileges (const std::string privileges)

  Set the Privileges object.
- void setEmail (const std::string email)

Set the Email object.
- bool operator< (const User &user) const

  Operator < overload (this comparison is made by the privileges)
- bool operator> (const User &user) const

  Operator > overload (this comparison is made by the privileges)
- bool operator== (const User &user) const

  Operator == overload (this comparison is made by the nif)
- void printUser () const

  Print the user.
- virtual ∼User ()

  Destroy the User object.

Private Attributes

- std::string name

  This is the name of the user.
- std::string nif

  This is the nif of the user.
- std::string password

  This is the password of the user.
- std::string privileges

  This is the privileges of the user.
- std::string email

  This is the email of the user.

Friends

- std::ostream & operator<< (std::ostream &os, const User &user)

  Operator << overload.
- std::istream & operator>> (std::istream &is, User &user)

  Operator >> overload.

4.23.1 Detailed Description

Definition at line 14 of file User.h.

4.23.2 Constructor & Destructor Documentation

4.23.2.1 User() [1/2]

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

Construct a new User object.

Creates a new User object with the default values (name, nif, password, privileges, email).

Returns

User object

Definition at line 5 of file User.cpp.

```
            {
    name = "";
    nif = "";
    password = "";
    privileges = "";
    email = "";
}
```

References email, name, nif, password, and privileges.

4.23.2.2 User() [2/2]

```
User::User (
              const std::string name,
              const std::string nif,
              std::string password,
              std::string privileges,
              std::string email )  [explicit]
```

Construct a new User object.

Creates a new User object with the values passed as parameters.

Parameters

name	of the user
nif	of the user
password	of the user
privileges	of the user
email	of the user

Returns

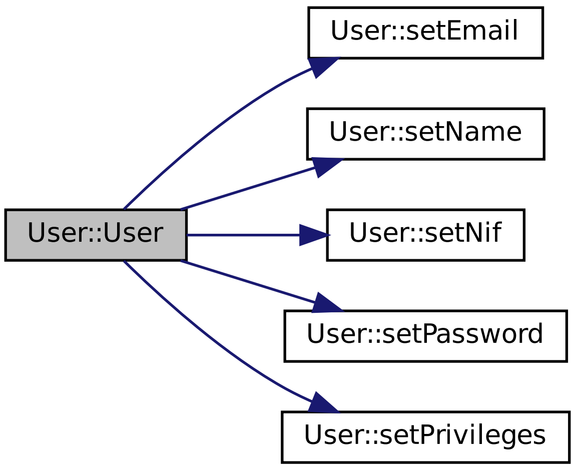
User object

Definition at line 13 of file User.cpp.

```
                                                    {
    // LLamar a los metodos set para que se encarguen de hacer las conversiones
    // necesarias
    setName(name);
    setNif(nif);
    setPassword(password);
    setPrivileges(privileges);
    setEmail(email);
}
```

References email, name, nif, password, privileges, setEmail(), setName(), setNif(), setPassword(), and set←Privileges().

Here is the call graph for this function:



4.23.2.3 ∼User()

```
User::∼User ( )  [virtual]
```

Destroy the User object.

Definition at line 25 of file User.cpp.

```
{}
```

4.23.3 Member Function Documentation

4.23.3.1 getEmail()

```
std::string User::getEmail ( ) const
```

Get the Email object.

Returns

std::string

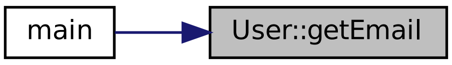
Definition at line 35 of file User.cpp.

```
35 { return email; }
```

References email.

Referenced by main().

Here is the caller graph for this function:



4.23.3.2 getName()

```
std::string User::getName ( ) const
```

Get the Name object.

Returns

std::string

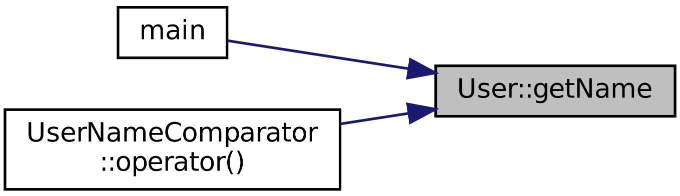
Definition at line 27 of file User.cpp.

```
27 { return name; }
```

References name.

Referenced by main(), and UserNameComparator::operator()().

Here is the caller graph for this function:



4.23.3.3 getNif()

```
std::string User::getNif ( ) const
```

Get the Nif object.

Returns

std::string

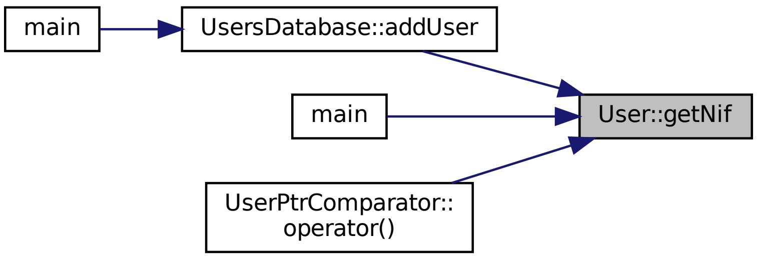
Definition at line 29 of file User.cpp.

```
29 { return nif; }
```

References nif.

Referenced by UsersDatabase::addUser(), main(), and UserPtrComparator::operator()().

Here is the caller graph for this function:



4.23.3.4 getPassword()

```
std::string User::getPassword ( ) const
```

Get the Password object.

Returns

std::string

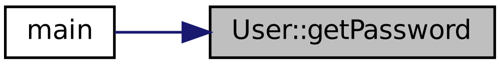
Definition at line 31 of file User.cpp.

```
31 { return password; }
```

References password.

Referenced by main().

Here is the caller graph for this function:



4.23.3.5 getPrivileges()

```
std::string User::getPrivileges ( ) const
```

Get the Privileges object.

Returns

std::string

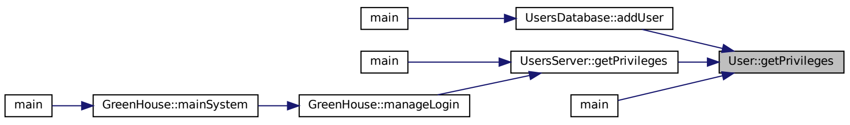
Definition at line 33 of file User.cpp.

```
33 { return privileges; }
```

References privileges.

Referenced by UsersDatabase::addUser(), UsersServer::getPrivileges(), and main().

Here is the caller graph for this function:



4.23.3.6 operator<()

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

Operator < overload (this comparison is made by the privileges)

Parameters

user	

Returns

true

false

Definition at line 62 of file User.cpp.

```
62                                        {
63   return privileges > other.privileges;
64 }
```

References privileges.

4.23.3.7 operator==()

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

Operator == overload (this comparison is made by the nif)

Parameters

user	

Returns

true

false

Definition at line 73 of file User.cpp.

```
73 { return nif == other.nif; }
```

References nif.

4.23.3.8 operator>()

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

Operator > overload (this comparison is made by the privileges)

Parameters

user	

Returns

true

false

Definition at line 68 of file User.cpp.

```
                                        {
  return privileges < other.privileges;
};
```

References privileges.

4.23.3.9 printUser()

```
void User::printUser ( ) const
```

Print the user.

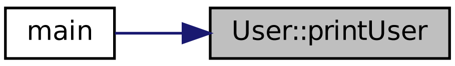
Definition at line 52 of file User.cpp.

```
                          {
  std::cout « "----------User------------" « std::endl;
  std::cout « "Name: " « name « std::endl;
  std::cout « "NIF: " « nif « std::endl;
  std::cout « "Password: " « password « std::endl;
  std::cout « "Privileges: " « privileges « std::endl;
  std::cout « "Email: " « email « std::endl;
  std::cout « "--------------------------" « std::endl;
}
```

References email, name, nif, password, and privileges.

Referenced by main().

Here is the caller graph for this function:



4.23.3.10 setEmail()

```
void User::setEmail (
              const std::string email )
```

Set the Email object.

Parameters

email	

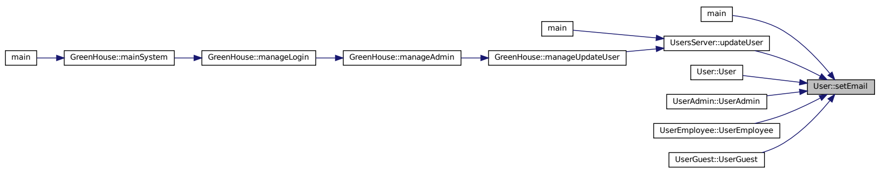
Definition at line 50 of file User.cpp.

```
50 { this->email = email; }
```

References email.

Referenced by main(), UsersServer::updateUser(), User(), UserAdmin::UserAdmin(), UserEmployee::User↩
Employee(), and UserGuest::UserGuest().

Here is the caller graph for this function:



4.23.3.11 setName()

```
void User::setName (
              const std::string name )
```

Set the Name object.

Parameters

name	

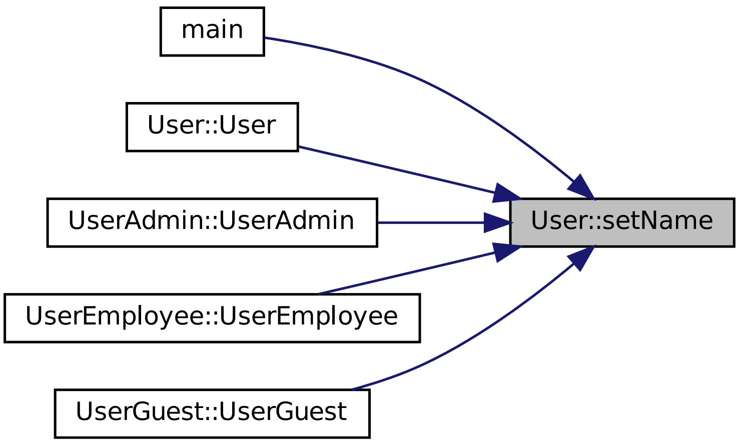
Definition at line 37 of file User.cpp.

```
37 { this->name = name; }
```

References name.

Referenced by main(), User(), UserAdmin::UserAdmin(), UserEmployee::UserEmployee(), and UserGuest::User↩
Guest().

Here is the caller graph for this function:



4.23.3.12 setNif()

```
void User::setNif (
             const std::string nif )
```

Set the Nif object.

Parameters

nif	

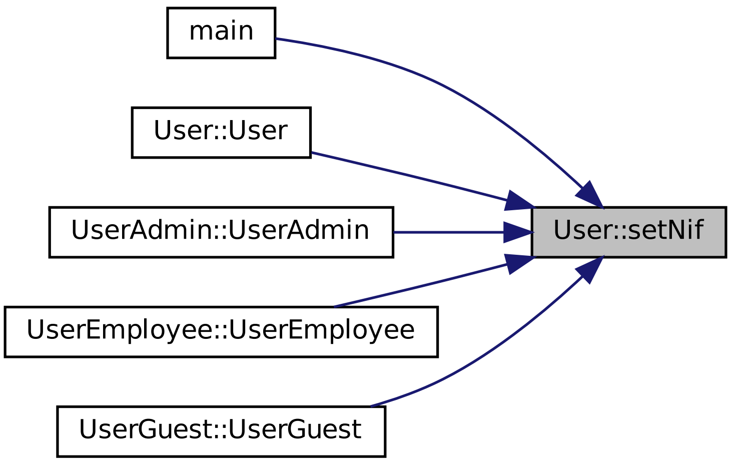
Definition at line 39 of file User.cpp.

```
39 { this->nif = nif; }
```

References nif.

Referenced by main(), User(), UserAdmin::UserAdmin(), UserEmployee::UserEmployee(), and UserGuest::User↩ Guest().

Here is the caller graph for this function:



4.23.3.13 setPassword()

```
void User::setPassword (
             const std::string password )
```

Set the Password object.

Parameters

password	

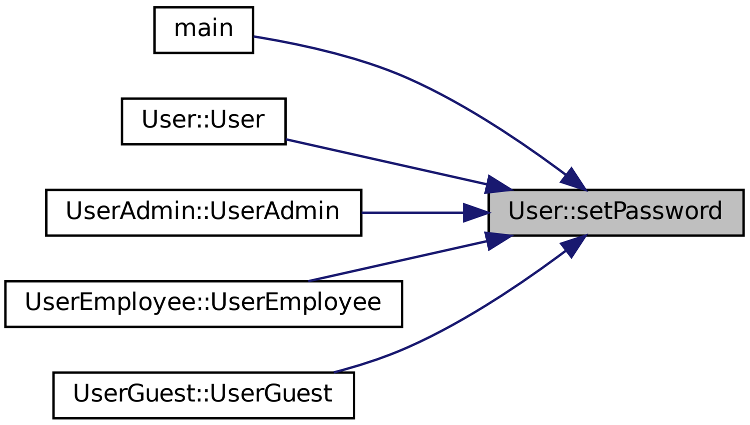
Definition at line 41 of file User.cpp.

```
41 { this->password = password; }
```

References password.

Referenced by main(), User(), UserAdmin::UserAdmin(), UserEmployee::UserEmployee(), and UserGuest::User↩
Guest().

Here is the caller graph for this function:



4.23.3.14 setPrivileges()

```
void User::setPrivileges (
            const std::string privileges )  [virtual]
```

Set the Privileges object.

Parameters

privileges	

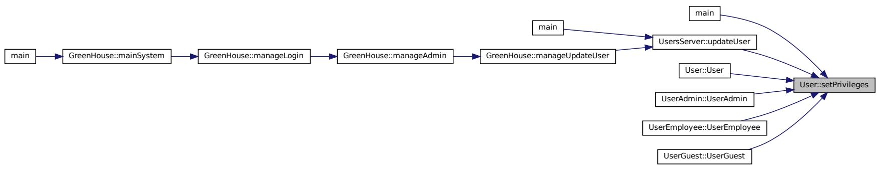
Definition at line 43 of file User.cpp.

```
43                                  {
44   for (std::string::size_type i = 0; i < privileges.length(); i++) {
45     privileges[i] = toupper(privileges[i]);
46   }
47   this->privileges = privileges;
48 }
```

References privileges.

Referenced by main(), UsersServer::updateUser(), User(), UserAdmin::UserAdmin(), UserEmployee::User←Employee(), and UserGuest::UserGuest().

Here is the caller graph for this function:



4.23.4 Friends And Related Function Documentation

4.23.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const User & user ) [friend]
```

Operator << overload.

Parameters

os	
user	

Returns

std::ostream&

Definition at line 76 of file User.cpp.

```
                                                       {
    os « user.getName() « " " « user.getNif() « " " « user.getPassword()
       « " " « user.getPrivileges() « " " « user.getEmail() « std::endl;
    return os;
}
```

4.23.4.2 operator>>

```
std::istream& operator>> (
            std::istream & is,
            User & user ) [friend]
```

Operator >> overload.

Parameters

is	
user	

Returns

std::istream&

Definition at line 83 of file User.cpp.

```
                                                 {
    std::string privilege;
    is » user.name » user.nif » user.password » privilege » user.email;
    user.setPrivileges(privilege);
    return is;
}
```

4.23.5 Member Data Documentation

4.23.5.1 email

```
std::string User::email  [private]
```

This is the email of the user.

Definition at line 184 of file User.h.

Referenced by getEmail(), printUser(), setEmail(), User(), UserAdmin::UserAdmin(), UserEmployee::User↩
Employee(), and UserGuest::UserGuest().

4.23.5.2 name

```
std::string User::name  [private]
```

This is the name of the user.

Definition at line 164 of file User.h.

Referenced by getName(), printUser(), setName(), User(), UserAdmin::UserAdmin(), UserEmployee::User↩
Employee(), and UserGuest::UserGuest().

4.23.5.3 nif

```
std::string User::nif  [private]
```

This is the nif of the user.

Definition at line 169 of file User.h.

Referenced by getNif(), operator==(), printUser(), setNif(), User(), UserAdmin::UserAdmin(), UserEmployee::User↩
Employee(), and UserGuest::UserGuest().

4.23.5.4 password

```
std::string User::password  [private]
```

This is the password of the user.

Definition at line 174 of file User.h.

Referenced by getPassword(), printUser(), setPassword(), User(), UserAdmin::UserAdmin(), UserEmployee::↩
UserEmployee(), and UserGuest::UserGuest().

4.23.5.5 privileges

```
std::string User::privileges  [private]
```

This is the privileges of the user.

Definition at line 179 of file User.h.

Referenced by getPrivileges(), operator<(), operator>(), printUser(), setPrivileges(), and User().

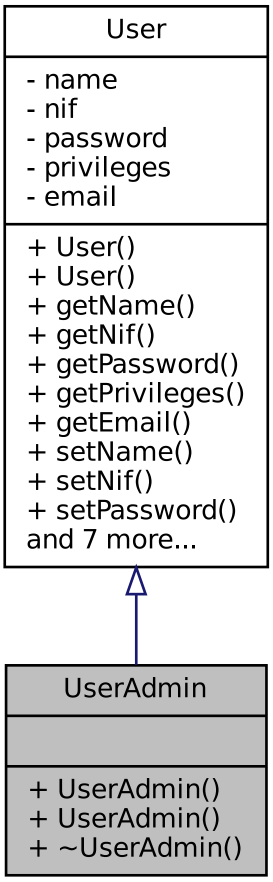
The documentation for this class was generated from the following files:

- src/User.h
- src/User.cpp

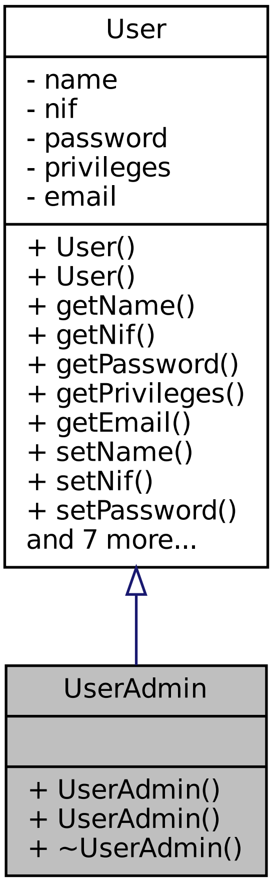
4.24 UserAdmin Class Reference

```
#include <UserAdmin.h>
```

Inheritance diagram for UserAdmin:



Collaboration diagram for UserAdmin:



Public Member Functions

- UserAdmin ()

  Construct a new User Admin object.
- UserAdmin (const std::string name, const std::string nif, std::string password, std::string email)

  Construct a new User Admin object.
- virtual ∼UserAdmin ()

  Destroy the User Admin object.

4.24.1 Detailed Description

Definition at line 16 of file UserAdmin.h.

4.24.2 Constructor & Destructor Documentation

4.24.2.1 UserAdmin() [1/2]

```
UserAdmin::UserAdmin ( )
```

Construct a new User Admin object.

Creates a new UserAdmin object with the default values (name, nif, password, email).

Returns

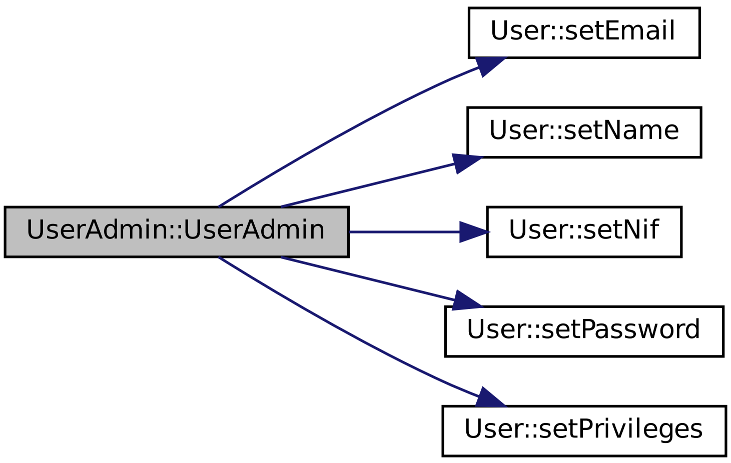
UserAdmin object

Definition at line 3 of file UserAdmin.cpp.

```
                  {
    // Utilizar los setters para asignar valores a los atributos
    setName("");
    setNif("");
    setPassword("");
    setEmail("");
    setPrivileges("ADMIN");
}
```

References User::setEmail(), User::setName(), User::setNif(), User::setPassword(), and User::setPrivileges().

Here is the call graph for this function:



4.24.2.2 UserAdmin() [2/2]

```
UserAdmin::UserAdmin (
            const std::string name,
            const std::string nif,
            std::string password,
            std::string email )  [explicit]
```

Construct a new User Admin object.

Creates a new UserAdmin object with the values passed as parameters.

Parameters

name	of the user
nif	of the user
password	of the user
email	of the user

Returns

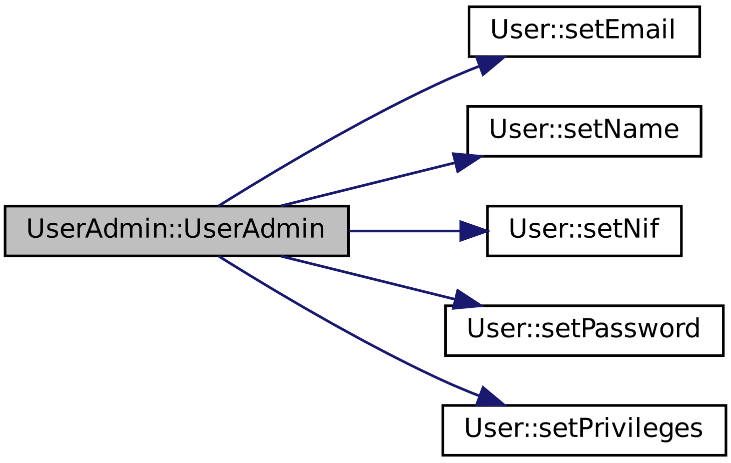
UserAdmin object

Definition at line 12 of file UserAdmin.cpp.

```
                                                            {
    // Utilizar los setters para asignar valores a los atributos
    setName(name);
    setNif(nif);
    setPassword(password);
    setEmail(email);
    setPrivileges("ADMIN");
}
```

References User::email, User::name, User::nif, User::password, User::setEmail(), User::setName(), User::setNif(), User::setPassword(), and User::setPrivileges().

Here is the call graph for this function:



4.24.2.3 ∼UserAdmin()

```
UserAdmin::∼UserAdmin ( )  [virtual]
```

Destroy the User Admin object.

Definition at line 22 of file UserAdmin.cpp.

```
{}
```

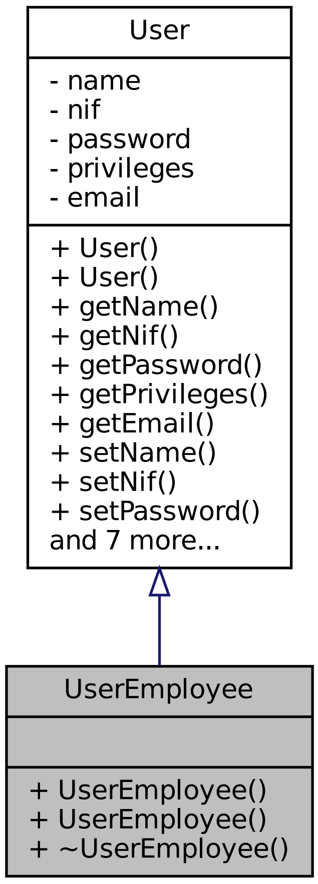
The documentation for this class was generated from the following files:

- src/UserAdmin.h
- src/UserAdmin.cpp

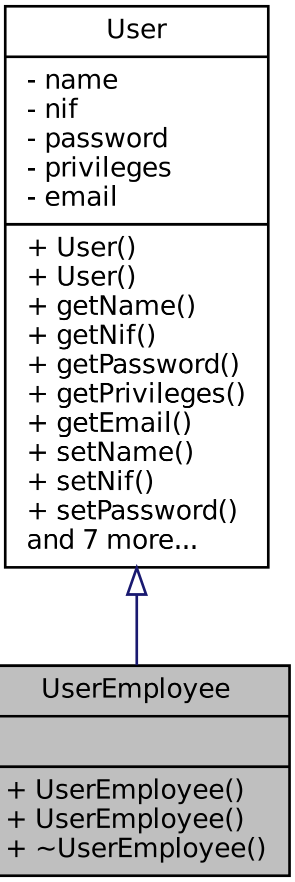
4.25 UserEmployee Class Reference

```
#include <UserEmployee.h>
```

Inheritance diagram for UserEmployee:



Collaboration diagram for UserEmployee:



Public Member Functions

- UserEmployee ()

    Construct a new User Employee object.
- UserEmployee (const std::string name, const std::string nif, std::string password, std::string email)

    Construct a new User Employee object.
- virtual ~UserEmployee ()

    Destroy the User Employee object.

4.25.1 Detailed Description

Definition at line 15 of file UserEmployee.h.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 UserEmployee() [1/2]

```
UserEmployee::UserEmployee ( )
```

Construct a new User Employee object.

Creates a new UserEmployee object with the default values (name, nif, password, email).

Returns

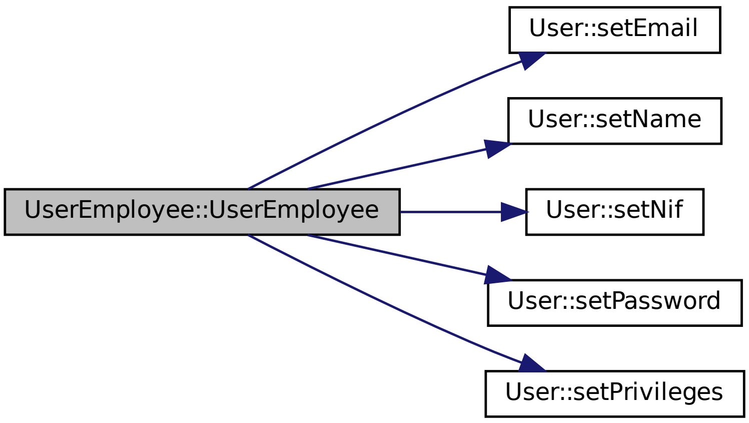
UserEmployee object

Definition at line 3 of file UserEmployee.cpp.

```
                             {
    // Utilizar los setters para asignar valores a los atributos
    setName("");
    setNif("");
    setPassword("");
    setEmail("");
    setPrivileges("EMPLOYEE");
}
```

References User::setEmail(), User::setName(), User::setNif(), User::setPassword(), and User::setPrivileges().

Here is the call graph for this function:



4.25.2.2 UserEmployee() [2/2]

```
UserEmployee::UserEmployee (
            const std::string name,
            const std::string nif,
            std::string password,
            std::string email ) [explicit]
```

Construct a new User Employee object.

Creates a new UserEmployee object with the values passed as parameters.

Parameters

name	of the user
nif	of the user
password	of the user
email	of the user

Returns

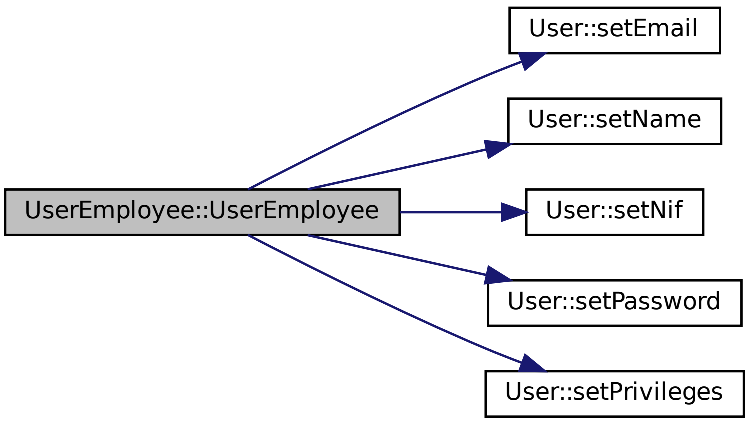
UserEmployee object

Definition at line 12 of file UserEmployee.cpp.

```
                                                                {
    // Utilizar los setters para asignar valores a los atributos
    setName(name);
    setNif(nif);
    setPassword(password);
    setEmail(email);
    setPrivileges("EMPLOYEE");
}
```

References User::email, User::name, User::nif, User::password, User::setEmail(), User::setName(), User::setNif(), User::setPassword(), and User::setPrivileges().

Here is the call graph for this function:



4.25.2.3 ∼UserEmployee()

```
UserEmployee::∼UserEmployee ( )  [virtual]
```

Destroy the User Employee object.

Definition at line 22 of file UserEmployee.cpp.

```
{}
```

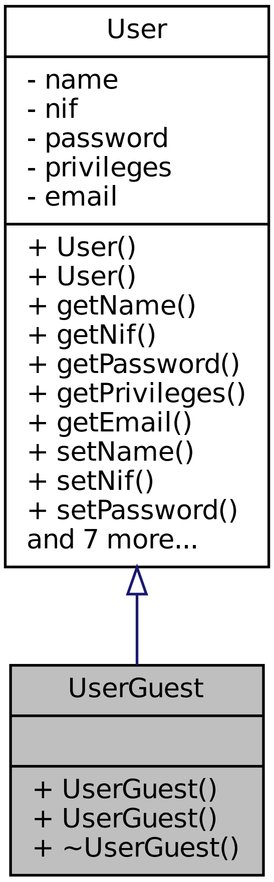
The documentation for this class was generated from the following files:

- src/UserEmployee.h
- src/UserEmployee.cpp

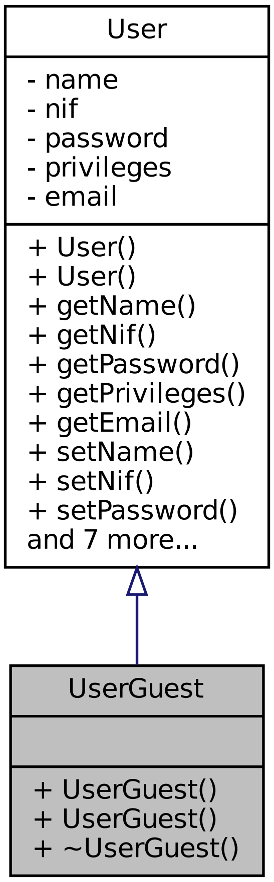
4.26 UserGuest Class Reference

```
#include <UserGuest.h>
```

Inheritance diagram for UserGuest:



Collaboration diagram for UserGuest:



Public Member Functions

- UserGuest ()

    Construct a new User Guest object.
- UserGuest (const std::string name, const std::string nif, std::string password, std::string email)

    Construct a new User Guest object.
- virtual ~UserGuest ()

    Destroy the User Guest object.

4.26.1 Detailed Description

Definition at line 16 of file UserGuest.h.

4.26.2 Constructor & Destructor Documentation

4.26.2.1 UserGuest() [1/2]

```
UserGuest::UserGuest ( )
```

Construct a new User Guest object.

Creates a new UserGuest object with the default values (name, nif, password, email).

Returns

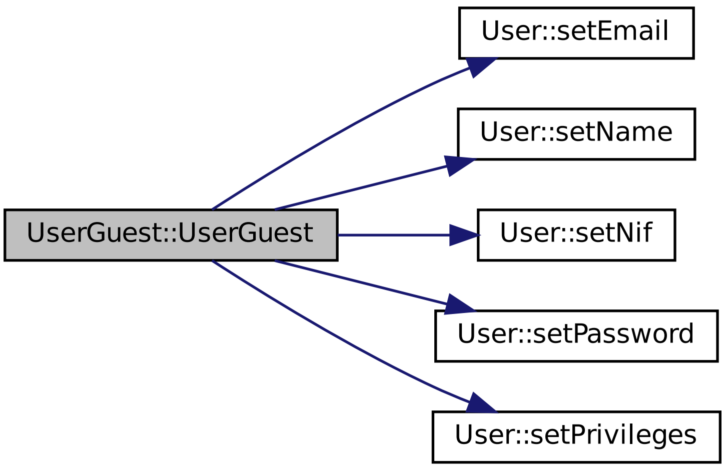
UserGuest object

Definition at line 3 of file UserGuest.cpp.

```
                       {
    // Utilizar los setters para asignar valores a los atributos
    setName("");
    setNif("");
    setPassword("");
    setEmail("");
    setPrivileges("GUEST");
}
```

References User::setEmail(), User::setName(), User::setNif(), User::setPassword(), and User::setPrivileges().

Here is the call graph for this function:



4.26.2.2 UserGuest() [2/2]

```
UserGuest::UserGuest (
              const std::string name,
              const std::string nif,
              std::string password,
              std::string email ) [explicit]
```

Construct a new User Guest object.

Creates a new UserGuest object with the values passed as parameters.

Parameters

name	of the user
nif	of the user
password	of the user
email	of the user

Returns

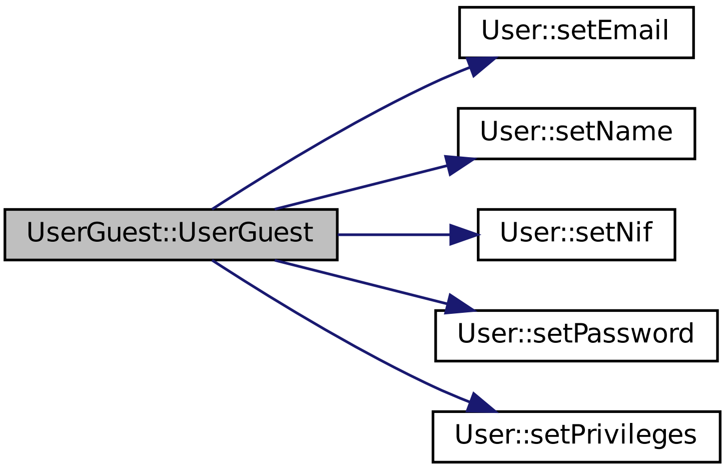
UserGuest object

Definition at line 12 of file UserGuest.cpp.

```
                                                           {
    // Utilizar los setters para asignar valores a los atributos
    setName(name);
    setNif(nif);
    setPassword(password);
    setEmail(email);
    setPrivileges("GUEST");
}
```

References User::email, User::name, User::nif, User::password, User::setEmail(), User::setName(), User::setNif(), User::setPassword(), and User::setPrivileges().

Here is the call graph for this function:



4.26.2.3 ∼UserGuest()

```
UserGuest::∼UserGuest ( )  [virtual]
```

Destroy the User Guest object.

Definition at line 22 of file UserGuest.cpp.

```
{}
```

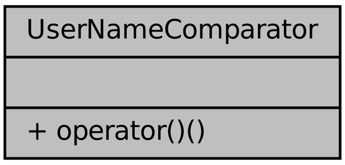
The documentation for this class was generated from the following files:

- src/UserGuest.h
- src/UserGuest.cpp

4.27 UserNameComparator Class Reference

```
#include <UsersDatabase.h>
```

Collaboration diagram for UserNameComparator:



Public Member Functions

- bool operator() (const User ∗lhs, const User ∗rhs) const

4.27.1 Detailed Description

Definition at line 32 of file UsersDatabase.h.

4.27.2 Member Function Documentation

4.27.2.1 operator()()

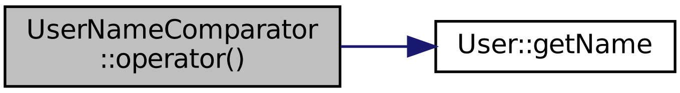
```
bool UserNameComparator::operator() (
            const User ∗ lhs,
            const User ∗ rhs ) const  [inline]
```

Definition at line 34 of file UsersDatabase.h.

```
                                                        {
      return lhs->getName() < rhs->getName();
   }
```

References User::getName().

Here is the call graph for this function:



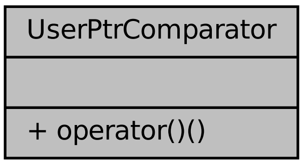
The documentation for this class was generated from the following file:

- src/UsersDatabase.h

4.28 UserPtrComparator Class Reference

```
#include <UsersDatabase.h>
```

Collaboration diagram for UserPtrComparator:



Public Member Functions

- bool operator() (const User ∗us1, const User ∗us2) const

4.28.1 Detailed Description

Definition at line 23 of file UsersDatabase.h.

4.28.2 Member Function Documentation

4.28.2.1 operator()()

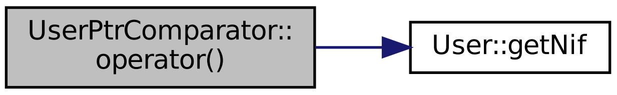
```
bool UserPtrComparator::operator() (
             const User * us1,
             const User * us2 ) const  [inline]
```

Definition at line 25 of file UsersDatabase.h.

```
                                                   {
    // Comparar dnis de usuarios si tienen el mismo dni no se puede añadir el
    // usuario
    return us1->getNif() < us2->getNif();
  }
```

References User::getNif().

Here is the call graph for this function:



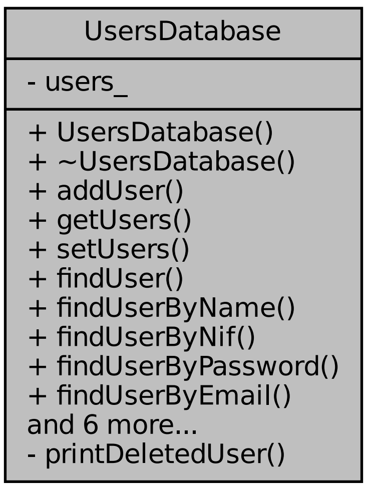
The documentation for this class was generated from the following file:

- src/UsersDatabase.h

4.29 UsersDatabase Class Reference

```
#include <UsersDatabase.h>
```

Collaboration diagram for UsersDatabase:



Public Member Functions

- UsersDatabase ()

    Construct a new Users Database object Creates a new UsersDatabase object that contains the pointers to users.
- ∼UsersDatabase ()

Destroy the Users Database object.
- void addUser (const User ∗user)

  Add a user to the set of users.
- std::set< const User ∗, UserPtrComparator > getUsers () const

  Get the Users object.
- void setUsers (const std::set< const User ∗, UserPtrComparator > &users)

  Set the Users object.
- User ∗ findUser (const User &user) const

  Find a user in the set of users.
- User ∗ findUserByName (const std::string name) const

  Find a user in the set of users with the name.
- User ∗ findUserByNif (const std::string nif) const

  Fiend a user in the set of users with the NIF.
- User ∗ findUserByPassword (const std::string password) const

  Find a user in the set of users with the password.
- User ∗ findUserByEmail (const std::string email) const

  Find a user in the set of users with the email.
- void deleteUser (const User &user)

  Delete a user from the set of users.
- void deleteUserByName (const std::string name)

  Delete a user from the set of users with the name.
- void deleteUserByNif (const std::string nif)

  Delete a user from the set of users with the NIF.
- void deleteUserByEmail (const std::string email)

  Delete a user from the set of users with the email.
- bool isValidPrivileges (const std::string privileges) const

  This method checks if the privileges are valid.
- void printUsers () const

  Print all the users.

Private Member Functions

- void printDeletedUser (const User ∗user) const

  This method prints the user that has been deleted.

Private Attributes

- std::set< const User ∗, UserPtrComparator > users_

  This is the set of pointers to users.

4.29.1 Detailed Description

Definition at line 40 of file UsersDatabase.h.

4.29.2 Constructor & Destructor Documentation

4.29.2.1 UsersDatabase()

```
UsersDatabase::UsersDatabase ( )
```

Construct a new Users Database object Creates a new UsersDatabase object that contains the pointers to users.

Returns

UsersDatabase object

Definition at line 13 of file UsersDatabase.cpp.

```
{}
```

4.29.2.2 ∼UsersDatabase()

```
UsersDatabase::∼UsersDatabase ( )
```

Destroy the Users Database object.

Definition at line 15 of file UsersDatabase.cpp.

```
                          {
  // Liberar la memoria de los usuarios
  for (auto user : users_) {
    delete user;
  }
}
```

References users_.

4.29.3 Member Function Documentation

4.29.3.1 addUser()

```
void UsersDatabase::addUser (
            const User * user )
```

Add a user to the set of users.

Parameters

user	

Definition at line 28 of file UsersDatabase.cpp.

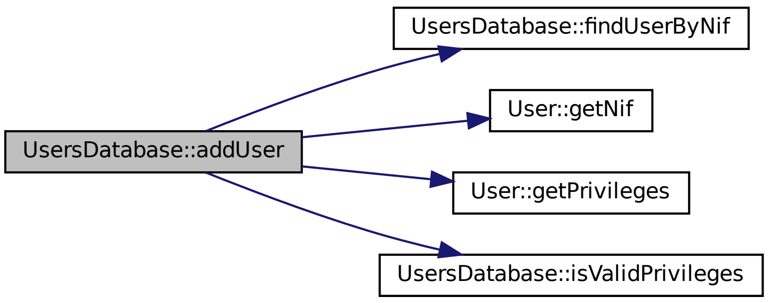
```
                                        {
  // Intento añadir un usuario al set de usuarios si no existe, si existe no lo
  // añado y llamo a su destructor, tambien tiene que tener unos privilegios
  // validos

  if (findUserByNif(user->getNif()) == nullptr &&
      (isValidPrivileges(user->getPrivileges()))) {
```

```
    users_.insert(user);
  } else {
    if (!isValidPrivileges(user->getPrivileges())) {
      std::cout « "Privileges are not valid (ADMIN/EMPLOYEE/GUEST)"
                « std::endl;
    } else {
      std::cout « "User already exists" « std::endl;
    }
    delete user;
  }
}
```

References findUserByNif(), User::getNif(), User::getPrivileges(), isValidPrivileges(), and users_.

Referenced by main().

Here is the call graph for this function:


UsersDatabase::findUserByNif
User::getNif
UsersDatabase::addUser
User::getPrivileges
UsersDatabase::isValidPrivileges

Here is the caller graph for this function:

main
UsersDatabase::addUser

4.29.3.2 deleteUser()

```
void UsersDatabase::deleteUser (
             const User & user )
```

Delete a user from the set of users.

Parameters

user	

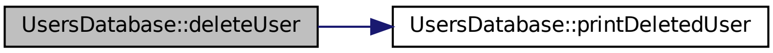
Definition at line 148 of file UsersDatabase.cpp.

```
                                                {
  for (std::set<const User *>::iterator it = users_.begin(); it != users_.end();
       it++) {
    // Si encuentro el usuario lo borro y también el puntero con el destructor
    // de la clase User
    if (*(*it) == user) {
      printDeletedUser(*it);
      delete *it; // Llama al destructor de User y libera la memoria
      users_.erase(it);
      return;
    }
  }

  std::cout « "User not found" « std::endl;
  std::cout « std::endl;
}
```

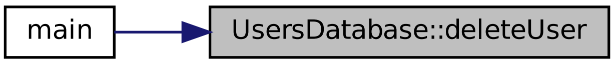
References printDeletedUser(), and users_.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.29.3.3 deleteUserByEmail()

```
void UsersDatabase::deleteUserByEmail (
            const std::string email )
```

Delete a user from the set of users with the email.

Parameters

email	

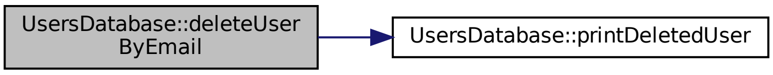
Definition at line 195 of file UsersDatabase.cpp.

```
                                                           {
  for (std::set<const User *>::iterator it = users_.begin(); it != users_.end();
       it++) {
    if ((*it)->getEmail() == email) {
      printDeletedUser(*it);
      delete *it; // Llama al destructor de User y libera la memoria
      users_.erase(it);
      return;
    }
  }

  std::cout « "User not found" « std::endl;
  std::cout « std::endl;
}
```

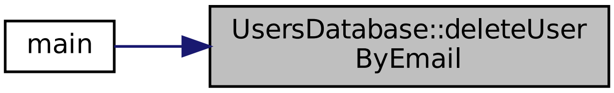
References printDeletedUser(), and users_.

Referenced by main().

Here is the call graph for this function:


UsersDatabase::deleteUser ByEmail
UsersDatabase::printDeletedUser

Here is the caller graph for this function:



4.29.3.4 deleteUserByName()

```
void UsersDatabase::deleteUserByName (
               const std::string name )
```

Delete a user from the set of users with the name.

Parameters

name	

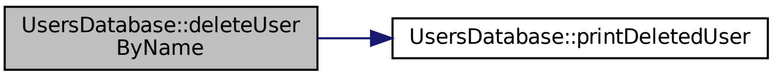
Definition at line 165 of file UsersDatabase.cpp.

```
                                                          {
  for (std::set<const User *>::iterator it = users_.begin(); it != users_.end();
       it++) {
    // Si esta el usuario lo borro y tambien borro el puntero
    if ((*it)->getName() == name) {
      printDeletedUser(*it);
      delete *it; // Llama al destructor de User y libera la memoria
      users_.erase(it);
      return;
    }
  }
  std::cout « "User not found" « std::endl;
  std::cout « std::endl;
}
```

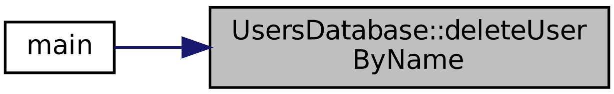
References printDeletedUser(), and users_.

Referenced by main().

Here is the call graph for this function:


UsersDatabase::deleteUser ByName
UsersDatabase::printDeletedUser

Here is the caller graph for this function:



4.29.3.5 deleteUserByNif()

```
void UsersDatabase::deleteUserByNif (
              const std::string nif )
```

Delete a user from the set of users with the NIF.

Parameters

nif	

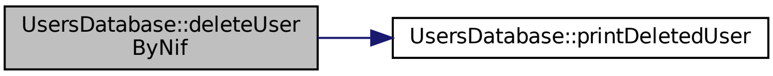
Definition at line 180 of file UsersDatabase.cpp.

```
                                                           {
  for (std::set<const User *>::iterator it = users_.begin(); it != users_.end();
       it++) {
    if ((*it)->getNif() == nif) {
      printDeletedUser(*it);
      delete *it; // Llama al destructor de User y libera la memoria
      users_.erase(it);
      return;
    }
  }

  std::cout « "User not found" « std::endl;
  std::cout « std::endl;
}
```

References printDeletedUser(), and users_.

Referenced by main().

Here is the call graph for this function:


UsersDatabase::deleteUser ByNif
UsersDatabase::printDeletedUser

Here is the caller graph for this function:

main
UsersDatabase::deleteUser ByNif

4.29.3.6 findUser()

```
User * UsersDatabase::findUser (
             const User & user ) const
```

Find a user in the set of users.

Parameters

user	

Returns

User∗

Definition at line 78 of file UsersDatabase.cpp.

```
                                                    {
    // Si el usuario existe devuelvo el usuario, si no existe devuelvo un usuario
    // con todos los atributos vacios
    for (std::set<const User *, UserPtrComparator>::const_iterator it =
             users_.begin();
         it != users_.end(); it++) {
      if (*(*it) == user) {
        return const_cast<User *>(*it);
      }
    }

    return nullptr;
}
```

References users_.

Referenced by main().

Here is the caller graph for this function:

main
UsersDatabase::findUser

4.29.3.7 findUserByEmail()

```
User * UsersDatabase::findUserByEmail (
              const std::string email ) const
```

Find a user in the set of users with the email.

Parameters

email	

Returns

User∗

Definition at line 134 of file UsersDatabase.cpp.

```
                                                               {
  // Si el usuario existe devuelvo el usuario, si no existe devuelvo un usuario
  // con todos los atributos vacios
  for (std::set<const User *, UserPtrComparator>::const_iterator it =
           users_.begin();
       it != users_.end(); it++) {
    if ((*it)->getEmail() == email) {
      return const_cast<User *>(*it);
    }
  }

  return nullptr;
}
```

References users_.

Referenced by main().

Here is the caller graph for this function:

main
UsersDatabase::findUserByEmail

4.29.3.8 findUserByName()

```
User * UsersDatabase::findUserByName (
             const std::string name ) const
```

Find a user in the set of users with the name.

Parameters

name	

Returns

User*

Definition at line 92 of file UsersDatabase.cpp.

```
                                                              {
  // Si el usuario existe devuelvo el usuario, si no existe devuelvo un usuario
  // con todos los atributos vacios
  for (std::set<const User *, UserPtrComparator>::const_iterator it =
           users_.begin();
       it != users_.end(); it++) {
    if ((*it)->getName() == name) {
      return const_cast<User *>(*it);
    }
  }

  return nullptr;
}
```

References users_.

Referenced by main().

Here is the caller graph for this function:

main
UsersDatabase::findUserByName

4.29.3.9 findUserByNif()

```
User * UsersDatabase::findUserByNif (
             const std::string nif ) const
```

Fiend a user in the set of users with the NIF.

Parameters

nif	

Returns

User∗

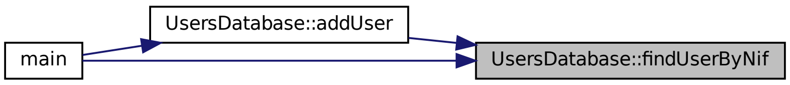
Definition at line 106 of file UsersDatabase.cpp.

```
                                                            {
  // Si el usuario existe devuelvo el usuario, si no existe devuelvo un usuario
  // con todos los atributos vacios
  for (std::set<const User *, UserPtrComparator>::const_iterator it =
           users_.begin();
       it != users_.end(); it++) {
    if ((*it)->getNif() == nif) {
      return const_cast<User *>(*it);
    }
  }

  return nullptr;
}
```

References users_.

Referenced by addUser(), and main().

Here is the caller graph for this function:



4.29.3.10 findUserByPassword()

```
User * UsersDatabase::findUserByPassword (
             const std::string password ) const
```

Find a user in the set of users with the password.

Parameters

password	

Returns

User∗

Definition at line 120 of file UsersDatabase.cpp.

```
                                                              {
  // Si el usuario existe devuelvo el usuario, si no existe devuelvo un usuario
  // con todos los atributos vacios
  for (std::set<const User *, UserPtrComparator>::const_iterator it =
           users_.begin();
       it != users_.end(); it++) {
    if ((*it)->getPassword() == password) {
      return const_cast<User *>(*it);
    }
  }

  return nullptr;
}
```

References users_.

4.29.3.11 getUsers()

```
std::set< const User *, UserPtrComparator > UsersDatabase::getUsers ( ) const
```

Get the Users object.

Returns

std::set<const User ∗, UserPtrComparator>

Definition at line 47 of file UsersDatabase.cpp.
```
{
  return users_;
}
```

References users_.

Referenced by main().

Here is the caller graph for this function:

main
UsersDatabase::getUsers

4.29.3.12 isValidPrivileges()

```
bool UsersDatabase::isValidPrivileges (
            const std::string privileges ) const
```

This method checks if the privileges are valid.

Parameters

privileges	

Returns

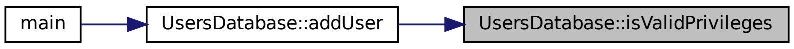
true

false

Definition at line 22 of file UsersDatabase.cpp.
```
{
  // Compruebo si los privilegios son validos
  return privileges == "ADMIN" || privileges == "EMPLOYEE" ||
         privileges == "GUEST";
}
```

Referenced by addUser().

Here is the caller graph for this function:



4.29.3.13 printDeletedUser()

```
void UsersDatabase::printDeletedUser (
             const User * user ) const  [private]
```

This method prints the user that has been deleted.

This method prints the user that has been deleted, is private method becouse to print a delete user first you have to delete one. So when you delete a user, this method print the user that has been deleted.

Parameters

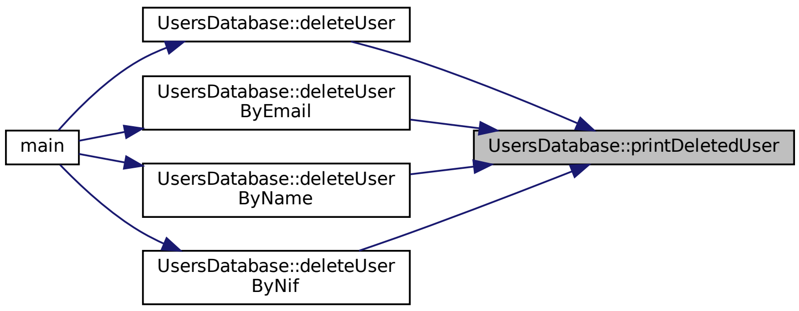
user	

Definition at line 221 of file UsersDatabase.cpp.

```
                                                         {
    std::cout « "User deleted: " « (*user).getName() « "-" « (*user).getNif()
              « std::endl;
    std::cout « std::endl;
}
```

Referenced by deleteUser(), deleteUserByEmail(), deleteUserByName(), and deleteUserByNif().

Here is the caller graph for this function:


UsersDatabase::deleteUser
UsersDatabase::deleteUser ByEmail
main
UsersDatabase::printDeletedUser
UsersDatabase::deleteUser ByName
UsersDatabase::deleteUser ByNif

4.29.3.14 printUsers()

```
void UsersDatabase::printUsers ( ) const
```

Print all the users.

Definition at line 210 of file UsersDatabase.cpp.

```
                                        {
  std::cout « "*** LIST OF USERS ***" « std::endl;
  // Imprimir todos los usuarios
  for (const auto &user : users_) {
    (*user).printUser();
    std::cout « std::endl;
  }

  std::cout « "*** END OF LIST ***" « std::endl;
}
```

References users_.

Referenced by main().

Here is the caller graph for this function:

main
UsersDatabase::printUsers

4.29.3.15 setUsers()

```
void UsersDatabase::setUsers (
              const std::set< const User *, UserPtrComparator > & users )
```

Set the Users object.

Parameters

users	

Definition at line 51 of file UsersDatabase.cpp.

```
                                                         {
  // El set users debe de convertir los punteros de usuarios a Objetos de
  // usuarios y luego añadirlos al set de usuarios Bucle para convertirlos en
  // objetos usuario
  for (auto user : users) {
    try {
      if (user->getPrivileges() == "ADMIN") {
        users_.insert(new UserAdmin(*static_cast<const UserAdmin *>(user)));
      } else if (user->getPrivileges() == "EMPLOYEE") {
        users_.insert(
            new UserEmployee(*static_cast<const UserEmployee *>(user)));
      } else if (user->getPrivileges() == "GUEST") {
        users_.insert(new UserGuest(*static_cast<const UserGuest *>(user)));
      } else {
        std::cerr « "Unknown user privilege: " « user->getPrivileges()
                  « '\n';
      }
    } catch (std::bad_alloc &ba) {
      std::cerr « "bad_alloc caught: " « ba.what() « '\n';
      // Borrar el usuario que no se ha podido añadir
      delete user;
      throw;
    }
  }
}
```

References users_.

Referenced by main().

Here is the caller graph for this function:

main
UsersDatabase::setUsers

4.29.4 Member Data Documentation

4.29.4.1 users_

`std::set<const User *, UserPtrComparator> UsersDatabase::users_ [private]`

This is the set of pointers to users.

This is the atributte that contains the pointers to users.

Definition at line 175 of file UsersDatabase.h.

Referenced by addUser(), deleteUser(), deleteUserByEmail(), deleteUserByName(), deleteUserByNif(), find←User(), findUserByEmail(), findUserByName(), findUserByNif(), findUserByPassword(), getUsers(), printUsers(), setUsers(), and ∼UsersDatabase().

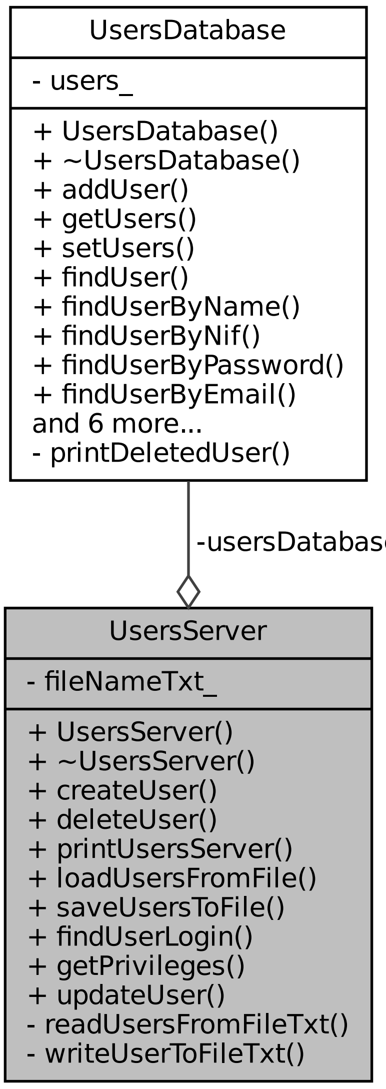
The documentation for this class was generated from the following files:

- src/UsersDatabase.h
- src/UsersDatabase.cpp

4.30 UsersServer Class Reference

```
#include <UsersServer.h>
```

Collaboration diagram for UsersServer:



Public Member Functions

- UsersServer ()

    Construct a new Users Server object Creates a new UsersServer object that contains the UsersDatabase object.
- ~UsersServer ()

    Destroy the Users Server object.

- void createUser (const std::string name, const std::string nif, const std::string password, const std::string privileges, const std::string email)

  Create a User object.
- void deleteUser (const std::string nif)

  Delete a User object.
- void printUsersServer () const

  Print all Users Print all users in the UsersDatabase object.
- void loadUsersFromFile ()

  Load Users from File Load users from a file.
- void saveUsersToFile ()

  Save Users to File Save users to a file.
- bool findUserLogin (std::string name, std::string password, std::string nif)

  Find User Login.
- std::string getPrivileges (std::string nif)

  Get Privileges.
- void updateUser (const std::string name, const std::string nif, const std::string password, const std::string privileges, const std::string email)

  Update User.

Private Member Functions

- void readUsersFromFileTxt ()

  This method reads the users from a file txt.
- void writeUserToFileTxt ()

  This method writes the users to a file txt.

Private Attributes

- UsersDatabase usersDatabase_

  This is the UsersDatabase object.
- std::string fileNameTxt_ = "users.txt"

  This is the name of the file.

4.30.1 Detailed Description

Definition at line 19 of file UsersServer.h.

4.30.2 Constructor & Destructor Documentation

4.30.2.1 UsersServer()

```
UsersServer::UsersServer ( )
```

Construct a new Users Server object Creates a new UsersServer object that contains the UsersDatabase object.

Returns

UsersServer object

Definition at line 17 of file UsersServer.cpp.

```
                             {
    // Creamos tres usuarios por defecto
    createUser("admin", "12345678X", "admin", "ADMIN", "admin@example.com");
    createUser("employee", "12345678Y", "employee", "EMPLOYEE",
               "employee@example.com");
    createUser("guest", "12345678Z", "guest", "GUEST", "guest@example.com");
}
```

4.30.2.2 ∼UsersServer()

```
UsersServer::∼UsersServer ( )
```

Destroy the Users Server object.

Definition at line 25 of file UsersServer.cpp.

```
                              {
    // Esto destruira el set de punteros de la base de datos
}
```

4.30.3 Member Function Documentation

4.30.3.1 createUser()

```
void UsersServer::createUser (
            const std::string name,
            const std::string nif,
            const std::string password,
            const std::string privileges,
            const std::string email )
```

Create a User object.

Parameters

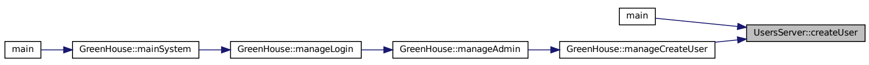
name	
nif	
password	
privileges	
email	

Definition at line 29 of file UsersServer.cpp.

```
32                                                        {
33   // Debe de crear el usuario y añadir el puntero al set de usuarios
34   User *user = new User(name, nif, password, privileges, email);
35   usersDatabase_.addUser(user);
36 }
```

Referenced by main(), and GreenHouse::manageCreateUser().

Here is the caller graph for this function:



4.30.3.2 deleteUser()

```
void UsersServer::deleteUser (
              const std::string nif )
```

Delete a User object.

Parameters

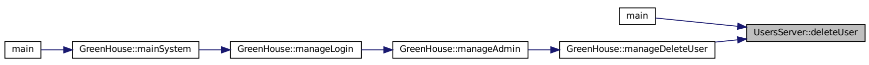
nif	

Definition at line 38 of file UsersServer.cpp.

```
38                                                  {
39   usersDatabase_.deleteUserByNif(nif);
40 }
```

Referenced by main(), and GreenHouse::manageDeleteUser().

Here is the caller graph for this function:



4.30.3.3 findUserLogin()

```
bool UsersServer::findUserLogin (
              std::string name,
              std::string password,
              std::string nif )
```

Find User Login.

Parameters

name	
password	
nif	

Returns

true

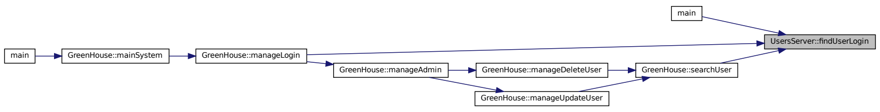
false

Definition at line 51 of file UsersServer.cpp.

```
                                                {
  // Debemos de buscar encontrar a un usuario por su nombre, contraseña y nif
  // Si tienen el mismo puntero devolver true, si no false
  if (usersDatabase_.findUserByNif(nif) != nullptr) {
    if (usersDatabase_.findUserByNif(nif)->getName() == name &&
        usersDatabase_.findUserByNif(nif)->getPassword() == password) {
      return true;
    }
  }

  return false;
}
```

Referenced by main(), GreenHouse::manageLogin(), and GreenHouse::searchUser().

Here is the caller graph for this function:



4.30.3.4 getPrivileges()

```
std::string UsersServer::getPrivileges (
              std::string nif )
```

Get Privileges.

Parameters

nif	

Returns

std::string the privileges

Definition at line 42 of file UsersServer.cpp.

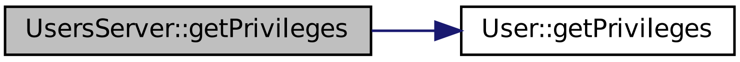
```
                                                    {
```

```
  User *user = usersDatabase_.findUserByNif(nif);
  if (user != nullptr) {
    return user->getPrivileges();
  } else {
    return "GUEST";
  }
}
```

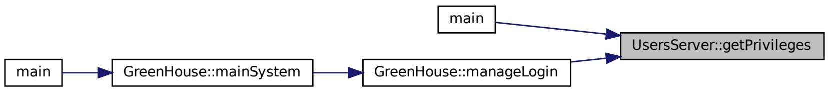
References User::getPrivileges().

Referenced by main(), and GreenHouse::manageLogin().

Here is the call graph for this function:



Here is the caller graph for this function:



4.30.3.5 loadUsersFromFile()

```
void UsersServer::loadUsersFromFile ( )
```

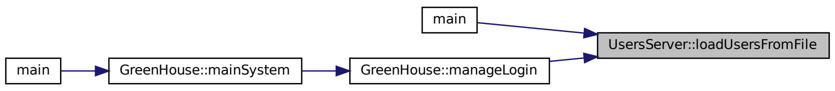
Load Users from File Load users from a file.

Definition at line 115 of file UsersServer.cpp.

```
                                       {
  // Cargo los usuarios del archivo
  // Siguiendo el formato de nombre nif password privilegios email

  std::cout « "Loading users from file " « fileNameTxt_ « "..." « std::endl;

  try {
    readUsersFromFileTxt();
  } catch (FileReadError &e) {
    std::cerr « e.what() « std::endl;

  } catch (FileCloseError &e) {
    std::cerr « e.what() « std::endl;

  } catch (FileOpenError &e) {
    std::cerr « e.what() « std::endl;

  } catch (FileNotFoundError &e) {
    std::cerr « e.what() « std::endl;
  }

  /*
  if (file.is_open()) {
    std::string name, nif, password, privileges, email;
    while (file » name » nif » password » privileges » email) {
      createUser(name, nif, password, privileges, email);
    }
    file.close();
  } else {
    std::cerr « "Error: Unable to open file for reading." « std::endl;
  }
  */
}
```

Referenced by main(), and GreenHouse::manageLogin().

Here is the caller graph for this function:



4.30.3.6 printUsersServer()

```
void UsersServer::printUsersServer ( ) const
```

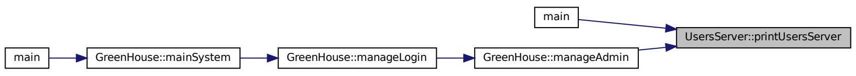
Print all Users Print all users in the UsersDatabase object.

Definition at line 80 of file UsersServer.cpp.

```
                                        {
  this->usersDatabase_.printUsers();
}
```

Referenced by main(), and GreenHouse::manageAdmin().

Here is the caller graph for this function:



4.30.3.7 readUsersFromFileTxt()

```
void UsersServer::readUsersFromFileTxt ( )  [private]
```

This method reads the users from a file txt.

Definition at line 84 of file UsersServer.cpp.

```
                                          {

  try {
    std::ifstream file(fileNameTxt_);

    if (!file.good()) {
      file.close();
      throw FileNotFoundError(fileNameTxt_);
    }
    if (!file.is_open()) {
      throw FileOpenError(fileNameTxt_);
    }

    std::string name, nif, password, privileges, email;
    while (file » name » nif » password » privileges » email) {
      std::cout « "Read user: " « name « " " « nif « " " « password « " "
                « privileges « " " « email « std::endl;
      createUser(name, nif, password, privileges, email);
    }

    file.close();

    if (file.is_open()) {
      throw FileCloseError(fileNameTxt_);
    }
  } catch (std::exception &e) {
    // std::cerr « e.what() « std::endl;
    throw;
  }
}
```

4.30.3.8 saveUsersToFile()

```
void UsersServer::saveUsersToFile ( )
```

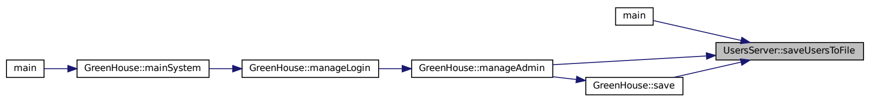
Save Users to File Save users to a file.

Definition at line 181 of file UsersServer.cpp.

```
                                   {
  // Guardo los usuarios en el archivo
  std::cout « "Saving users to file " « fileNameTxt_ « "..." « std::endl;

  try {
    writeUserToFileTxt();
  } catch (FileOpenError &e) {
    std::cerr « e.what() « std::endl;

  } catch (FileCloseError &e) {
    std::cerr « e.what() « std::endl;

  } catch (FileWriteError &e) {
    std::cerr « e.what() « std::endl;

  } catch (FileNotFoundError &e) {
    std::cerr « e.what() « std::endl;
  }
  /*
  std::cout « "Saving users to file..." « std::endl;
  std::ofstream file(fileNameTxt_);
  if (file.is_open()) {
    for (const User *user : usersDatabase_.getUsers()) {
      file « user->getName() « " " « user->getNif() « " "
           « user->getPassword() « " " « user->getPrivileges() « " "
           « user->getEmail() « std::endl;
      std::cout « "User saved: " « user->getName() « "-" « user->getNif()
                « std::endl;
    }
    file.close();
  } else {
    std::cerr « "Error: Unable to open file for writing." « std::endl;
  }
  */
}
```

Referenced by main(), GreenHouse::manageAdmin(), and GreenHouse::save().

Here is the caller graph for this function:



4.30.3.9 updateUser()

```
void UsersServer::updateUser (
              const std::string name,
              const std::string nif,
              const std::string password,
              const std::string privileges,
              const std::string email )
```

Update User.

Parameters

name	
nif	
password	
privileges	
email	

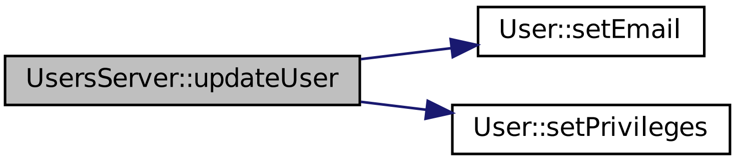
Definition at line 65 of file UsersServer.cpp.

```
                                                  {
  User *userToUpdate = usersDatabase_.findUserByNif(nif);
  // Si el usuario existe, y los privilegios que se quieren cambiar
  // isValidPrivileges entonces se cambian los privilegios y el email
  if (userToUpdate != nullptr && usersDatabase_.isValidPrivileges(privileges)) {
    userToUpdate->setPrivileges(privileges);
    userToUpdate->setEmail(email);
  } else {
    std::cout « "User dosen't change, maybe correct privileges" « std::endl;
  }
}
```

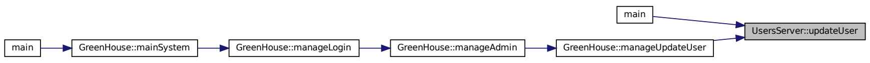
References User::setEmail(), and User::setPrivileges().

Referenced by main(), and GreenHouse::manageUpdateUser().

Here is the call graph for this function:



Here is the caller graph for this function:



4.30.3.10 writeUserToFileTxt()

```
void UsersServer::writeUserToFileTxt ( )  [private]
```

This method writes the users to a file txt.

Definition at line 149 of file UsersServer.cpp.

```
149                                        {
150   try {
151     std::ofstream file(fileNameTxt_);
152     if (!file.good()) {
153       file.close();
154       throw FileNotFoundError(fileNameTxt_);
155     }
156     if (!file.is_open()) {
157       throw FileOpenError(fileNameTxt_);
158     }
159 
160     // Write users to file
161     for (const User *user : usersDatabase_.getUsers()) {
162       file « user->getName() « " " « user->getNif() « " "
163            « user->getPassword() « " " « user->getPrivileges() « " "
164            « user->getEmail() « std::endl;
165       std::cout « "User saved: " « user->getName() « "-" « user->getNif()
166                 « std::endl;
167     }
168 
169     // Close the file
170     file.close();
171 
172     // Check if file was closed properly
173     if (file.is_open()) {
174       throw FileCloseError(fileNameTxt_);
175     }
176   } catch (std::exception &e) {
177     std::cerr « e.what() « std::endl;
178   }
179 }
```

4.30.4 Member Data Documentation

4.30.4.1 fileNameTxt_

```
std::string UsersServer::fileNameTxt_ = "users.txt"  [private]
```

This is the name of the file.

Definition at line 110 of file UsersServer.h.

4.30.4.2 usersDatabase_

```
UsersDatabase UsersServer::usersDatabase_  [private]
```

This is the UsersDatabase object.

Definition at line 104 of file UsersServer.h.

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

- src/UsersServer.h
- src/UsersServer.cpp

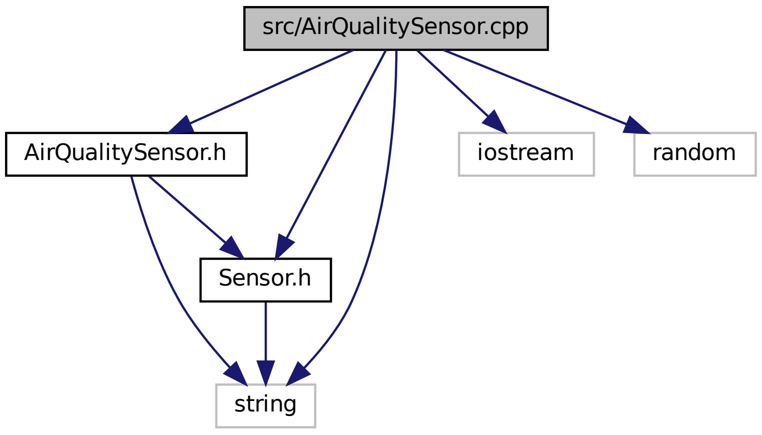
Chapter 5

File Documentation

5.1 src/AirQualitySensor.cpp File Reference

```
#include "AirQualitySensor.h"
#include <iostream>
#include <random>
#include <string>
#include "Sensor.h"
```

Include dependency graph for AirQualitySensor.cpp:



Functions

- std::ostream & operator<< (std::ostream &os, const AirQualitySensor &sensor)

5.1.1 Function Documentation

5.1.1.1 operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const AirQualitySensor & sensor )
```

Parameters

os	
sensor	

Returns

std::ostream&

Definition at line 35 of file AirQualitySensor.cpp.

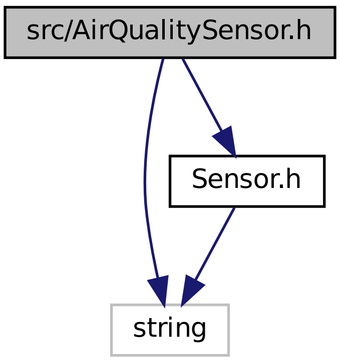
```
                                                          {
   sensor.printData();
   return os;
}
```

5.2 src/AirQualitySensor.h File Reference

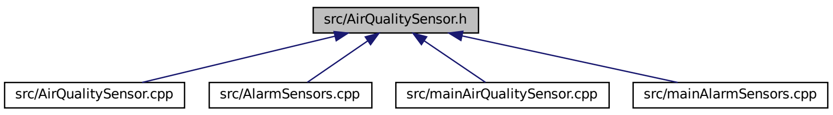
This is the class AirQualitySensor. It contains the attributes and methods of the AirQualitySensor class.

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

Include dependency graph for AirQualitySensor.h:



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



Classes

- class AirQualitySensor

5.2.1 Detailed Description

This is the class AirQualitySensor. It contains the attributes and methods of the AirQualitySensor class.

Author

Adrián Montes Linares

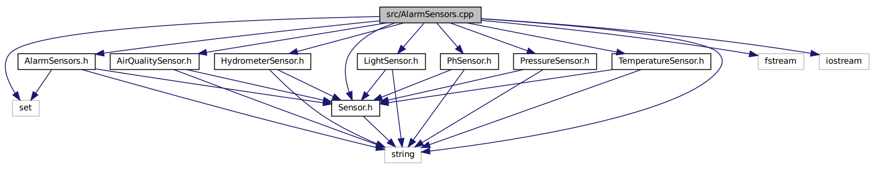
Date

21/04/2024

5.3 src/AlarmSensors.cpp File Reference

```
#include "AlarmSensors.h"
#include <fstream>
#include <iostream>
#include <set>
#include <string>
#include "Sensor.h"
#include "AirQualitySensor.h"
#include "HydrometerSensor.h"
#include "LightSensor.h"
#include "PhSensor.h"
#include "PressureSensor.h"
#include "TemperatureSensor.h"
```

Include dependency graph for AlarmSensors.cpp:



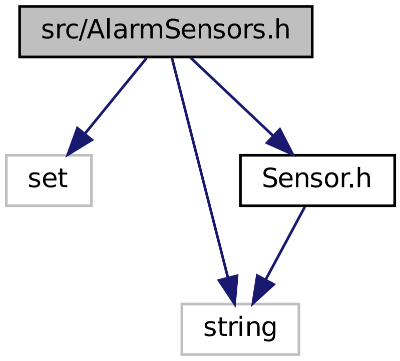
5.4 src/AlarmSensors.h File Reference

This is the class AlarmSensors. It contains the attributes and methods of the AlarmSensors class.

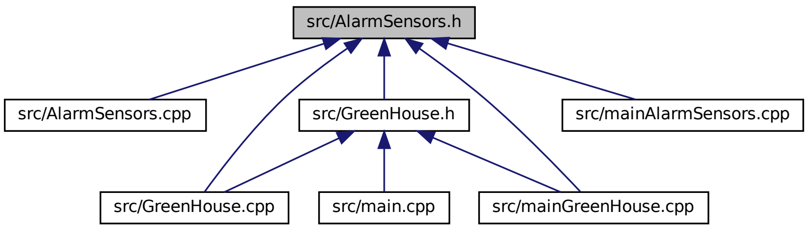
```
#include <set>
#include <string>
```

`#include "Sensor.h"`

Include dependency graph for AlarmSensors.h:



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



Classes

- class AlarmSensors

5.4.1 Detailed Description

This is the class AlarmSensors. It contains the attributes and methods of the AlarmSensors class.

Author

Adrián Montes Linares

Date

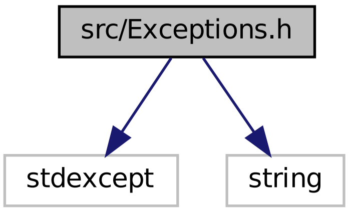
21/04/2024

5.5 src/Exceptions.h File Reference

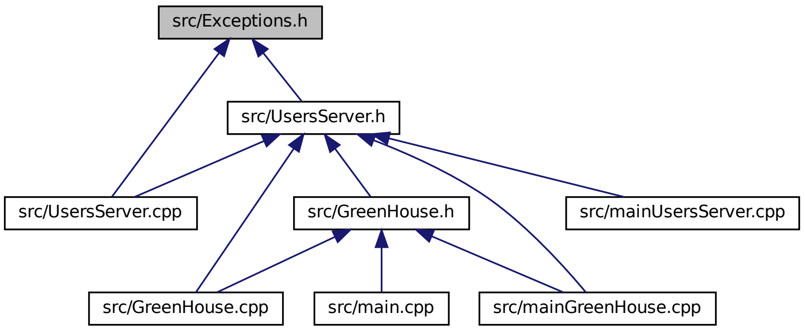
This file contains the attributes and methods of the Exceptions class.

```
#include <stdexcept>
#include <string>
```

Include dependency graph for Exceptions.h:



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



Classes

- class FileOpenError
- class FileCloseError
- class FileReadError
- class FileWriteError
- class FilePermissionError
- class FileNotFoundError
- class FileLockError
- class FileCorruptError

5.5.1 Detailed Description

This file contains the attributes and methods of the Exceptions class.

Author

Adrián Montes Linares

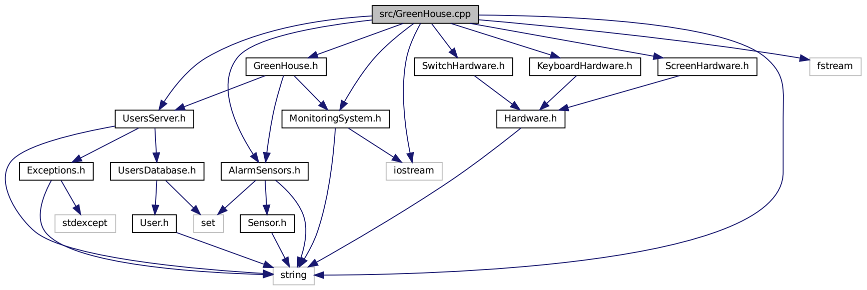
Date

21/04/2024

5.6 src/GreenHouse.cpp File Reference

```
#include "GreenHouse.h"
#include <fstream>
#include <iostream>
#include <string>
#include "AlarmSensors.h"
#include "KeyboardHardware.h"
#include "MonitoringSystem.h"
#include "ScreenHardware.h"
#include "SwitchHardware.h"
#include "UsersServer.h"
```

Include dependency graph for GreenHouse.cpp:



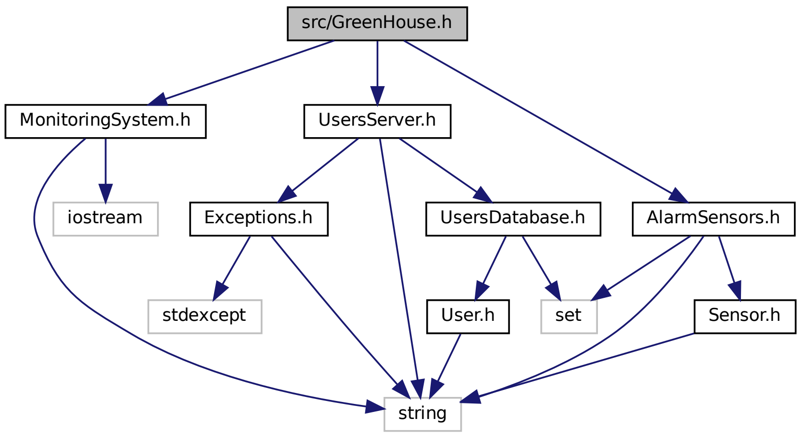
5.7 src/GreenHouse.h File Reference

This is the class GreenHouse. It contains the attributes and methods of the GreenHouse class, this class is the main of the hole system.

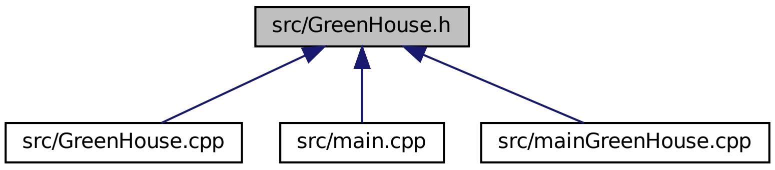
```
#include "AlarmSensors.h"
#include "MonitoringSystem.h"
```

```
#include "UsersServer.h"
```

Include dependency graph for GreenHouse.h:



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



Classes

- class GreenHouse

5.7.1 Detailed Description

This is the class GreenHouse. It contains the attributes and methods of the GreenHouse class, this class is the main of the hole system.

Author

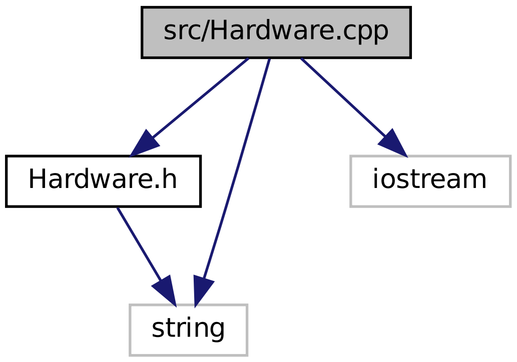
Adrián Montes Linares

Date

21/04/2024

5.8 src/Hardware.cpp File Reference

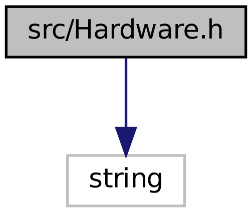
```
#include "Hardware.h"
#include <iostream>
#include <string>
```
Include dependency graph for Hardware.cpp:



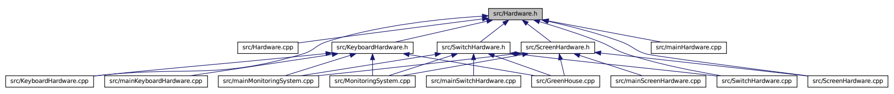
5.9 src/Hardware.h File Reference

This is the class Hardware. It contains the attributes and methods of the Hardware class, this class is the parent of the hole hardware system.

```
#include <string>
```
Include dependency graph for Hardware.h:



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



Classes

- class Hardware

5.9.1 Detailed Description

This is the class Hardware. It contains the attributes and methods of the Hardware class, this class is the parent of the hole hardware system.

Author

Adrián Montes Linares

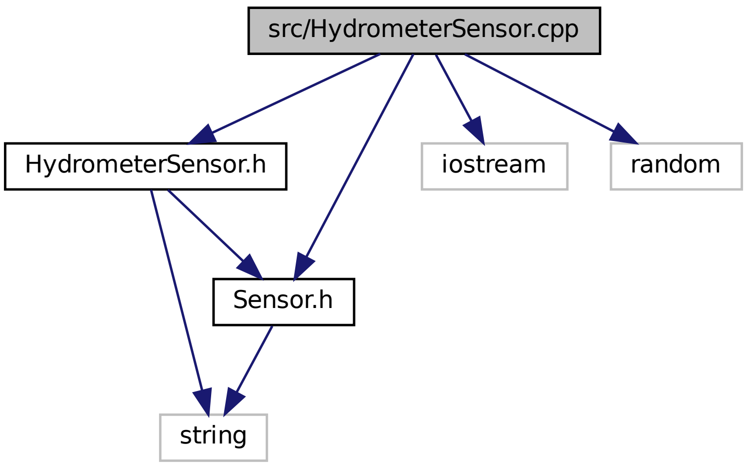
Date

21/04/2024

5.10 src/HydrometerSensor.cpp File Reference

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

Include dependency graph for HydrometerSensor.cpp:



Functions

- std::ostream & operator<< (std::ostream &os, const HydrometerSensor &sensor)

5.10.1 Function Documentation

5.10.1.1 operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const HydrometerSensor & sensor )
```

Returns

double

Definition at line 35 of file HydrometerSensor.cpp.

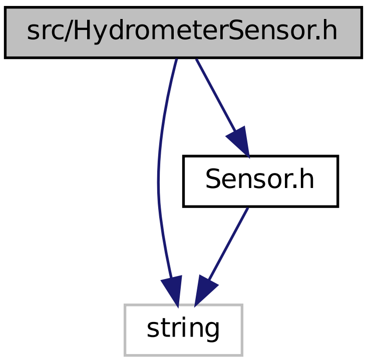
```
                                                              {
    sensor.printData();
    return os;
 }
```

5.11 src/HydrometerSensor.h File Reference

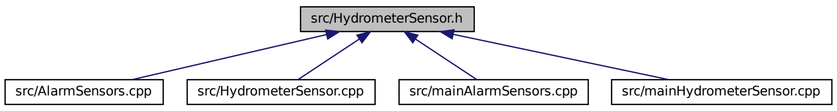
This is the class HydrometerSensor. It contains the attributes and methods of the HydrometerSensor class.

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

Include dependency graph for HydrometerSensor.h:



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



Classes

- class HydrometerSensor

5.11.1 Detailed Description

This is the class HydrometerSensor. It contains the attributes and methods of the HydrometerSensor class.

Author

Adrián Montes Linares

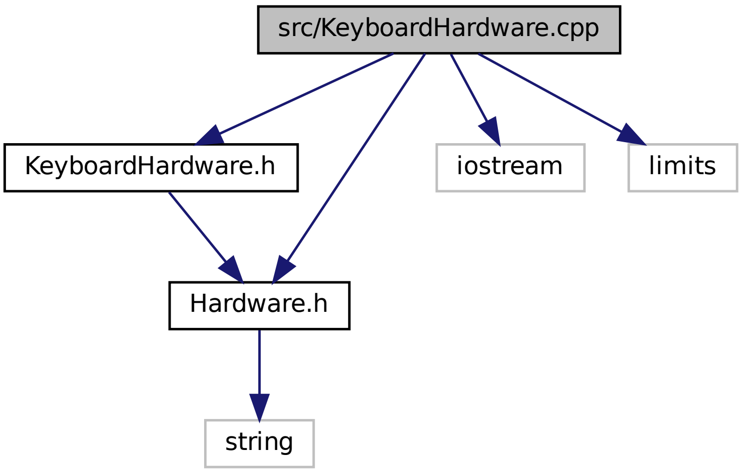
Date

21/04/2024

5.12 src/KeyboardHardware.cpp File Reference

```
#include "KeyboardHardware.h"
#include <iostream>
#include <limits>
#include "Hardware.h"
```

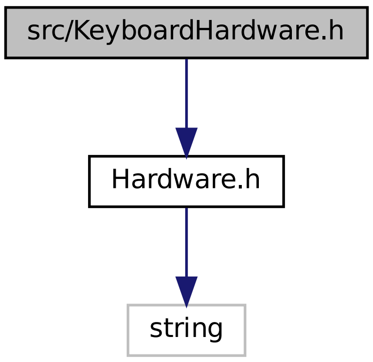
Include dependency graph for KeyboardHardware.cpp:



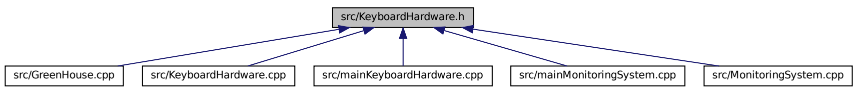
5.13 src/KeyboardHardware.h File Reference

This is the class KeyboardHardware. It contains the attributes and methods of the KeyboardHardware class, this class is a child of the Hardware class.

```
#include "Hardware.h"
```
Include dependency graph for KeyboardHardware.h:



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



Classes

- class KeyboardHardware

5.13.1 Detailed Description

This is the class KeyboardHardware. It contains the attributes and methods of the KeyboardHardware class, this class is a child of the Hardware class.

Author

Adrián Montes Linares

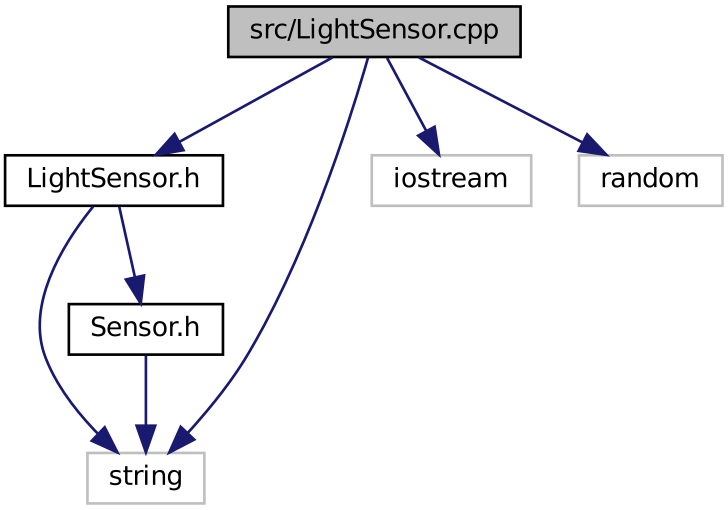
Date

21/04/2024

5.14 src/LightSensor.cpp File Reference

```
#include "LightSensor.h"
#include <iostream>
#include <random>
#include <string>
```

Include dependency graph for LightSensor.cpp:



Functions

- std::ostream & operator<< (std::ostream &os, const LightSensor &sensor)

5.14.1 Function Documentation

5.14.1.1 operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const LightSensor & sensor )
```

Returns

int

Definition at line 33 of file LightSensor.cpp.

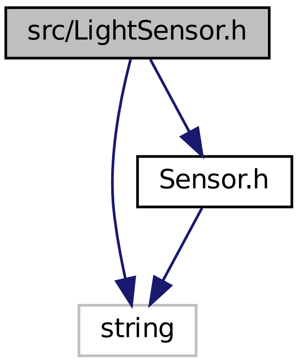
```
                                                                        {
    sensor.printData();
    return os;
}
```

5.15 src/LightSensor.h File Reference

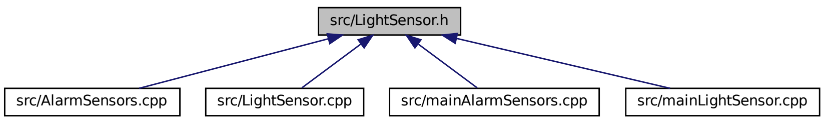
This is the class LightSensor. It contains the attributes and methods of the LightSensor class.

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

Include dependency graph for LightSensor.h:



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



Classes

- class LightSensor

5.15.1 Detailed Description

This is the class LightSensor. It contains the attributes and methods of the LightSensor class.

Author

Adrián Montes Linares

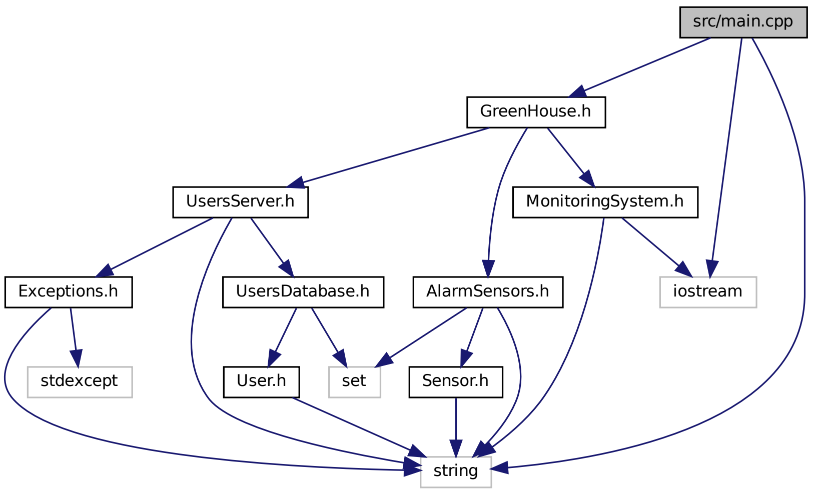
Date

21/04/2024

5.16 src/main.cpp File Reference

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

Include dependency graph for main.cpp:



Functions

- int main ()

5.16.1 Function Documentation

5.16.1.1 main()

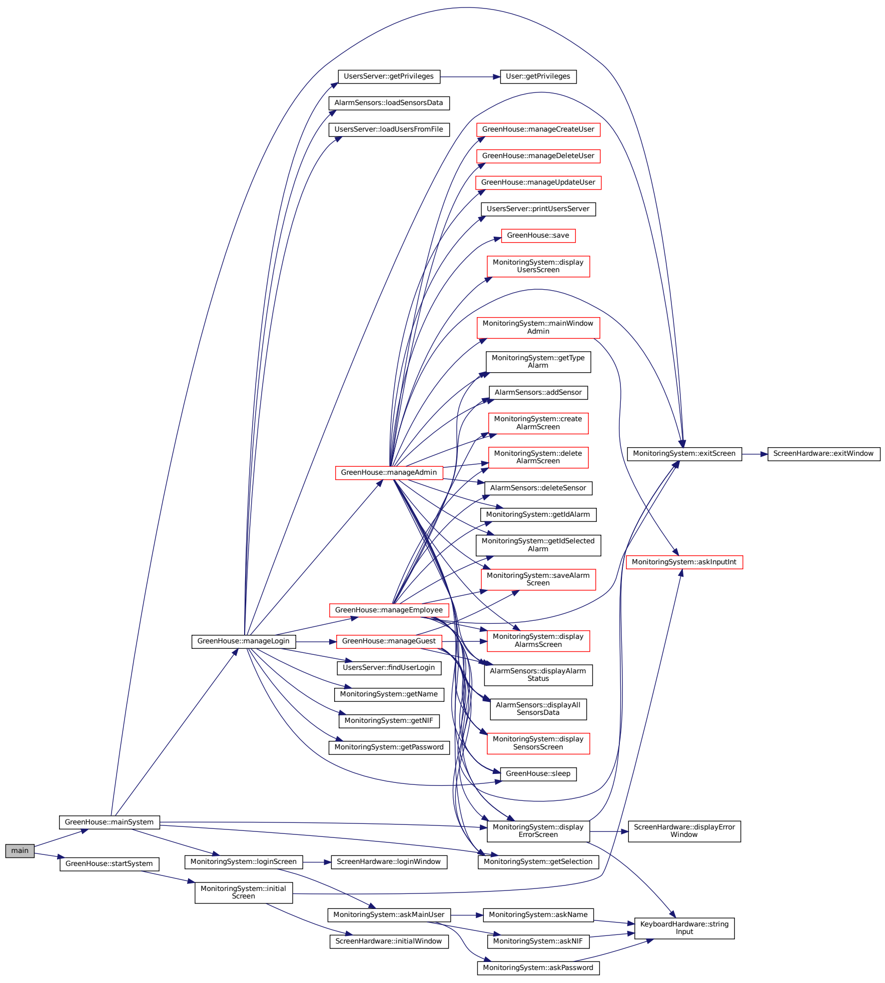
```
int main ( )
```

Definition at line 6 of file main.cpp.

```
            {
  GreenHouse greenhouse;
  greenhouse.startSystem();
  greenhouse.mainSystem();
   return 0;
}
```

References GreenHouse::mainSystem(), and GreenHouse::startSystem().

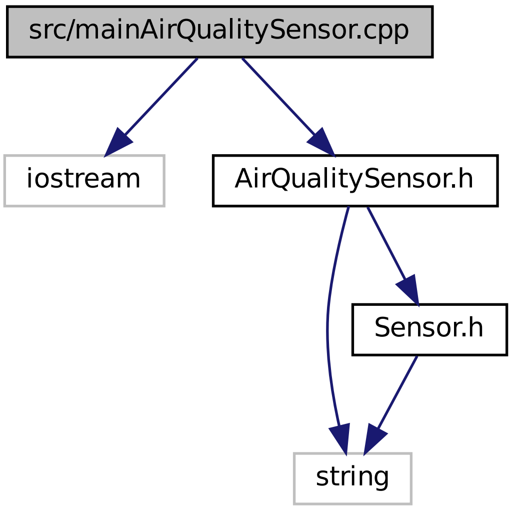
Here is the call graph for this function:


UsersServer::getPrivileges
User::getPrivileges
AlarmSensors::loadSensorsData
UsersServer::loadUsersFromFile
GreenHouse::manageCreateUser
GreenHouse::manageDeleteUser
GreenHouse::manageUpdateUser
UsersServer::printUsersServer
GreenHouse::save
MonitoringSystem::display UsersScreen
MonitoringSystem::mainWindow Admin
MonitoringSystem::getType Alarm
AlarmSensors::addSensor
MonitoringSystem::create AlarmScreen
MonitoringSystem::delete AlarmScreen
MonitoringSystem::exitScreen
ScreenHardware::exitWindow
GreenHouse::manageAdmin
AlarmSensors::deleteSensor
MonitoringSystem::getIdAlarm
MonitoringSystem::getIdSelected Alarm
MonitoringSystem::askInputInt
MonitoringSystem::saveAlarm Screen
GreenHouse::manageEmployee
GreenHouse::manageLogin
GreenHouse::manageGuest
MonitoringSystem::display AlarmsScreen
UsersServer::findUserLogin
AlarmSensors::displayAlarm Status
MonitoringSystem::getName
AlarmSensors::displayAll SensorsData
MonitoringSystem::getNIF
MonitoringSystem::display SensorsScreen
MonitoringSystem::getPassword
GreenHouse::sleep
GreenHouse::mainSystem
MonitoringSystem::display ErrorScreen
ScreenHardware::displayError Window
main
GreenHouse::startSystem
MonitoringSystem::loginScreen
ScreenHardware::loginWindow
MonitoringSystem::getSelection
MonitoringSystem::initial Screen
MonitoringSystem::askMainUser
MonitoringSystem::askName
KeyboardHardware::string Input
ScreenHardware::initialWindow
MonitoringSystem::askNIF
MonitoringSystem::askPassword

5.17 src/mainAirQualitySensor.cpp File Reference

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

Include dependency graph for mainAirQualitySensor.cpp:



Functions

- int main ()

5.17.1 Function Documentation

5.17.1.1 main()

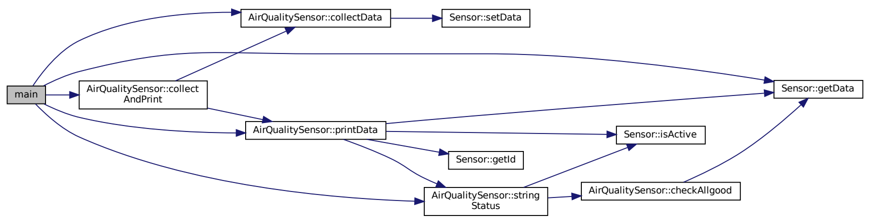
```
int main ( )
```

Definition at line 6 of file mainAirQualitySensor.cpp.

```
          {
 // Genero un sensor tipo calidad del aire
 AirQualitySensor airQualitySensor(1, true);
 // Imprimo la calidad del aire por defecto
  airQualitySensor.printData();
  // Cambio el valor de la calidad del aire
  airQualitySensor.collectData();
  // Imprimo la nueva calidad del aire
  airQualitySensor.printData();
  // Vuelvo a imprimir la calidad del aire
  cout « "Air Quality: " « airQualitySensor.getData() « endl;
  cout « "Status: " « airQualitySensor.stringStatus() « endl;

  airQualitySensor.collectData();
  // Imprimo el sensor de nuevo
  airQualitySensor.printData();

  // Print collect and print
  airQualitySensor.collectAndPrint();
}
```

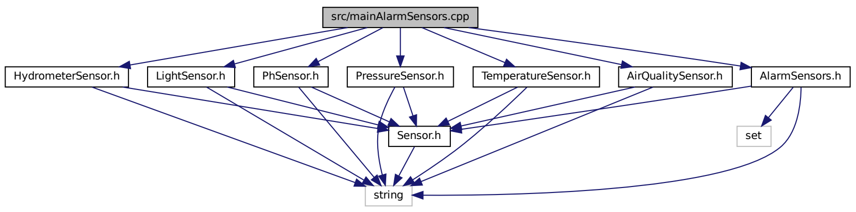
References AirQualitySensor::collectAndPrint(), AirQualitySensor::collectData(), Sensor::getData(), AirQuality↩ Sensor::printData(), and AirQualitySensor::stringStatus().

Here is the call graph for this function:


AirQualitySensor::collectData
Sensor::setData
main
AirQualitySensor::collect AndPrint
Sensor::getData
AirQualitySensor::printData
Sensor::isActive
Sensor::getId
AirQualitySensor::string Status
AirQualitySensor::checkAllgood

5.18 src/mainAlarmSensors.cpp File Reference

```
#include "AirQualitySensor.h"
#include "AlarmSensors.h"
#include "HydrometerSensor.h"
#include "LightSensor.h"
#include "PhSensor.h"
#include "PressureSensor.h"
#include "TemperatureSensor.h"
```
Include dependency graph for mainAlarmSensors.cpp:



Functions

- int main ()

5.18.1 Function Documentation

5.18.1.1 main()

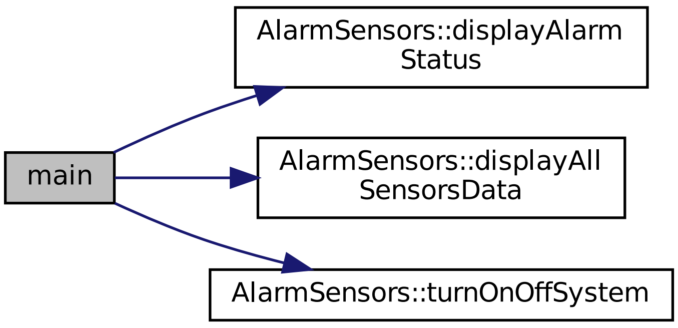
```
int main ( )
```

Definition at line 9 of file mainAlarmSensors.cpp.

```
          {
/*
TemperatureSensor temp(1, true);
AirQualitySensor air(2, true);
HydrometerSensor hyd(3, true);
PressureSensor pres(4, true);
LightSensor light(5, true);
PhSensor ph(6, true);
*/

AlarmSensors *alarm = new AlarmSensors(
    new TemperatureSensor(1, true), new AirQualitySensor(2, true),
    new HydrometerSensor(3, true), new PressureSensor(4, true),
    new LightSensor(5, true), new PhSensor(6, true));
// Todas las opetaciones de la clase AlarmSensors
// AlarmSensors alarm(&temp, &air, &hyd, &pres, &light, &ph);
alarm->displayAlarmStatus();
alarm->displayAllSensorsData();
alarm->displayAlarmStatus();
alarm->turnOnOffSystem(0);
alarm->displayAllSensorsData();
alarm->displayAlarmStatus();

return 0;
}
```

References AlarmSensors::displayAlarmStatus(), AlarmSensors::displayAllSensorsData(), and AlarmSensors↩
::turnOnOffSystem().

Here is the call graph for this function:

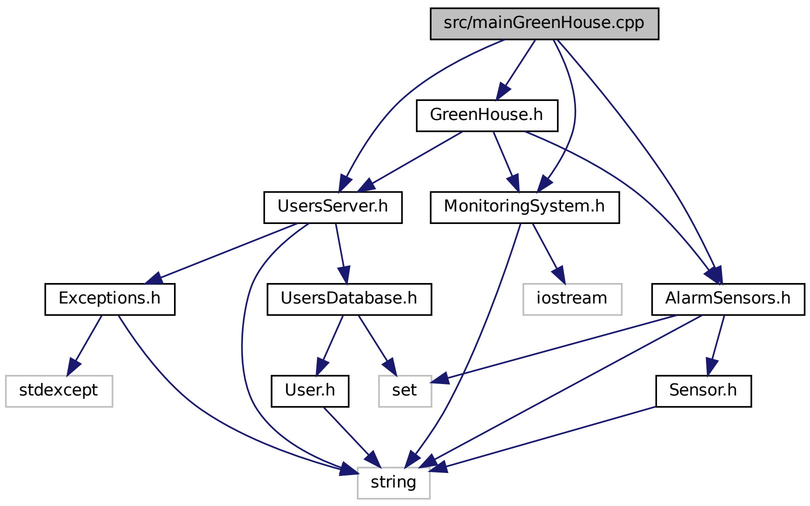

main
AlarmSensors::displayAlarm Status
AlarmSensors::displayAll SensorsData
AlarmSensors::turnOnOffSystem

5.19 src/mainGreenHouse.cpp File Reference

```
#include "AlarmSensors.h"
#include "GreenHouse.h"
#include "MonitoringSystem.h"
```

`#include "UsersServer.h"`

Include dependency graph for mainGreenHouse.cpp:



Functions

- int main ()

5.19.1 Function Documentation

5.19.1.1 main()

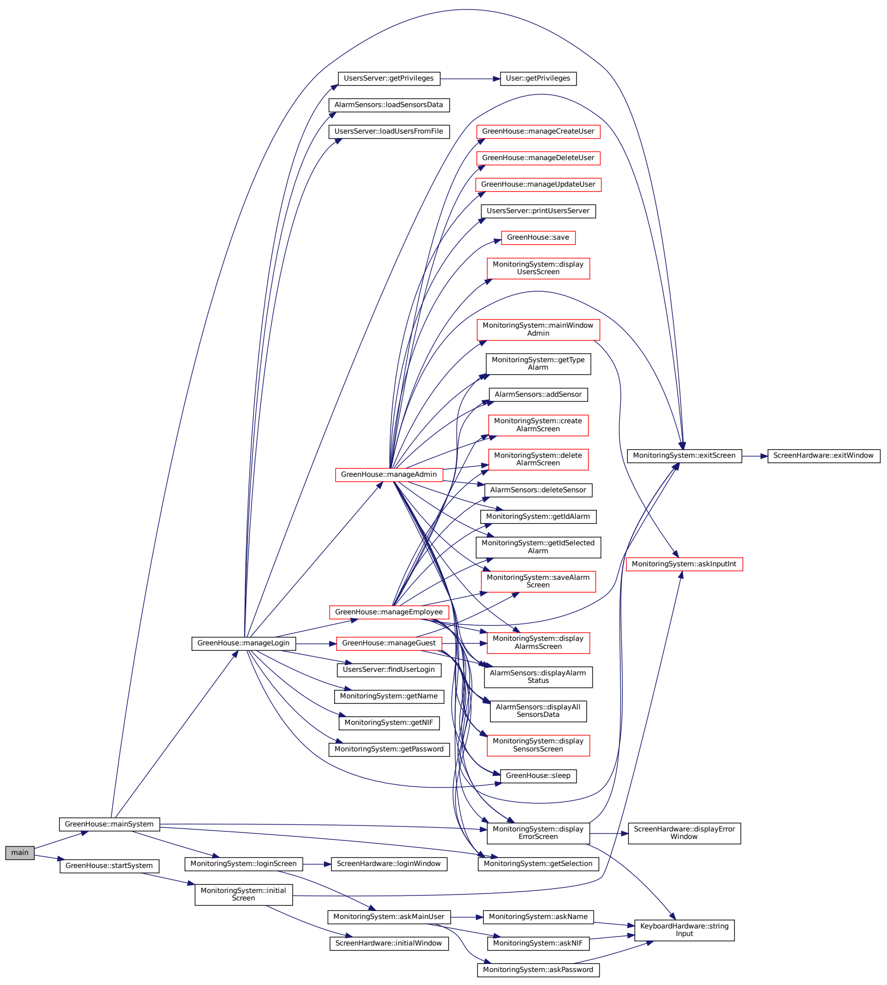
```
int main ( )
```

Definition at line 6 of file mainGreenHouse.cpp.

```
            {
    GreenHouse gh;
    gh.startSystem();
    gh.mainSystem();
     return 0;
 }
```

References GreenHouse::mainSystem(), and GreenHouse::startSystem().

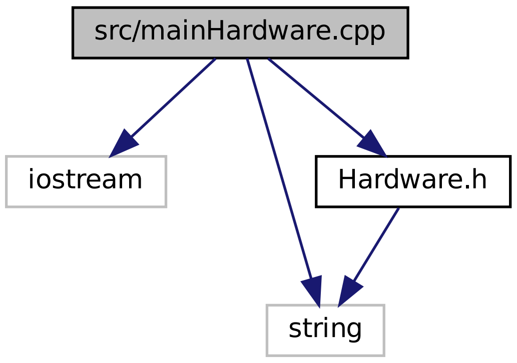
Here is the call graph for this function:


UsersServer::getPrivileges
User::getPrivileges
AlarmSensors::loadSensorsData
UsersServer::loadUsersFromFile
GreenHouse::manageCreateUser
GreenHouse::manageDeleteUser
GreenHouse::manageUpdateUser
UsersServer::printUsersServer
GreenHouse::save
MonitoringSystem::display UsersScreen
MonitoringSystem::mainWindow Admin
MonitoringSystem::getType Alarm
AlarmSensors::addSensor
MonitoringSystem::create AlarmScreen
MonitoringSystem::delete AlarmScreen
GreenHouse::manageAdmin
MonitoringSystem::exitScreen
ScreenHardware::exitWindow
AlarmSensors::deleteSensor
MonitoringSystem::getIdAlarm
MonitoringSystem::getIdSelected Alarm
MonitoringSystem::askInputInt
MonitoringSystem::saveAlarm Screen
GreenHouse::manageEmployee
GreenHouse::manageLogin
GreenHouse::manageGuest
MonitoringSystem::display AlarmsScreen
UsersServer::findUserLogin
AlarmSensors::displayAlarm Status
MonitoringSystem::getName
AlarmSensors::displayAll SensorsData
MonitoringSystem::getNIF
MonitoringSystem::display SensorsScreen
MonitoringSystem::getPassword
GreenHouse::sleep
GreenHouse::mainSystem
MonitoringSystem::display ErrorScreen
ScreenHardware::displayError Window
main
GreenHouse::startSystem
MonitoringSystem::loginScreen
ScreenHardware::loginWindow
MonitoringSystem::getSelection
MonitoringSystem::initial Screen
MonitoringSystem::askMainUser
MonitoringSystem::askName
KeyboardHardware::string Input
ScreenHardware::initialWindow
MonitoringSystem::askNIF
MonitoringSystem::askPassword

5.20 src/mainHardware.cpp File Reference

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

Include dependency graph for mainHardware.cpp:



Functions

- int main ()

5.20.1 Function Documentation

5.20.1.1 main()

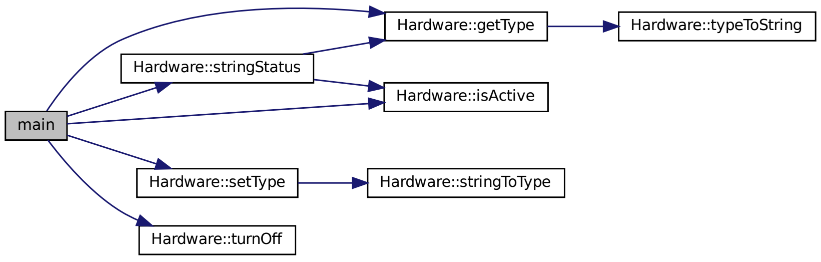
```
int main ( )
```

Definition at line 7 of file mainHardware.cpp.

```
           {
  // Aqui tengo que probar los distintos metodos de un objeto Hardware
  Hardware hardware(true, Hardware::Types_Hardware::SWITCH);
   // Pruebas get type set type
   std::cout « hardware.getType() « std::endl;
   hardware.setType("SCREEN");
   std::cout « hardware.getType() « std::endl;

   // Vamos a ver si esta activado y luego desactivamos
   if (hardware.isActive()) {
     std::cout « "Esta activo" « std::endl;
   } else {
     std::cout « "No esta activo" « std::endl;
   }
   hardware.turnOff();
   if (hardware.isActive()) {
     std::cout « "Esta activo" « std::endl;
   } else {
     std::cout « "No esta activo" « std::endl;
   }
   std::cout « hardware.stringStatus() « std::endl;
}
```

References Hardware::getType(), Hardware::isActive(), Hardware::setType(), Hardware::stringStatus(), and Hardware::turnOff().

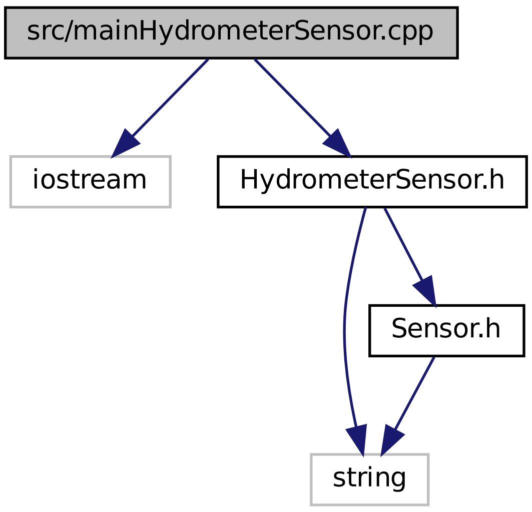
Here is the call graph for this function:



5.21 src/mainHydrometerSensor.cpp File Reference

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

Include dependency graph for mainHydrometerSensor.cpp:



Functions

- int main ()

5.21.1 Function Documentation

5.21.1.1 main()

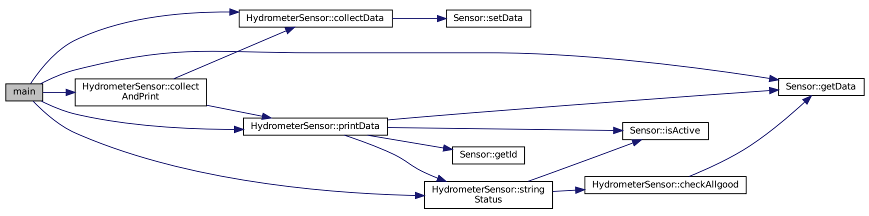
```
int main ( )
```

Definition at line 6 of file mainHydrometerSensor.cpp.

```
            {
  // Genero un sensor tipo hidrometro (mide la humedad del aire)
  HydrometerSensor hydrometer(1, true);
  // Imprimo la hidrometro (mide la humedad del aire) por defecto
   hydrometer.printData();
   // Cambio el valor de la hidrometro (mide la humedad del aire)
   hydrometer.collectData();
   // Imprimo la nueva hidrometro (mide la humedad del aire)
   hydrometer.printData();
   // Vuelvo a imprimir la hidrometro (mide la humedad del aire)
   cout « "Air Quality: " « hydrometer.getData() « endl;
   cout « "Status: " « hydrometer.stringStatus() « endl;

   hydrometer.collectData();
   // Imprimo el sensor de nuevo
   hydrometer.printData();

   // Print collect and print
   hydrometer.collectAndPrint();
}
```

References HydrometerSensor::collectAndPrint(), HydrometerSensor::collectData(), Sensor::getData(), Hydrometer←Sensor::printData(), and HydrometerSensor::stringStatus().

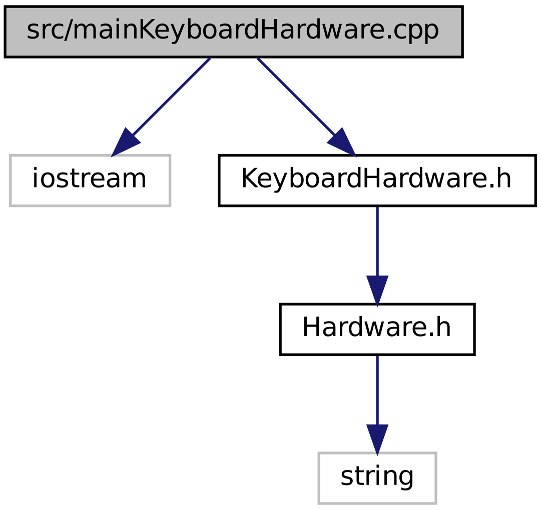
Here is the call graph for this function:


main
HydrometerSensor::collectData
Sensor::setData
HydrometerSensor::collect AndPrint
Sensor::getData
HydrometerSensor::printData
Sensor::isActive
Sensor::getId
HydrometerSensor::string Status
HydrometerSensor::checkAllgood

5.22 src/mainKeyboardHardware.cpp File Reference

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

Include dependency graph for mainKeyboardHardware.cpp:



Functions

- int main ()

5.22.1 Function Documentation

5.22.1.1 main()

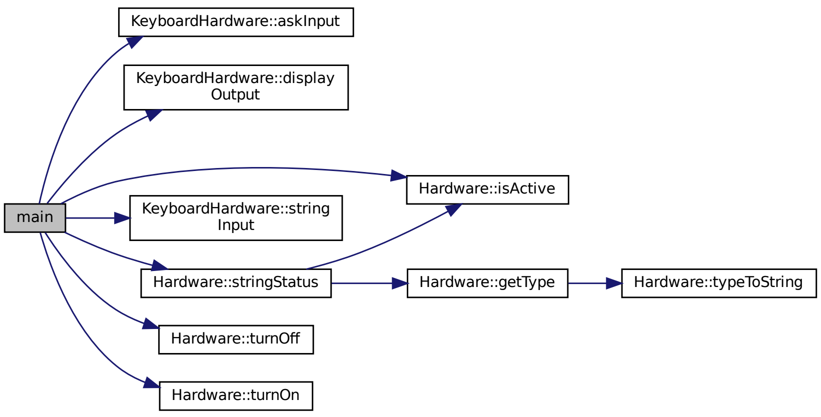
```
int main ( )
```

Definition at line 6 of file mainKeyboardHardware.cpp.

```
            {
  KeyboardHardware keyboard(true);
  if (keyboard.isActive()) {
    std::cout « "Active" « std::endl;
  } else {
    std::cout « "No Active" « std::endl;
  }
  // apagamos
  keyboard.turnOff();
  std::cout « keyboard.stringStatus() « std::endl;
  // encnedemos
  keyboard.turnOn();
  std::cout « keyboard.stringStatus() « std::endl;

  // Preguntamos un input y luego mostramos el ultimo input
  std::cout « keyboard.askInput() « std::endl;
  std::cout « keyboard.stringInput() « std::endl;
  keyboard.displayOutput();

  return 0;
}
```

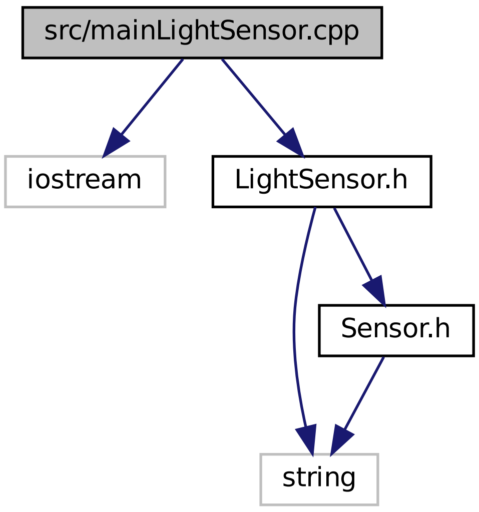
References KeyboardHardware::askInput(), KeyboardHardware::displayOutput(), Hardware::isActive(), Keyboard↩
Hardware::stringInput(), Hardware::stringStatus(), Hardware::turnOff(), and Hardware::turnOn().

Here is the call graph for this function:



5.23 src/mainLightSensor.cpp File Reference

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



Functions

- int main ()

5.23.1 Function Documentation

5.23.1.1 main()

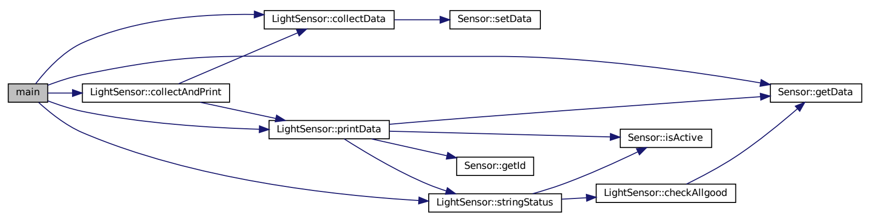
```
int main ( )
```

Definition at line 5 of file mainLightSensor.cpp.

```
5            {
6   LightSensor lightSensor(1, true);
7   // Imprimo la luz por defecto
8   lightSensor.printData();
9   // Cambio el valor de la luz
10   lightSensor.collectData();
11   // Imprimo la nueva luz
12   lightSensor.printData();
13   // Print collect and print
14   std::cout « "Light: " « lightSensor.getData() « std::endl;
15   // Print collect and print
16   std::cout « "Status: " « lightSensor.stringStatus() « std::endl;
17   // Imprimo el sensor de nuevo
18   lightSensor.collectData();
19   // Imprimo el sensor de nuevo
20   lightSensor.printData();
21   // Print collect and print
22   lightSensor.collectAndPrint();
23
24   return 0;
25 }
```

References LightSensor::collectAndPrint(), LightSensor::collectData(), Sensor::getData(), LightSensor::printData(), and LightSensor::stringStatus().

Here is the call graph for this function:

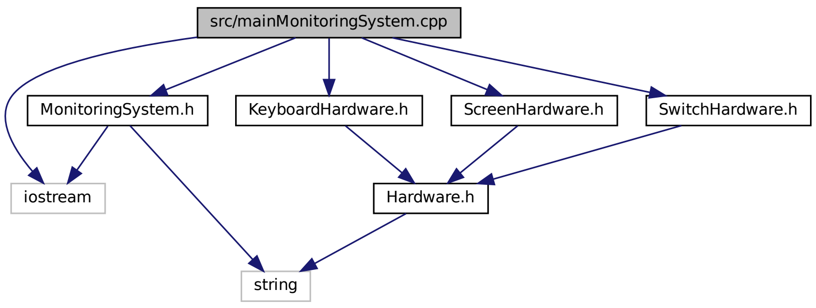


5.24 src/mainMonitoringSystem.cpp File Reference

```
#include <iostream>
#include "KeyboardHardware.h"
#include "MonitoringSystem.h"
#include "ScreenHardware.h"
```

```
#include "SwitchHardware.h"
```

Include dependency graph for mainMonitoringSystem.cpp:



Functions

- int main ()

5.24.1 Function Documentation

5.24.1.1 main()

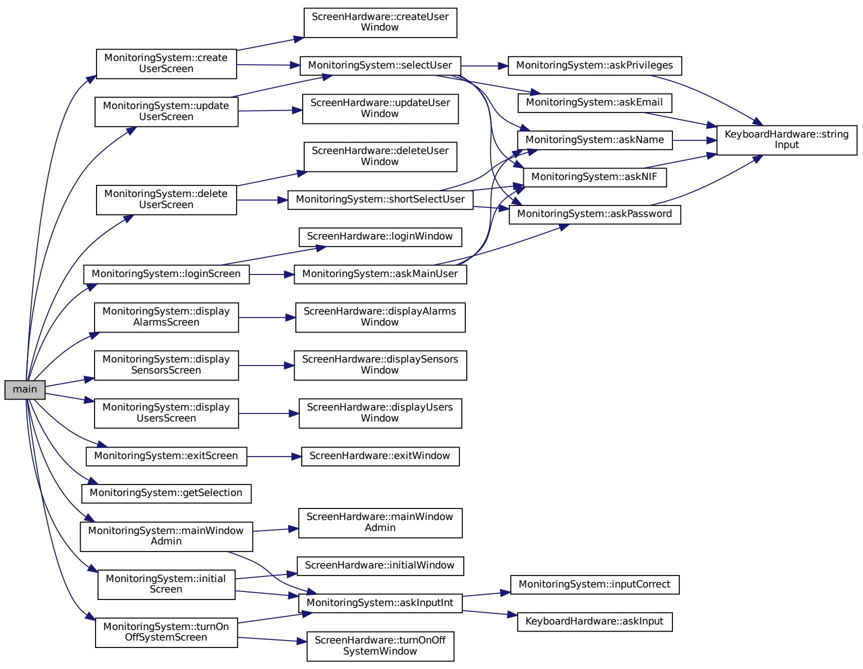
```
int main ( )
```

Definition at line 9 of file mainMonitoringSystem.cpp.

```
          {
  // Creo un MonitoringSystem
  MonitoringSystem *ms =
      new MonitoringSystem(new ScreenHardware(true), new KeyboardHardware(true),
                           new SwitchHardware(true));
  // Llamo a la funcion initialScreen() que muestra el menu inicial
  ms->initialScreen();
  if (ms->getSelection() == 1) {
    ms->loginScreen();
    // (SUponemos que ha entrado bien el usuario y que es admin)
    bool exit = false;
    do {
      ms->mainWindowAdmin();
      // Ahora probamos la ventana de employee
      // ms->mainWindowEmployee();
      // Ahora probamos la ventana de guest
      // ms->mainWindowGuest();
      if (ms->getSelection() == 1) {
        ms->createUserScreen();
      } else if (ms->getSelection() == 2) {
        ms->deleteUserScreen();
      } else if (ms->getSelection() == 3) {
        ms->updateUserScreen();
      } else if (ms->getSelection() == 4) {
        ms->displayUsersScreen();
      } else if (ms->getSelection() == 5) {
        ms->displaySensorsScreen();
      } else if (ms->getSelection() == 6) {
        ms->displayAlarmsScreen();
      } else if (ms->getSelection() == 7) {
        ms->turnOnOffSystemScreen();
      } else if (ms->getSelection() == 8) {
        ms->exitScreen();
        exit = true;
      }
    } while (!exit);

    /*ms->createUserScreen();
    ms->deleteUserScreen();
    ms->updateUserScreen();
    ms->displayUsersScreen();
    ms->displaySensorsScreen();
    ms->displayAlarmsScreen();
    ms->turnOnOffSystemScreen();
    ms->displayErrorScreen();*/
  } else {
    // Llamo a la funcion exitScreen() que muestra el exitWindow
    ms->exitScreen();
  }

  return 0;
}
```

References MonitoringSystem::createUserScreen(), MonitoringSystem::deleteUserScreen(), MonitoringSystem↩ ::displayAlarmsScreen(), MonitoringSystem::displaySensorsScreen(), MonitoringSystem::displayUsersScreen(), MonitoringSystem::exitScreen(), MonitoringSystem::getSelection(), MonitoringSystem::initialScreen(), Monitoring↩ System::loginScreen(), MonitoringSystem::mainWindowAdmin(), MonitoringSystem::turnOnOffSystemScreen(), and MonitoringSystem::updateUserScreen().

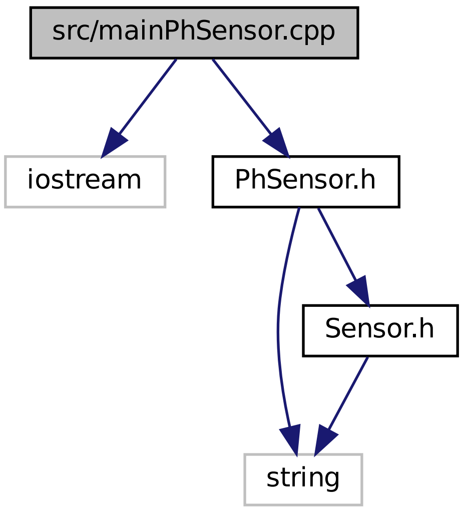
Here is the call graph for this function:


main
MonitoringSystem::create UserScreen
MonitoringSystem::update UserScreen
MonitoringSystem::delete UserScreen
MonitoringSystem::loginScreen
MonitoringSystem::display AlarmsScreen
MonitoringSystem::display SensorsScreen
MonitoringSystem::display UsersScreen
MonitoringSystem::exitScreen
MonitoringSystem::getSelection
MonitoringSystem::mainWindow Admin
MonitoringSystem::initial Screen
MonitoringSystem::turnOn OffSystemScreen
ScreenHardware::createUser Window
MonitoringSystem::selectUser
ScreenHardware::updateUser Window
ScreenHardware::deleteUser Window
MonitoringSystem::shortSelectUser
ScreenHardware::loginWindow
MonitoringSystem::askMainUser
ScreenHardware::displayAlarms Window
ScreenHardware::displaySensors Window
ScreenHardware::displayUsers Window
ScreenHardware::exitWindow
ScreenHardware::mainWindow Admin
ScreenHardware::initialWindow
MonitoringSystem::askInputInt
ScreenHardware::turnOnOff SystemWindow
MonitoringSystem::askPrivileges
MonitoringSystem::askEmail
MonitoringSystem::askName
MonitoringSystem::askNIF
MonitoringSystem::askPassword
KeyboardHardware::string Input
MonitoringSystem::inputCorrect
KeyboardHardware::askInput

5.25 src/mainPhSensor.cpp File Reference

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

Include dependency graph for mainPhSensor.cpp:



Functions

- int main ()

5.25.1 Function Documentation

5.25.1.1 main()

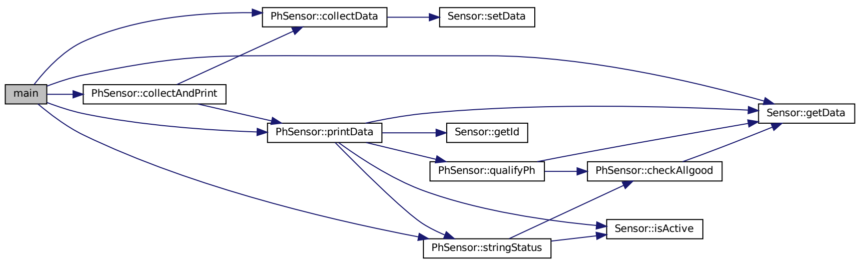
```
int main ( )
```

Definition at line 6 of file mainPhSensor.cpp.

```
        {
 PhSensor phsensor(1, true);
 // Imprimo el ph por defecto
 phsensor.printData();
  // Cambio el valor de la luz
  phsensor.collectData();
  // Imprimo la nueva luz
  phsensor.printData();
  // Print collect and print
  std::cout « "PH: " « phsensor.getData() « std::endl;
  // Print collect and print
  std::cout « "Status: " « phsensor.stringStatus() « std::endl;
  // Imprimo el sensor de nuevo
  phsensor.collectData();
  // Imprimo el sensor de nuevo
  phsensor.printData();
  // Print collect and print
  phsensor.collectAndPrint();
}
```

References PhSensor::collectAndPrint(), PhSensor::collectData(), Sensor::getData(), PhSensor::printData(), and PhSensor::stringStatus().

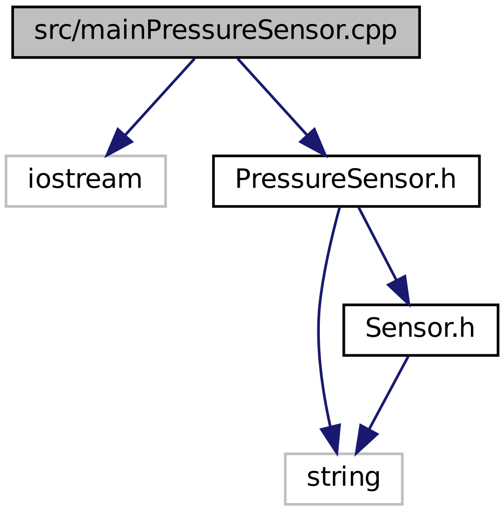
Here is the call graph for this function:



5.26 src/mainPressureSensor.cpp File Reference

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

Include dependency graph for mainPressureSensor.cpp:



Functions

- int main ()

5.26.1 Function Documentation

5.26.1.1 main()

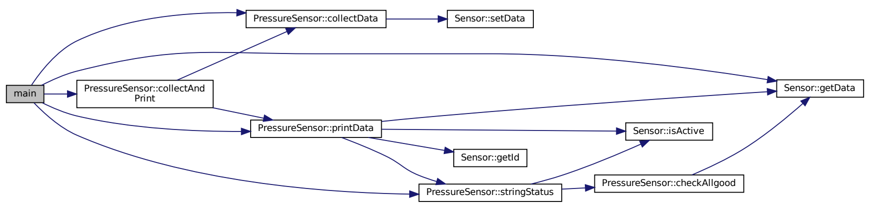
```
int main ( )
```

Definition at line 6 of file mainPressureSensor.cpp.

```
6           {
7   PressureSensor pressureSensor(1, true);
8   // Imprimo la presion por defecto
9   pressureSensor.printData();
10   // Cambio el valor de la presion
11   pressureSensor.collectData();
12   // Imprimo la nueva presion
13   pressureSensor.printData();
14   // Vuelvo a imprimir la presion
15   cout « "Pressure: " « pressureSensor.getData() « endl;
16   cout « "Status: " « pressureSensor.stringStatus() « endl;
17   pressureSensor.collectData();
18   // Imprimo el sensor de nuevo
19   pressureSensor.printData();
20   // Print collect and print
21   pressureSensor.collectAndPrint();
22 }
```

References PressureSensor::collectAndPrint(), PressureSensor::collectData(), Sensor::getData(), Pressure←Sensor::printData(), and PressureSensor::stringStatus().

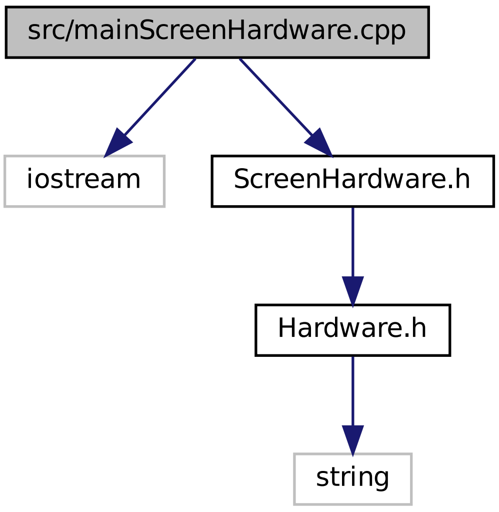
Here is the call graph for this function:


main
PressureSensor::collectData
Sensor::setData
PressureSensor::collectAnd Print
Sensor::getData
PressureSensor::printData
Sensor::isActive
Sensor::getId
PressureSensor::stringStatus
PressureSensor::checkAllgood

5.27 src/mainScreenHardware.cpp File Reference

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

Include dependency graph for mainScreenHardware.cpp:



Functions

- int main ()

5.27.1 Function Documentation

5.27.1.1 main()

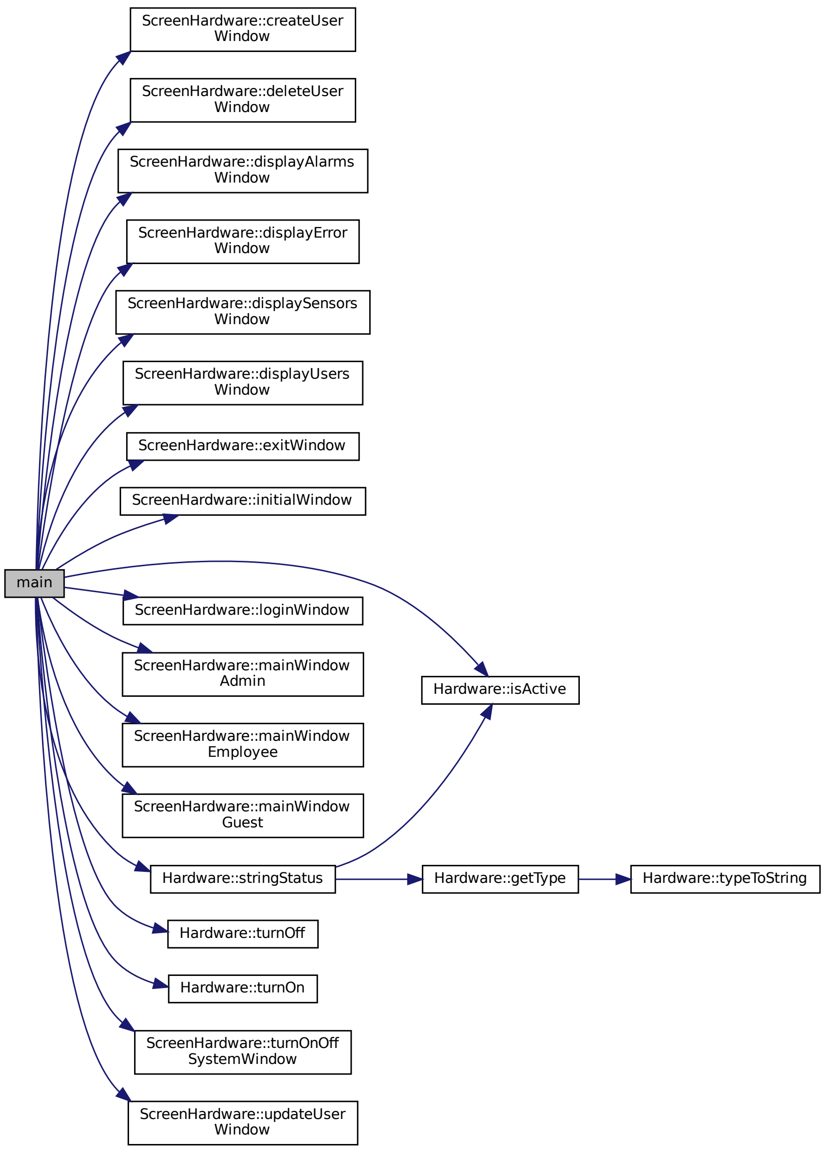
```
int main ( )
```

Definition at line 6 of file mainScreenHardware.cpp.

```
            {
  ScreenHardware screenHardware(true);
  if (screenHardware.isActive()) {
    std::cout « "Active" « std::endl;
   } else {
     std::cout « "No Active" « std::endl;
   }
   // apagamos
   screenHardware.turnOff();
   std::cout « screenHardware.stringStatus() « std::endl;
   // encnedemos
   screenHardware.turnOn();
   std::cout « screenHardware.stringStatus() « std::endl;
   // Ahora mostramos cada tipo de pantalla para comprobar que funciona
   screenHardware.initialWindow();
   screenHardware.loginWindow();
   screenHardware.mainWindowAdmin();
   screenHardware.mainWindowEmployee();
   screenHardware.mainWindowGuest();
   screenHardware.createUserWindow();
   screenHardware.deleteUserWindow();
   screenHardware.updateUserWindow();
   screenHardware.displayUsersWindow();
   screenHardware.displaySensorsWindow();
   screenHardware.displayAlarmsWindow();
   screenHardware.turnOnOffSystemWindow();
   screenHardware.displayErrorWindow();
   screenHardware.exitWindow();
 }
```

References ScreenHardware::createUserWindow(), ScreenHardware::deleteUserWindow(), ScreenHardware↩ ::displayAlarmsWindow(), ScreenHardware::displayErrorWindow(), ScreenHardware::displaySensorsWindow(), ScreenHardware::displayUsersWindow(), ScreenHardware::exitWindow(), ScreenHardware::initialWindow(), Hardware::isActive(), ScreenHardware::loginWindow(), ScreenHardware::mainWindowAdmin(), ScreenHardware↩ ::mainWindowEmployee(), ScreenHardware::mainWindowGuest(), Hardware::stringStatus(), Hardware::turnOff(), Hardware::turnOn(), ScreenHardware::turnOnOffSystemWindow(), and ScreenHardware::updateUserWindow().

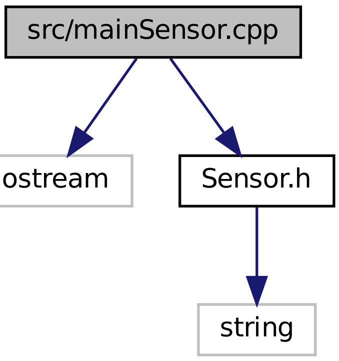
Here is the call graph for this function:


ScreenHardware::createUser Window
ScreenHardware::deleteUser Window
ScreenHardware::displayAlarms Window
ScreenHardware::displayError Window
ScreenHardware::displaySensors Window
ScreenHardware::displayUsers Window
ScreenHardware::exitWindow
ScreenHardware::initialWindow
main
ScreenHardware::loginWindow
ScreenHardware::mainWindow Admin
Hardware::isActive
ScreenHardware::mainWindow Employee
ScreenHardware::mainWindow Guest
Hardware::stringStatus
Hardware::getType
Hardware::typeToString
Hardware::turnOff
Hardware::turnOn
ScreenHardware::turnOnOff SystemWindow
ScreenHardware::updateUser Window

5.28 src/mainSensor.cpp File Reference

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

Include dependency graph for mainSensor.cpp:



Functions

- int main ()

5.28.1 Function Documentation

5.28.1.1 main()

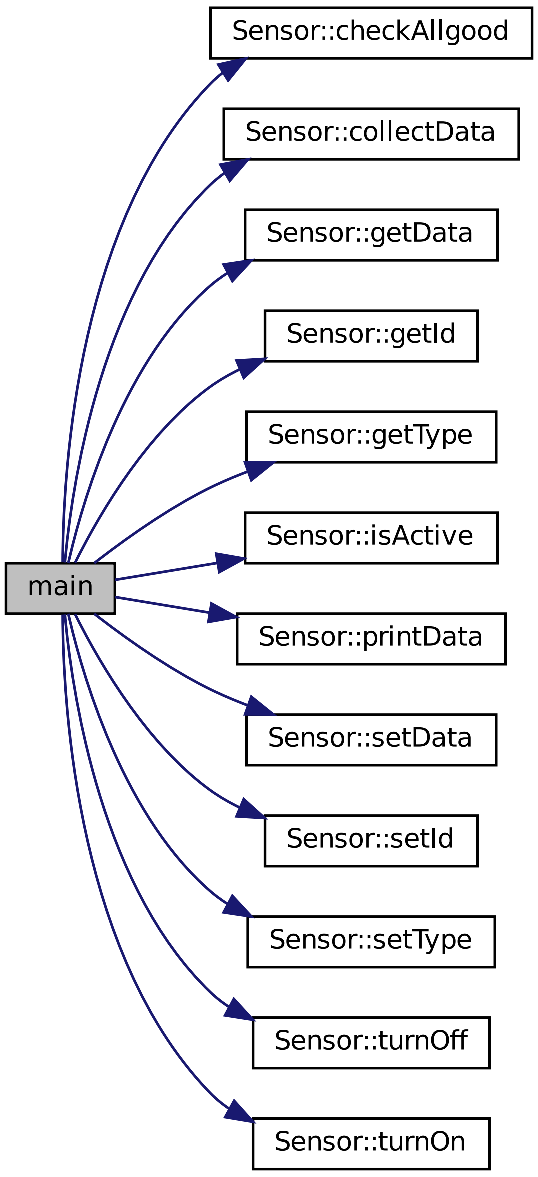
```
int main ( )
```

Definition at line 6 of file mainSensor.cpp.

```
           {
  // Creamos un sensor de cada tipo
  Sensor tempSensor(1, Sensor::Types::TEMPERATURE, true);
  Sensor airSensor(2, Sensor::Types::AIR_QUALITY, true);
   Sensor pressureSensor(5, Sensor::Types::PRESSURE, true);
   Sensor hSensor(6, Sensor::Types::HYDROMETER, true);
   Sensor lightSensor(3, Sensor::Types::LIGHT_SENSOR, true);
   Sensor phSensor(4, Sensor::Types::PH_SENSOR, true);

   // Ahora jugamos con los datos de los sensores
   tempSensor.collectData();
   cout « "Type: " « tempSensor.getType() « endl;
   // Ahora apagamos y encendemos
   tempSensor.turnOff();
   if (tempSensor.isActive()) {
     cout « "Sensor is active" « endl;
   } else {
     cout « "Sensor is inactive" « std::endl;
   }
   tempSensor.turnOn();
   if (tempSensor.isActive()) {
     cout « "Sensor is active" « endl;
   } else {
     cout « "Sensor is inactive" « std::endl;
   }
   // Asignamos un valor a la variable data_
   tempSensor.setData(25.5);
   cout « "Data: " « tempSensor.getData() « endl;
   // Cambiamos el id
   tempSensor.setId(10);
   cout « "ID: " « tempSensor.getId() « endl;
   // Cambiamos el tipo
   tempSensor.setType("AIR_QUALITY");
   cout « "Type: " « tempSensor.getType() « endl;

   // Print data
   tempSensor.printData();
   if (tempSensor.checkAllgood()) {
     cout « "All good!" « endl;
   } else {
     cout « "Not all good" « endl;
   }

   return 0;
 }
```

References Sensor::checkAllgood(), Sensor::collectData(), Sensor::getData(), Sensor::getId(), Sensor::getType(), Sensor::isActive(), Sensor::printData(), Sensor::setData(), Sensor::setId(), Sensor::setType(), Sensor::turnOff(), and Sensor::turnOn().

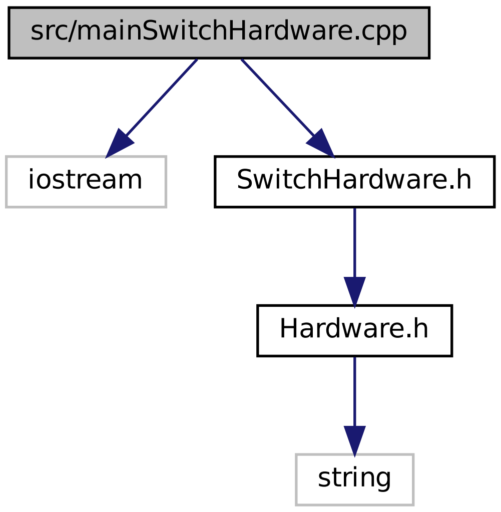
Here is the call graph for this function:



5.29 src/mainSwitchHardware.cpp File Reference

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

Include dependency graph for mainSwitchHardware.cpp:



Functions

- int main ()

5.29.1 Function Documentation

5.29.1.1 main()

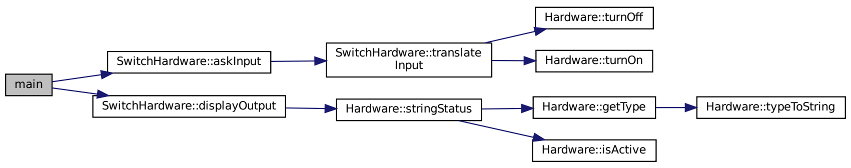
```
int main ( )
```

Definition at line 6 of file mainSwitchHardware.cpp.

```
            {
    // Generamos un swithc activado, luego preguntamos un input, displayamos el
    // output, y luego repetimos
    SwitchHardware sw(true);
     sw.displayOutput();
     sw.askInput();
     sw.displayOutput();
     sw.askInput();
     sw.displayOutput();
}
```

References SwitchHardware::askInput(), and SwitchHardware::displayOutput().

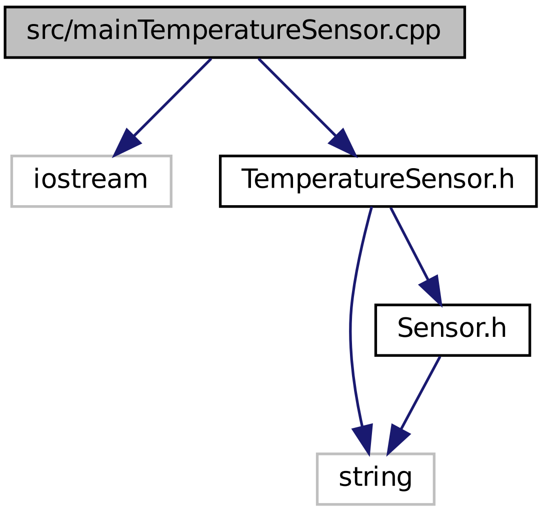
Here is the call graph for this function:



5.30 src/mainTemperatureSensor.cpp File Reference

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

Include dependency graph for mainTemperatureSensor.cpp:



Functions

- int main ()

5.30.1 Function Documentation

5.30.1.1 main()

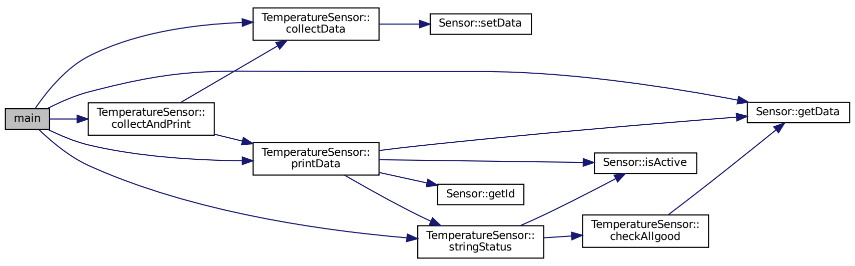
```
int main ( )
```

Definition at line 6 of file mainTemperatureSensor.cpp.

```
          {
 // Genero un sensor tipo temperatura
 TemperatureSensor tempSensor(1, true);
 // Imprimo la temperatura por defecto
  tempSensor.printData();
  // Cambio el valor de la temperatura
  tempSensor.collectData();
  // Imprimo la nueva temperatura
  tempSensor.printData();
  // Vuelvo a imprimir la temperatura
  cout « "Temperature: " « tempSensor.getData() « endl;
  cout « "Status: " « tempSensor.stringStatus() « endl;
  tempSensor.collectData();
  // Imprimo el sensor de nuevo
  tempSensor.printData();

  // Print collect and print
  tempSensor.collectAndPrint();
}
```

References TemperatureSensor::collectAndPrint(), TemperatureSensor::collectData(), Sensor::getData(), TemperatureSensor::printData(), and TemperatureSensor::stringStatus().

Here is the call graph for this function:

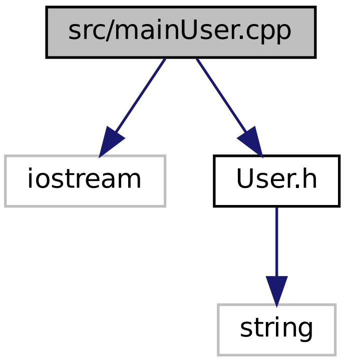


main
TemperatureSensor::collectData
Sensor::setData
TemperatureSensor::collectAndPrint
Sensor::getData
TemperatureSensor::printData
Sensor::isActive
Sensor::getId
TemperatureSensor::stringStatus
TemperatureSensor::checkAllgood

5.31 src/mainUser.cpp File Reference

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

Include dependency graph for mainUser.cpp:



Functions

- int main ()

5.31.1 Function Documentation

5.31.1.1 main()

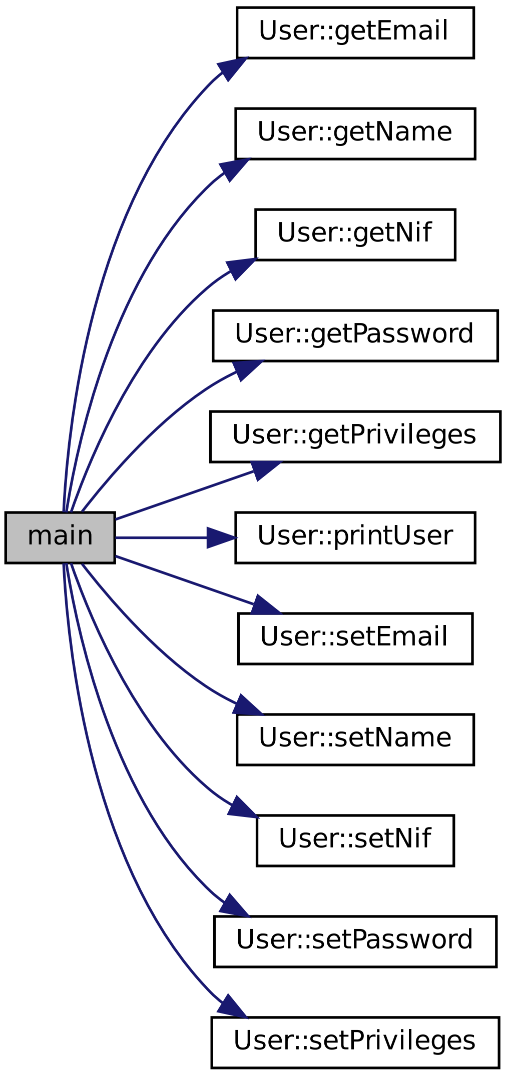
```
int main ( )
```

Definition at line 6 of file mainUser.cpp.

```
          {
 // Create a user with no name, NIF, password, privileges, and email
 // Values are assigned to the attributes using setters
 User user1;
  user1.setName("Adrian");
  user1.setNif("234453Y");
  user1.setPassword("ORANGE_JUICE");
  user1.setEmail("correoDeAdrian@potatoe.com");
  user1.setPrivileges("admin");
  // Print each attribute separately using getters
  cout « "Name: " « user1.getName() « endl;
  cout « "NIF: " « user1.getNif() « endl;
  cout « "Password: " « user1.getPassword() « endl;
  cout « "Privileges: " « user1.getPrivileges() « endl;
  cout « "Email: " « user1.getEmail() « endl;

  User user2("Lena", "LJ809K5ES43", "Bagumm.87¡", "guest",
             "liselese.ratte@aol.com");
  user2.printUser();

  // Compare users
  // If user 1 is greater than user 2 (privileges)
  if (user1.operator>(user2)) {
    cout « "The user " « user1.getName() « " has higher rank than "
         « user2.getName() « endl;
  } else {
    cout « "User " « user1.getName() « " has lower rank than "
         « user2.getName() « endl;
  }

  if (user1.operator<(user2)) {
    cout « "The user " « user1.getName() « " has lower rank than "
         « user2.getName() « endl;
  } else {
    cout « "User " « user1.getName() « " has higher rank than "
         « user2.getName() « endl;
  }

  // If user 1 is equal to user 2 (NIF)
  if (user1.operator==(user2)) {
    cout « "User " « user1.getName() « " is equal to user "
         « user2.getName() « endl;
  } else {
    cout « "User " « user1.getName() « " is NOT equal to user "
         « user2.getName() « endl;
  }
  // If user 1 is equal to user 1 (name, NIF, and password)
  if (user1.operator==(user1)) {
    cout « "User " « user1.getName() « " is equal to user "
         « user1.getName() « endl;
  } else {
    cout « "User " « user1.getName() « " is NOT equal to user "
         « user1.getName() « endl;
  }

  // Print a user using std::ostream
  User user3("Carlos", "010303403L", "1234_password", "employee",
             "carlos_sainz@gmail.com");
  cout « "PRINT USER WITH std::ostream" « endl;
  cout « user3 « endl;

  // Create a user using std::istream
  User user4;
  std::cout « "CREATE USER WITH std::istream" « endl;
  std::cout « "NAME, NIF, PASSWORD, PRIVILEGES, EMAIL" « endl;
  cin » user4;
  user4.printUser();
  cout « "PRINT USER WITH std::ostream" « endl;
  cout « user4 « endl;

  return 0;
}
```

References User::getEmail(), User::getName(), User::getNif(), User::getPassword(), User::getPrivileges(), User↵::printUser(), User::setEmail(), User::setName(), User::setNif(), User::setPassword(), and User::setPrivileges().

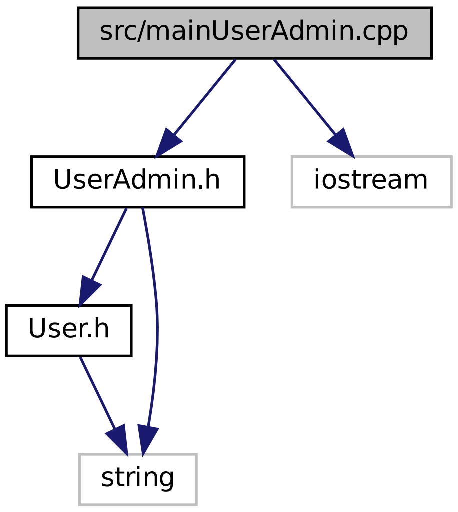
Here is the call graph for this function:



5.32 src/mainUserAdmin.cpp File Reference

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

Include dependency graph for mainUserAdmin.cpp:



Functions

- int main ()

5.32.1 Function Documentation

5.32.1.1 main()

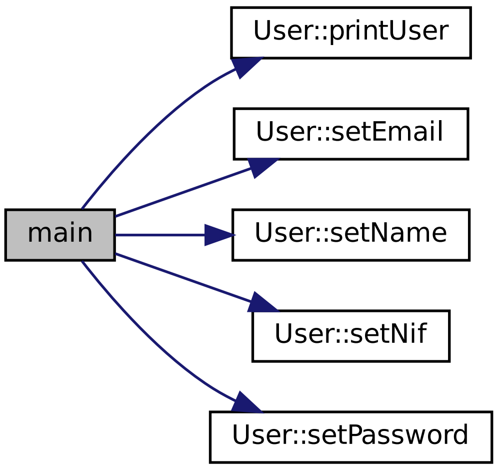
```
int main ( )
```

Definition at line 6 of file mainUserAdmin.cpp.

```
          {
 // Create a user with no name, NIF, password, privileges, and email
 // Values are assigned to the attributes using setters
 UserAdmin user1;
  user1.setName("Adrian");
  user1.setNif("234453Y");
  user1.setPassword("ORANGE_JUICE");
  user1.setEmail("adrian.adyra@gmail.com");
  // Imprimir el usuario
  user1.printUser();

  UserAdmin user2("Lena", "LJ809K5ES43", "Bagumm.87¡",
                 "liselese.ratte@aol.com");
  user2.printUser();
}
```

References User::printUser(), User::setEmail(), User::setName(), User::setNif(), and User::setPassword().

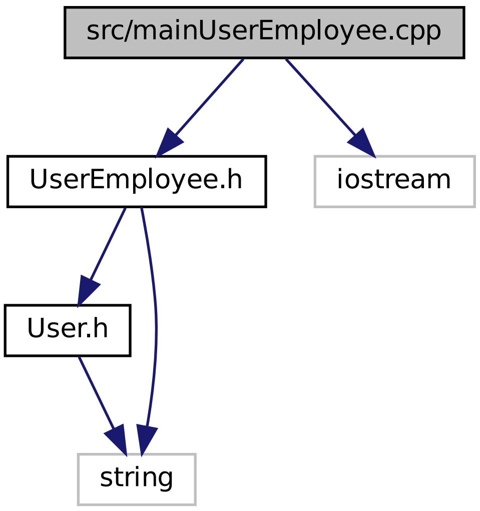
Here is the call graph for this function:



5.33 src/mainUserEmployee.cpp File Reference

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

Include dependency graph for mainUserEmployee.cpp:



Functions

- int main ()

5.33.1 Function Documentation

5.33.1.1 main()

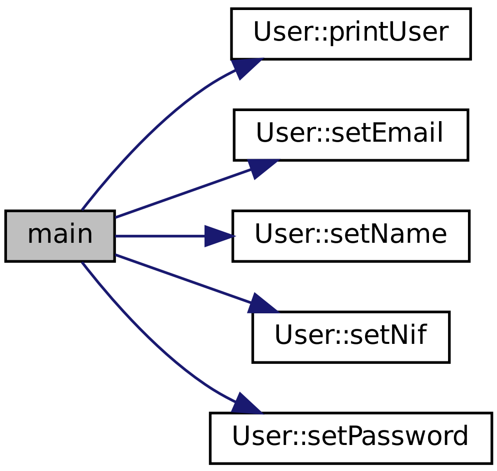
```
int main ( )
```

Definition at line 6 of file mainUserEmployee.cpp.

```
6            {
7   UserEmployee user1("Adrian", "234453Y", "ORANGE_JUICE", "correo");
8   user1.printUser();
9
10   UserEmployee user2;
11   user2.setName("Lena");
12   user2.setNif("LJ809K5ES43");
13   user2.setPassword("Bagumm.87¡");
14   user2.setEmail("correo2");
15   user2.printUser();
16
17   return 0;
18 }
```

References User::printUser(), User::setEmail(), User::setName(), User::setNif(), and User::setPassword().

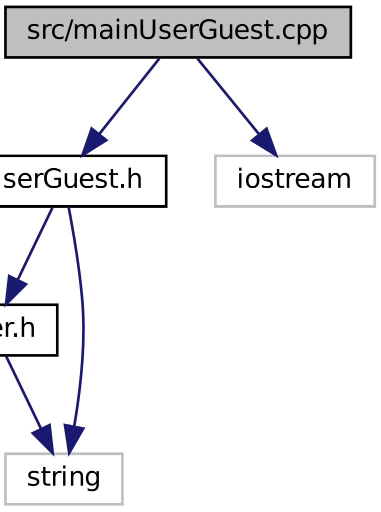
Here is the call graph for this function:



5.34 src/mainUserGuest.cpp File Reference

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

Include dependency graph for mainUserGuest.cpp:



Functions

- int main ()

5.34.1 Function Documentation

5.34.1.1 main()

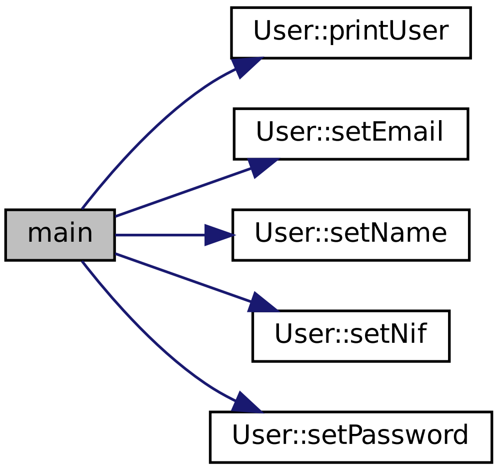
```
int main ( )
```

Definition at line 6 of file mainUserGuest.cpp.

```
         {
  UserGuest user1("Adrian", "234453Y", "ORANGE_JUICE", "correo");
  user1.printUser();

   UserGuest user2;
   user2.setName("Lena");
   user2.setNif("LJ809K5ES43");
   user2.setPassword("Bagumm.87¡");
   user2.setEmail("correo2");
   user2.printUser();

   return 0;
}
```

References User::printUser(), User::setEmail(), User::setName(), User::setNif(), and User::setPassword().

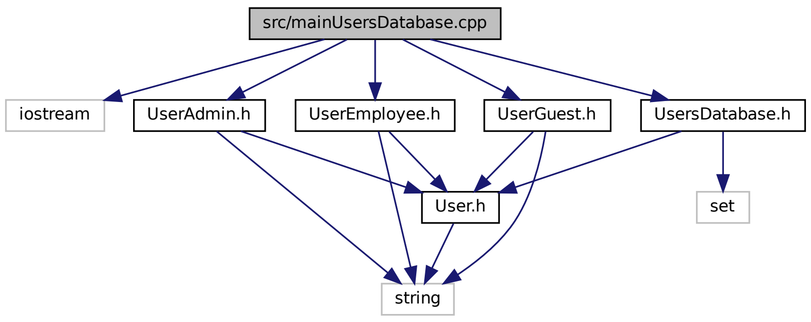
Here is the call graph for this function:



5.35 src/mainUsersDatabase.cpp File Reference

```
#include <iostream>
#include "UserAdmin.h"
#include "UserEmployee.h"
#include "UserGuest.h"
#include "UsersDatabase.h"
```

Include dependency graph for mainUsersDatabase.cpp:



Functions

- int main ()

5.35.1 Function Documentation

5.35.1.1 main()

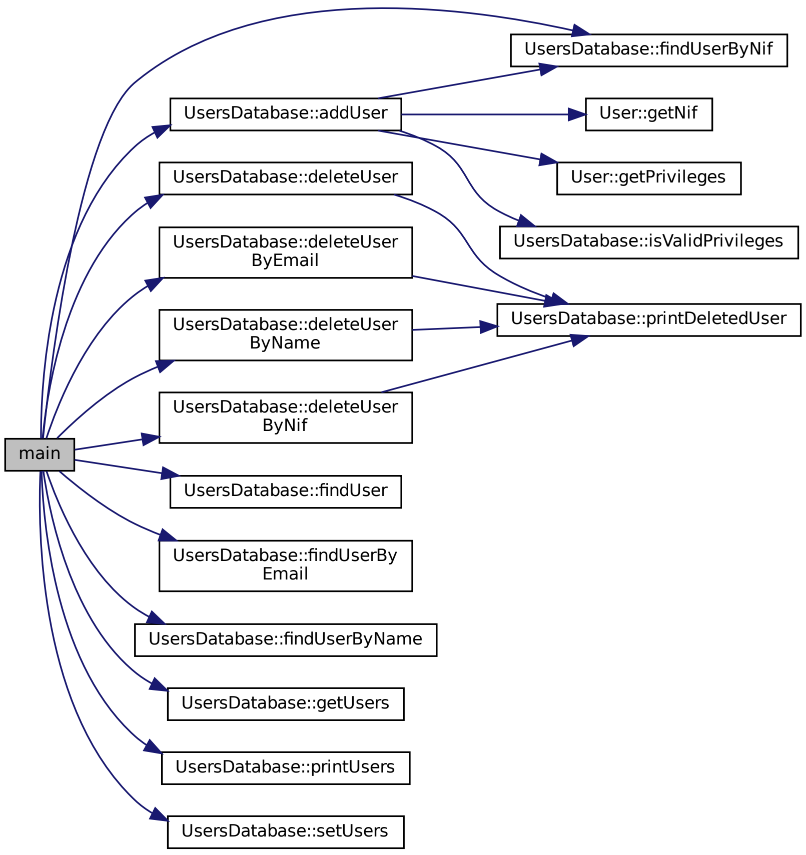
```
int main ( )
```

Definition at line 8 of file mainUsersDatabase.cpp.

```
          {
 // Crear una instancia de UsersDatabase
  UsersDatabase usersDatabase;

  // Agregar un alumno de cada tipo
  usersDatabase.addUser(
      new UserAdmin("Lena", "LJ809K5ES43", "Bagumm.87¡", "correoDeAdmin"));
  usersDatabase.addUser(
      new UserEmployee("PEPE", "NIFPEPE", "hhhhheh.rrrrr", "correoDeEmpleado"));
  usersDatabase.addUser(new UserGuest(
      "Juan", "slslslMINIFG", "sssdsdqadqwddddrrwwwrrr", "correoDeInvitado"));
  usersDatabase.addUser(new UserGuest(
      "Juan", "slslslMINIFG", "sssdsdqadqwddddrrwwwrrr", "correoDeInvitado"));
  usersDatabase.addUser(new UserGuest(
      "Sujeto -1", "-1", "sssdsdqadqwddddrrwwwrrr", "correoDeInvitado"));
  usersDatabase.addUser(new UserGuest(
      "Sujeto 1", "1", "sssdsdqadqwddddrrwwwrrr", "correoDeInvitado"));
  usersDatabase.addUser(new UserGuest(
      "Sujeto 2", "2", "sssdsdqadqwddddrrwwwrrr", "correoDeInvitado"));
  usersDatabase.addUser(
      new UserGuest("Sujeto 0", "0", "sssdsdqadqwddddrrwwwrrr", "correo0"));

  // Imprimir los usuarios
  usersDatabase.printUsers();

  // Obtener una copia de los usuarios
  UsersDatabase usersDatabaseCopy;
  std::set<const User *, UserPtrComparator> usersCopy =
      usersDatabase.getUsers();
  usersDatabaseCopy.setUsers(usersCopy);
  std::cout
      « "\n\n------------------------------------------------------\n\n\n"
      « std::endl;

  usersDatabaseCopy.printUsers();

  // Intentamos encontrar un usuario
  User *user = usersDatabaseCopy.findUser(
      UserAdmin("Lena", "LJ809K5ES43", "Bagumm.87¡", "correoDeAdmin"));
  if (user) {
    std::cout « "User found" « std::endl;
  } else {
    std::cout « "User not found" « std::endl;
  }

  // Intentamos encontrar un usuario por nombre
  user = usersDatabaseCopy.findUserByName("d");
  if (user) {
    std::cout « "User found" « std::endl;
  } else {
    std::cout « "User not found" « std::endl;
  }

  // Intentamos encontrar un usuario por NIF
  user = usersDatabaseCopy.findUserByNif("slslslMINIFG");
  if (user) {
    std::cout « "User found" « std::endl;
  } else {
    std::cout « "User not found" « std::endl;
  }

  // Intentamos encontrar un usuario por correo electrónico
  user = usersDatabaseCopy.findUserByEmail("correoDeInvitado");
  if (user) {
    std::cout « "User found" « std::endl;
  } else {
    std::cout « "User not found" « std::endl;
  }

  // Borrar un usuario
  usersDatabase.deleteUser(
      UserAdmin("Lena", "LJ809K5ES43", "Bagumm.87¡", "correoDeAdmin"));

  // Borrar por nombre
  usersDatabase.deleteUserByName("PEPE");

  // Borrar por NIF
  usersDatabase.deleteUserByNif("2");
  usersDatabase.deleteUserByNif("2");

  // Borrar por correo electrónico
  usersDatabase.deleteUserByEmail("correo0");

  std::cout
      « "\n\n-----------------------------------------------------\n\n\n"
      « std::endl;

  // Imprimir los usuarios
  usersDatabase.printUsers();

  return 0;
}
```

References UsersDatabase::addUser(), UsersDatabase::deleteUser(), UsersDatabase::deleteUserByEmail(), UsersDatabase::deleteUserByName(), UsersDatabase::deleteUserByNif(), UsersDatabase::findUser(), Users←Database::findUserByEmail(), UsersDatabase::findUserByName(), UsersDatabase::findUserByNif(), Users←Database::getUsers(), UsersDatabase::printUsers(), and UsersDatabase::setUsers().

Here is the call graph for this function:

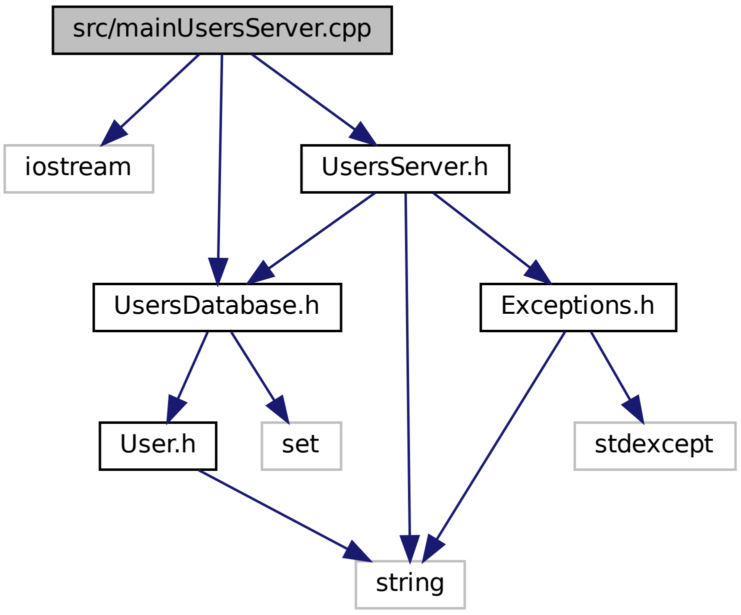

UsersDatabase::findUserByNif
UsersDatabase::addUser
User::getNif
UsersDatabase::deleteUser
User::getPrivileges
UsersDatabase::deleteUser ByEmail
UsersDatabase::isValidPrivileges
UsersDatabase::printDeletedUser
UsersDatabase::deleteUser ByName
UsersDatabase::deleteUser ByNif
main
UsersDatabase::findUser
UsersDatabase::findUserBy Email
UsersDatabase::findUserByName
UsersDatabase::getUsers
UsersDatabase::printUsers
UsersDatabase::setUsers

5.36 src/mainUsersServer.cpp File Reference

```
#include <iostream>
```

```
#include "UsersDatabase.h"
#include "UsersServer.h"
```

Include dependency graph for mainUsersServer.cpp:



Functions

- int main ()

5.36.1 Function Documentation

5.36.1.1 main()

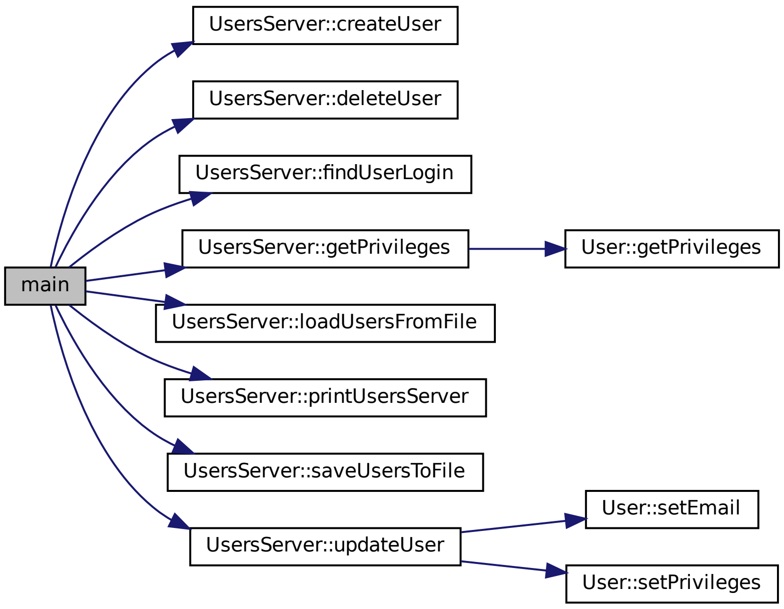
```
int main ( )
```

Definition at line 7 of file mainUsersServer.cpp.

```
            {
 // creamos un server de usuarios
 UsersServer usersServer;
  // Imprimimos usuarios
  usersServer.createUser("Lena", "LJ809K5ES43", "Bagumm.87¡", "admin",
                         "liselese.ratte@aol.com");
  usersServer.createUser("Lena", "LJ809K5ES43", "Bagumm.87¡", "guest",
                         "liselese.ratte@aol.com");
  cout « "*** Users created ***" « endl;
  usersServer.printUsersServer();
  cout « "---------------------" « endl;
  usersServer.deleteUser("LJ809K5ES43");
  usersServer.deleteUser("AJIJIKJIOKDDIJOIOJD");
  usersServer.loadUsersFromFile();

  usersServer.printUsersServer();
  cout « "----------------------" « endl;
  usersServer.createUser("adrian", "47552050X", "employee", "employee",
                         "adrian@example.com");
  usersServer.saveUsersToFile();
  if (usersServer.findUserLogin("adrian", "employee", "47552050X")) {
    cout « "User found" « endl;
  } else {
    cout « "User not found" « endl;
  }
  if (usersServer.findUserLogin("AAA", "employee", "47552050X")) {
    cout « "User found" « endl;
  } else {
    cout « "User not found" « endl;
  }
  cout « usersServer.getPrivileges("47552050X") « endl;
  cout « usersServer.getPrivileges("475ffX") « endl;
  usersServer.updateUser("adrian", "47552050X", "administrador", "admin",
                         "adrian3sAdminAhora");
  usersServer.printUsersServer();
  usersServer.saveUsersToFile();

  /*
  UsersServer usersServer2;
  usersServer2.loadUsersFromFile();
  std::cout « "*** Users loaded from file (SERVER 2) ***" « std::endl;
  usersServer2.printUsersServer();
  */
}
```

References UsersServer::createUser(), UsersServer::deleteUser(), UsersServer::findUserLogin(), UsersServer↩ ::getPrivileges(), UsersServer::loadUsersFromFile(), UsersServer::printUsersServer(), UsersServer::saveUsers↩ ToFile(), and UsersServer::updateUser().

Here is the call graph for this function:

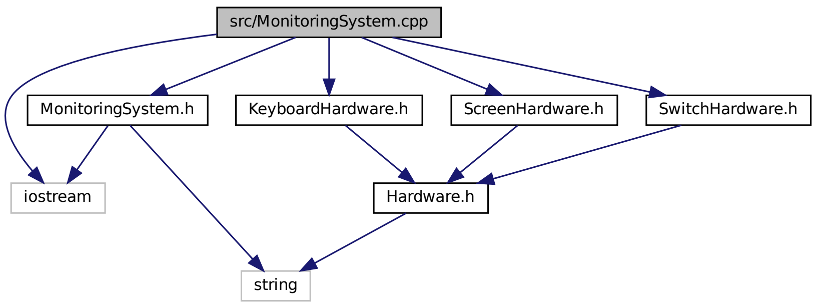

main
UsersServer::createUser
UsersServer::deleteUser
UsersServer::findUserLogin
UsersServer::getPrivileges
User::getPrivileges
UsersServer::loadUsersFromFile
UsersServer::printUsersServer
UsersServer::saveUsersToFile
UsersServer::updateUser
User::setEmail
User::setPrivileges

5.37 src/MonitoringSystem.cpp File Reference

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

```
#include "KeyboardHardware.h"
#include "ScreenHardware.h"
#include "SwitchHardware.h"
```

Include dependency graph for MonitoringSystem.cpp:



Variables

- const int MAIN_MENU_OPTIONS = 2
- const int ADMIN_MENU_OPTIONS = 12
- const int EMPLOYEE_MENU_OPTIONS = 7
- const int GUEST_MENU_OPTIONS = 4

5.37.1 Variable Documentation

5.37.1.1 ADMIN_MENU_OPTIONS

```
const int ADMIN_MENU_OPTIONS = 12
```

Definition at line 11 of file MonitoringSystem.cpp.

Referenced by MonitoringSystem::mainWindowAdmin().

5.37.1.2 EMPLOYEE_MENU_OPTIONS

```
const int EMPLOYEE_MENU_OPTIONS = 7
```

Definition at line 12 of file MonitoringSystem.cpp.

Referenced by MonitoringSystem::mainWindowEmployee().

5.37.1.3 GUEST_MENU_OPTIONS

```
const int GUEST_MENU_OPTIONS = 4
```

Definition at line 13 of file MonitoringSystem.cpp.

Referenced by MonitoringSystem::mainWindowGuest().

5.37.1.4 MAIN_MENU_OPTIONS

```
const int MAIN_MENU_OPTIONS = 2
```

Definition at line 10 of file MonitoringSystem.cpp.

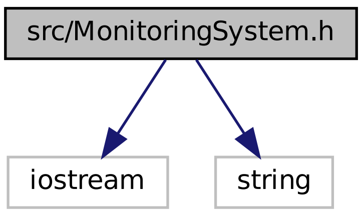
Referenced by MonitoringSystem::initialScreen().

5.38 src/MonitoringSystem.h File Reference

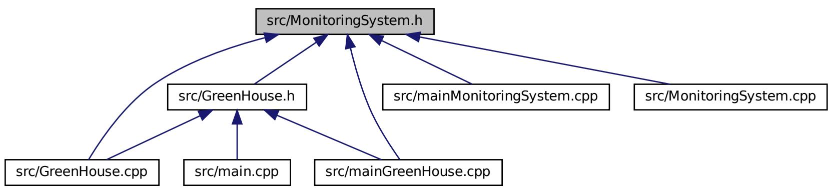
This is the class MonitoringSystem. It contains the attributes and methods of the MonitoringSystem class, this class.

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

Include dependency graph for MonitoringSystem.h:



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



Classes

- class MonitoringSystem

Variables

- const int MAIN_MENU_OPTIONS
- const int ADMIN_MENU_OPTIONS
- const int EMPLOYEE_MENU_OPTIONS
- const int GUEST_MENU_OPTIONS

5.38.1 Detailed Description

This is the class MonitoringSystem. It contains the attributes and methods of the MonitoringSystem class, this class.

Author

Adrián Montes Linares

Date

21/04/2024

5.38.2 Variable Documentation

5.38.2.1 ADMIN_MENU_OPTIONS

```
const int ADMIN_MENU_OPTIONS [extern]
```

Definition at line 11 of file MonitoringSystem.cpp.

Referenced by MonitoringSystem::mainWindowAdmin().

5.38.2.2 EMPLOYEE_MENU_OPTIONS

```
const int EMPLOYEE_MENU_OPTIONS [extern]
```

Definition at line 12 of file MonitoringSystem.cpp.

Referenced by MonitoringSystem::mainWindowEmployee().

5.38.2.3 GUEST_MENU_OPTIONS

```
const int GUEST_MENU_OPTIONS  [extern]
```

Definition at line 13 of file MonitoringSystem.cpp.

Referenced by MonitoringSystem::mainWindowGuest().

5.38.2.4 MAIN_MENU_OPTIONS

```
const int MAIN_MENU_OPTIONS  [extern]
```

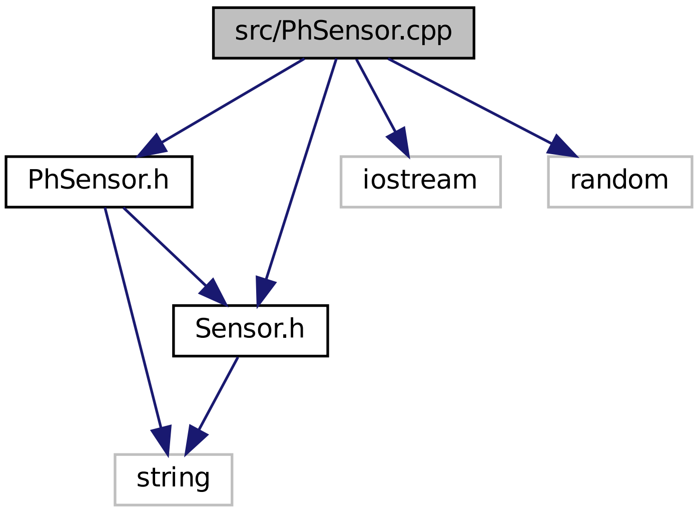
Definition at line 10 of file MonitoringSystem.cpp.

Referenced by MonitoringSystem::initialScreen().

5.39 src/PhSensor.cpp File Reference

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

Include dependency graph for PhSensor.cpp:



Functions

- std::ostream & operator<< (std::ostream &os, const PhSensor &sensor)

5.39.1 Function Documentation

5.39.1.1 operator<<()

```
std::ostream& operator<< (
              std::ostream & os,
              const PhSensor & sensor )
```

Definition at line 55 of file PhSensor.cpp.

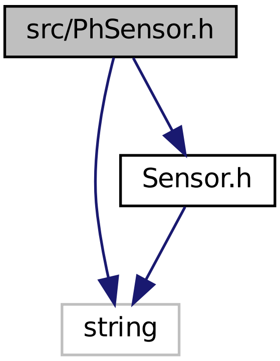
```
                                                                {
    sensor.printData();
    return os;
}
```

5.40 src/PhSensor.h File Reference

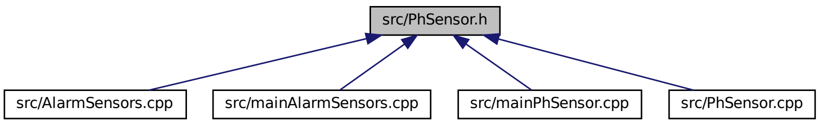
This is the class PhSensor. It contains the attributes and methods of the PhSensor class.

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

Include dependency graph for PhSensor.h:



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



Classes

- class PhSensor

5.40.1 Detailed Description

This is the class PhSensor. It contains the attributes and methods of the PhSensor class.

Author

Adrián Montes Linares

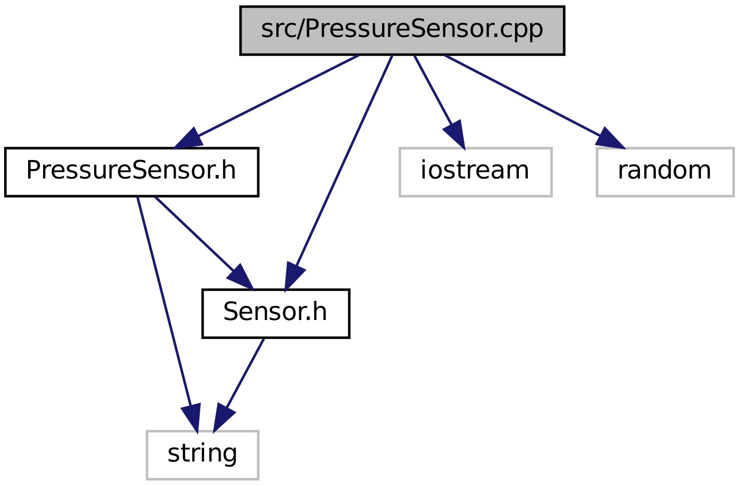
Date

21/04/2024

5.41 src/PressureSensor.cpp File Reference

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

Include dependency graph for PressureSensor.cpp:



Functions

- std::ostream & operator<< (std::ostream &os, const PressureSensor &sensor)

5.41.1 Function Documentation

5.41.1.1 operator<<()

```
std::ostream& operator<< (
              std::ostream & os,
              const PressureSensor & sensor )
```

Definition at line 32 of file PressureSensor.cpp.

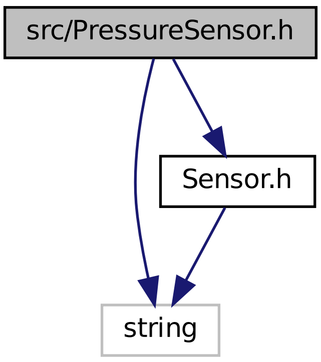
```
                                                              {
   sensor.printData();
   return os;
 }
```

5.42 src/PressureSensor.h File Reference

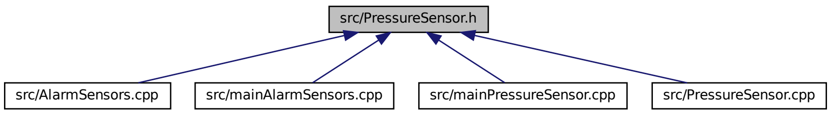
This is the class PressureSensor. It contains the attributes and methods of the PressureSensor class.

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

Include dependency graph for PressureSensor.h:



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



Classes

- class PressureSensor

5.42.1 Detailed Description

This is the class PressureSensor. It contains the attributes and methods of the PressureSensor class.

Author

Adrián Montes Linares

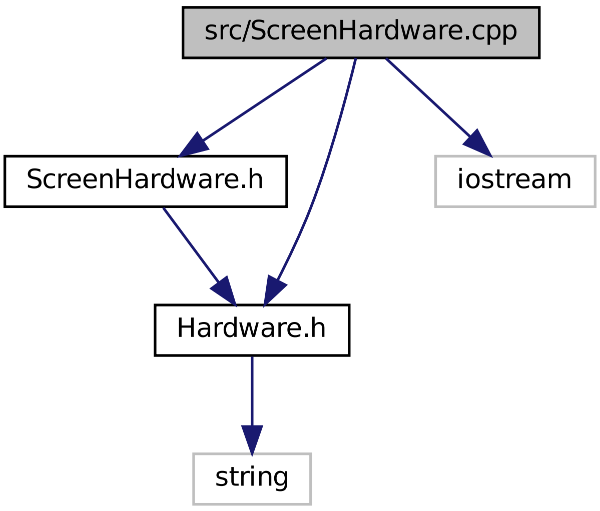
Date

21/04/2024

5.43 src/ScreenHardware.cpp File Reference

```
#include "ScreenHardware.h"
#include <iostream>
#include "Hardware.h"
```

Include dependency graph for ScreenHardware.cpp:



Variables

- const std::string USER_PROMPT
- const std::string ASK_DATA = "Please enter all the data required"

5.43.1 Variable Documentation

5.43.1.1 ASK_DATA

```
const std::string ASK_DATA = "Please enter all the data required"
```

Definition at line 13 of file ScreenHardware.cpp.

Referenced by ScreenHardware::createUserWindow(), ScreenHardware::loginWindow(), and ScreenHardware↩ ::updateUserWindow().

5.43.1.2 USER_PROMPT

```
const std::string USER_PROMPT
```

Initial value:

```
=
    "First the name(intro), then the new password(intro), then the "
    "nif(intro), then the new role(intro), then the status(intro), "
    "then the email(intro)"
```

Definition at line 8 of file ScreenHardware.cpp.

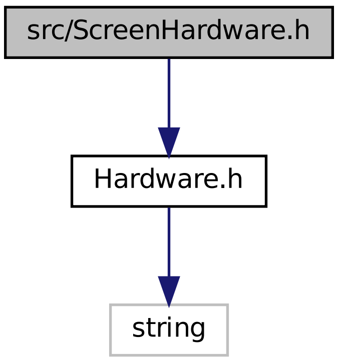
Referenced by ScreenHardware::createUserWindow(), and ScreenHardware::updateUserWindow().

5.44 src/ScreenHardware.h File Reference

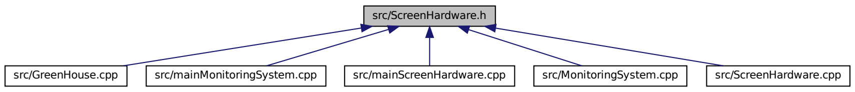
This is the class ScreenHardware. It contains the attributes and methods of the ScreenHardware class, this class is a child of the Hardware class. This class is used to display the output of the system and ask for an input before with the keyboard.

```
#include "Hardware.h"
```

Include dependency graph for ScreenHardware.h:



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



Classes

- class ScreenHardware

5.44.1 Detailed Description

This is the class ScreenHardware. It contains the attributes and methods of the ScreenHardware class, this class is a child of the Hardware class. This class is used to display the output of the system and ask for an input before with the keyboard.

Author

Adrián Montes Linares

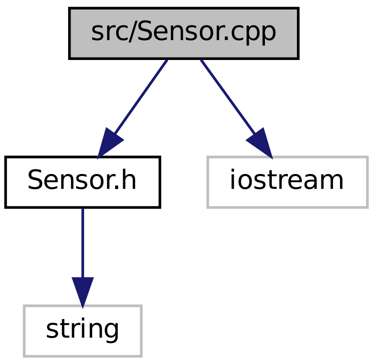
Date

21/04/2024

5.45 src/Sensor.cpp File Reference

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

Include dependency graph for Sensor.cpp:



Functions

- std::ostream & operator<< (std::ostream &os, const Sensor &Sensor)
- std::istream & operator>> (std::istream &is, Sensor &sensor)

5.45.1 Function Documentation

5.45.1.1 operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const Sensor & Sensor )
```

Parameters

os	
Sensor	

Returns

std::ostream&

Definition at line 102 of file Sensor.cpp.

```
                                                         {
  os « "ID: " « Sensor.getId() « " Type: " « Sensor.getType()
     « " Active: " « Sensor.isActive() « " Data: " « Sensor.getData()
     « std::endl;
  return os;
}
```

5.45.1.2 operator>>()

```
std::istream& operator>> (
            std::istream & is,
            Sensor & sensor )
```

Parameters

is	
Sensor	

Returns

std::istream&

Definition at line 109 of file Sensor.cpp.

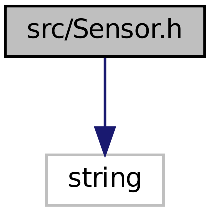
```
{
  cout « "Enter sensor ID: ";
  is » sensor.id_;
  cout « "Enter the type: ";
  std::string type;
  is » type;
  sensor.setType(type);
  cout « "Enter sensor active: ";
  is » sensor.active_;

  return is;
}
```

5.46 src/Sensor.h File Reference

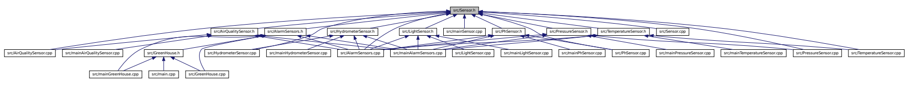
This is the class Sensor. It contains the attributes and methods of the Sensor class.

`#include <string>`

Include dependency graph for Sensor.h:



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



Classes

- class Sensor

5.46.1 Detailed Description

This is the class Sensor. It contains the attributes and methods of the Sensor class.

Author

Adrián Montes Linares

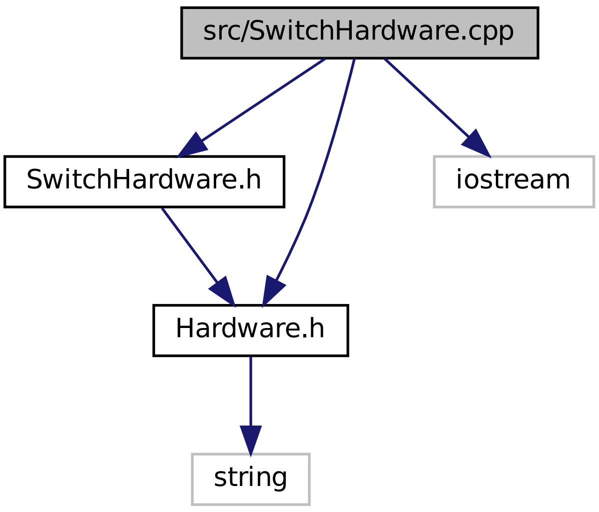
Date

21/04/2024

5.47 src/SwitchHardware.cpp File Reference

```
#include "SwitchHardware.h"
#include <iostream>
#include "Hardware.h"
```

Include dependency graph for SwitchHardware.cpp:

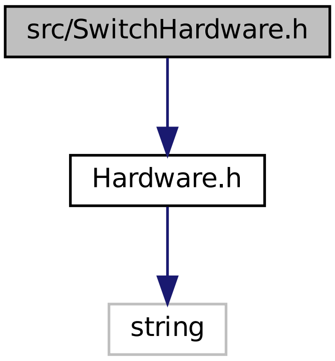


5.48 src/SwitchHardware.h File Reference

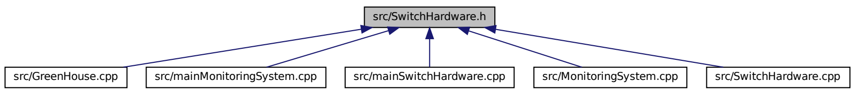
This is the class SwitchHardware. It contains the attributes and methods of the SwitchHardware class, this class is a child of the Hardware class.

`#include "Hardware.h"`

Include dependency graph for SwitchHardware.h:



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



Classes

- class SwitchHardware

5.48.1 Detailed Description

This is the class SwitchHardware. It contains the attributes and methods of the SwitchHardware class, this class is a child of the Hardware class.

Author

Adrián Montes Linares

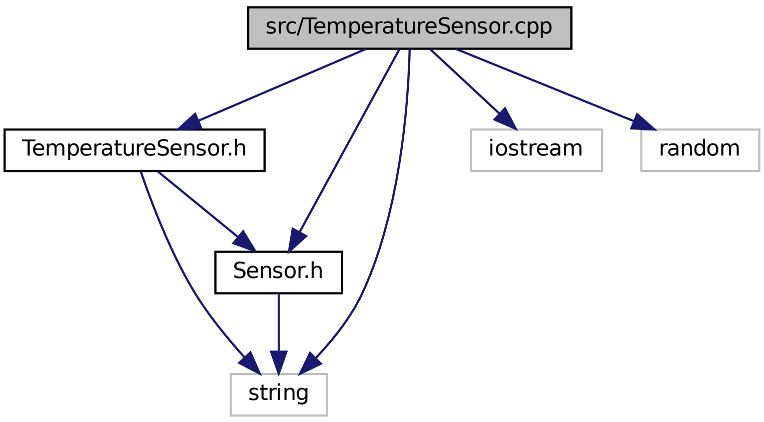
Date

21/04/2024

5.49 src/TemperatureSensor.cpp File Reference

```
#include "TemperatureSensor.h"
#include <iostream>
#include <random>
#include <string>
#include "Sensor.h"
```

Include dependency graph for TemperatureSensor.cpp:



Functions

- std::ostream & operator<< (std::ostream &os, const TemperatureSensor &sensor)

5.49.1 Function Documentation

5.49.1.1 operator<<()

```
std::ostream& operator<< (
             std::ostream & os,
             const TemperatureSensor & sensor )
```

Definition at line 36 of file TemperatureSensor.cpp.

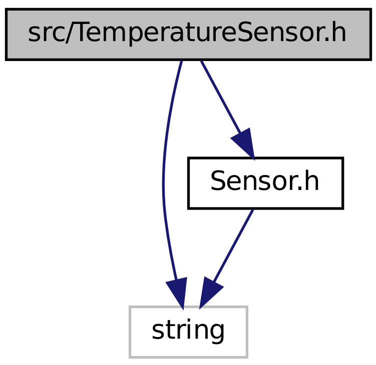
```
                                                                  {
    sensor.printData();
    return os;
}
```

5.50 src/TemperatureSensor.h File Reference

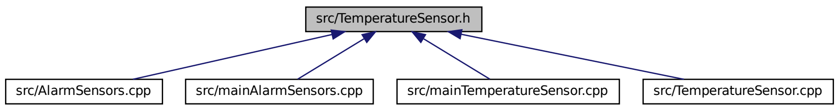
This is the class TemperatureSensor. It contains the attributes and methods of the TemperatureSensor class.

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

Include dependency graph for TemperatureSensor.h:



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



Classes

- class TemperatureSensor

5.50.1 Detailed Description

This is the class TemperatureSensor. It contains the attributes and methods of the TemperatureSensor class.

Author

Adrián Montes Linares

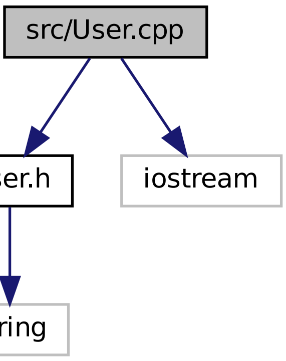
Date

21/04/2024

5.51 src/User.cpp File Reference

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

Include dependency graph for User.cpp:



Functions

- std::ostream & operator<< (std::ostream &os, const User &user)
- std::istream & operator>> (std::istream &is, User &user)

5.51.1 Function Documentation

5.51.1.1 operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const User & user )
```

Parameters

os	
user	

Returns

std::ostream&

Definition at line 76 of file User.cpp.

```
                                                        {
    os « user.getName() « " " « user.getNif() « " " « user.getPassword()
       « " " « user.getPrivileges() « " " « user.getEmail() « std::endl;
    return os;
}
```

5.51.1.2 operator>>()

```
std::istream& operator>> (
              std::istream & is,
              User & user )
```

Parameters

is	
user	

Returns

std::istream&

Definition at line 83 of file User.cpp.

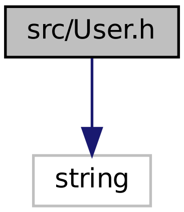
```
                                                  {
    std::string privilege;
    is » user.name » user.nif » user.password » privilege » user.email;
    user.setPrivileges(privilege);
    return is;
}
```

5.52 src/User.h File Reference

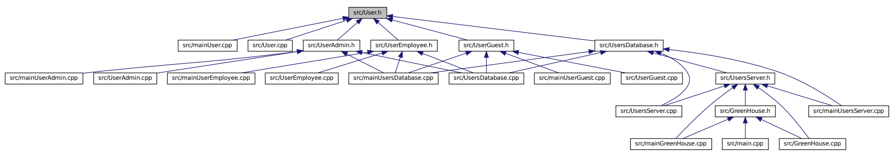
This is the class User. It contains the attributes and methods of the User class.

```
#include <string>
```

Include dependency graph for User.h:



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



Classes

- class User

5.52.1 Detailed Description

This is the class User. It contains the attributes and methods of the User class.

Author

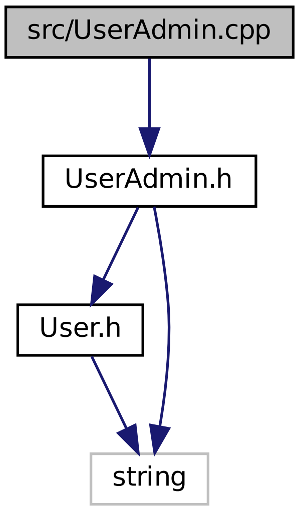
Adrián Montes Linares

Date

21/04/2024

5.53 src/UserAdmin.cpp File Reference

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

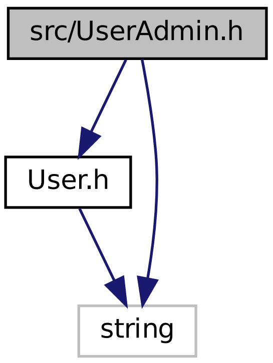


5.54 src/UserAdmin.h File Reference

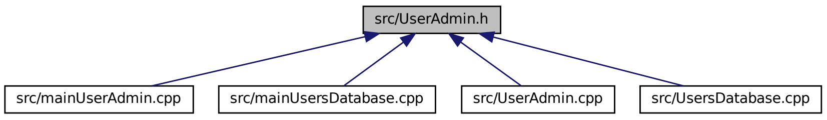
This is the class UserAdmin. It contains the attributes and methods of the UserAdmin class.

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

Include dependency graph for UserAdmin.h:



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



Classes

- class UserAdmin

5.54.1 Detailed Description

This is the class UserAdmin. It contains the attributes and methods of the UserAdmin class.

Author

Adrián Montes Linares

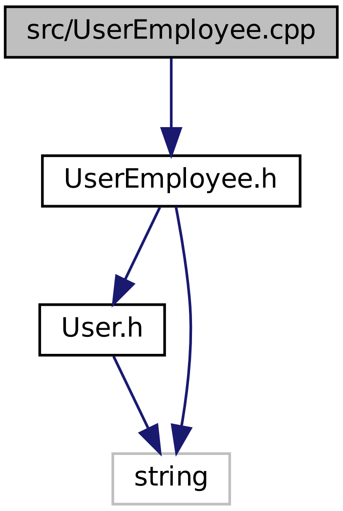
Date

21/04/2024

5.55 src/UserEmployee.cpp File Reference

```
#include "UserEmployee.h"
```

Include dependency graph for UserEmployee.cpp:

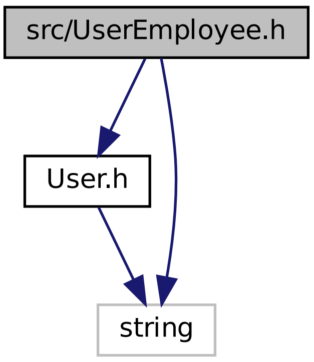


5.56 src/UserEmployee.h File Reference

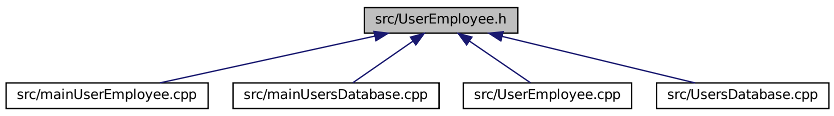
This is the class UserEmployee. It contains the attributes and methods of the UserEmployee class.

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

Include dependency graph for UserEmployee.h:



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



Classes

- class UserEmployee

5.56.1 Detailed Description

This is the class UserEmployee. It contains the attributes and methods of the UserEmployee class.

Author

Adrián Montes Linares

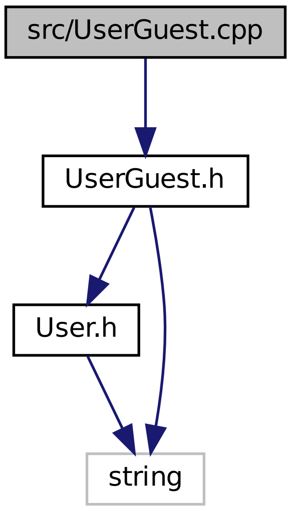
Date

21/04/2024

5.57 src/UserGuest.cpp File Reference

```
#include "UserGuest.h"
```

Include dependency graph for UserGuest.cpp:

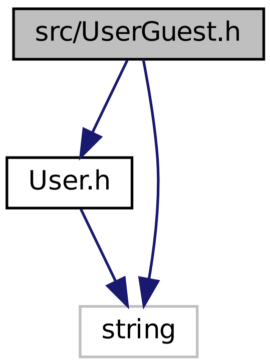


5.58 src/UserGuest.h File Reference

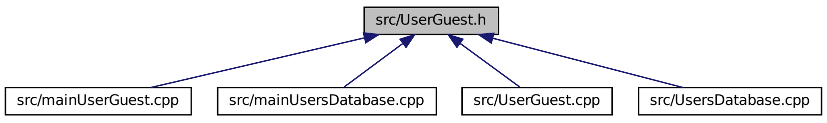
This is the class UserGuest. It contains the attributes and methods of the UserGuest class.

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

Include dependency graph for UserGuest.h:



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



Classes

- class UserGuest

5.58.1 Detailed Description

This is the class UserGuest. It contains the attributes and methods of the UserGuest class.

Author

Adrián Montes Linares

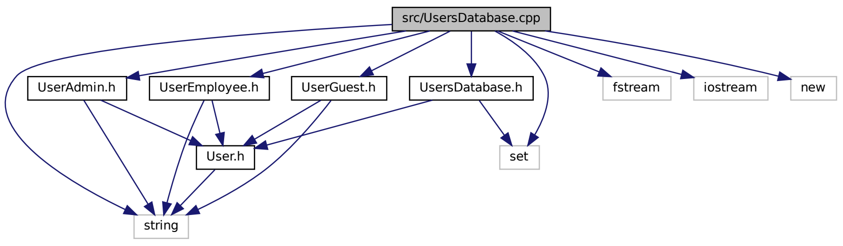
Date

21/04/2024

5.59 src/UsersDatabase.cpp File Reference

```
#include "UsersDatabase.h"
#include <fstream>
#include <iostream>
#include <new>
#include <set>
#include <string>
#include "UserAdmin.h"
#include "UserEmployee.h"
#include "UserGuest.h"
```

Include dependency graph for UsersDatabase.cpp:

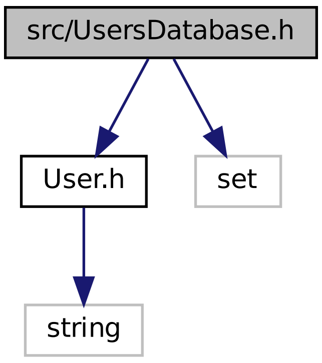


5.60 src/UsersDatabase.h File Reference

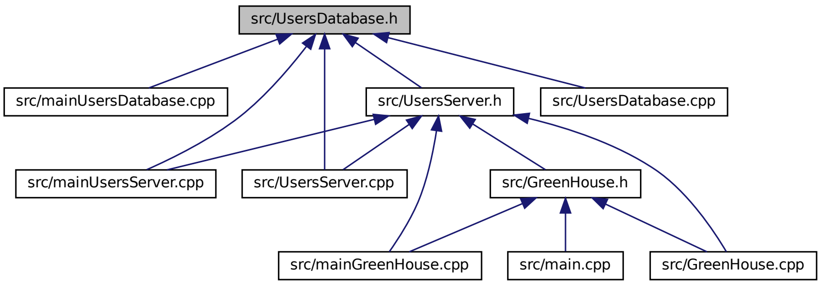
This is the class UsersDatabase. It contains the attributes and methods of the UsersDatabase class.

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

Include dependency graph for UsersDatabase.h:



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



Classes

- class UserPtrComparator
- class UserNameComparator
- class UsersDatabase

5.60.1 Detailed Description

This is the class UsersDatabase. It contains the attributes and methods of the UsersDatabase class.

Author

Adrián Montes Linares

Date

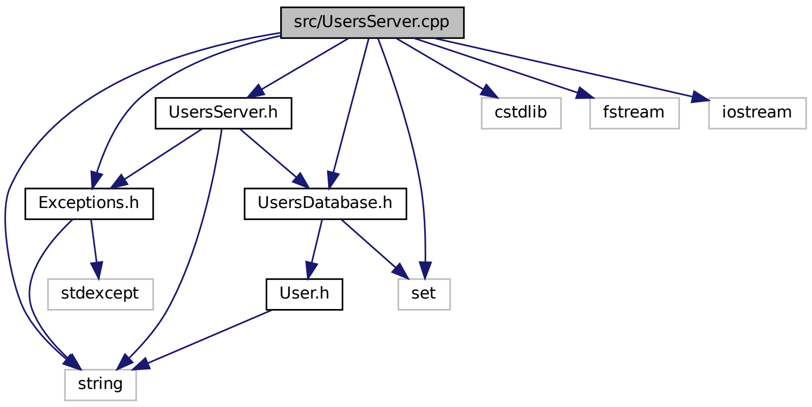
21/04/2024

5.61 src/UsersServer.cpp File Reference

```
#include "UsersServer.h"
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <string>
#include <set>
#include "Exceptions.h"
```

```
#include "UsersDatabase.h"
```

Include dependency graph for UsersServer.cpp:

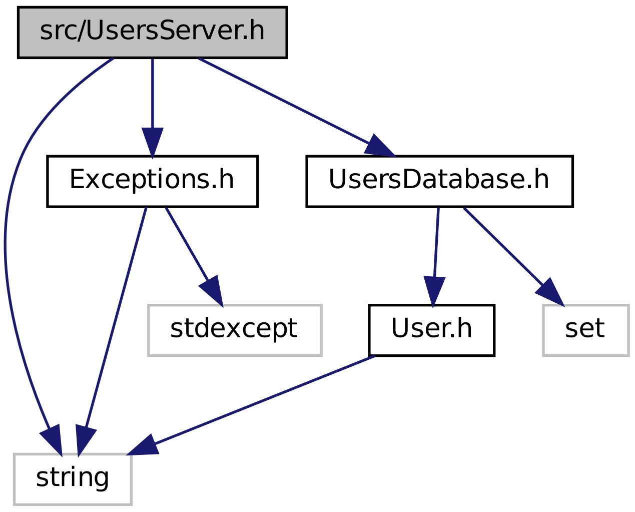


5.62 src/UsersServer.h File Reference

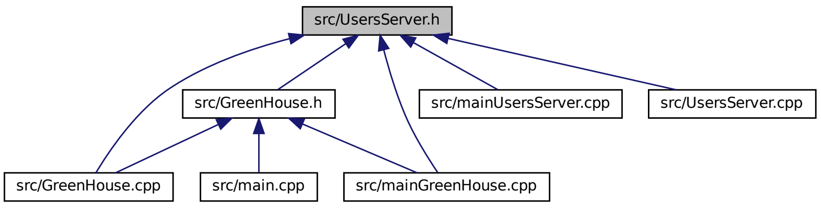
This is the class UsersServer. It contains the attributes and methods of the UsersServer class.

```
#include <string>
#include "Exceptions.h"
#include "UsersDatabase.h"
```

Include dependency graph for UsersServer.h:



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



Classes

- class UsersServer

5.62.1 Detailed Description

This is the class UsersServer. It contains the attributes and methods of the UsersServer class.

Author

Adrián Montes Linares

Date

21/04/2024