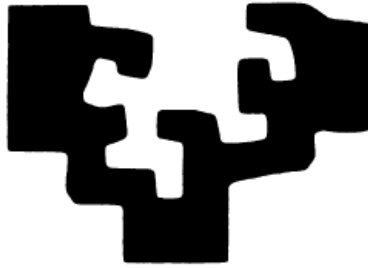


eman ta zabal zazu



universidad
del país vasco

euskal herriko
unibertsitatea

Facultad de Informática

Informatika Fakultatea

TITULACIÓN:
Ingeniería Informática

Home-Automation System

Alumno: Joseba Nevado Rodríguez

Director: Alejandro García-Alonso Montoya

Proyecto Fin de Carrera, 10 de julio de 2012

© 2012 Joseba Nevado

Special Thanks

I want to thank Professor Alex García-Alonso for his support and leadership during this project, as well as Bassem Barake for being a perfect technical and motivation supporter and Professor Abdulmotaleb ElSaddik for making this experience possible.

RESUMEN

Nuestro proyecto trata de obtener un sistema centralizado para el manejo y monitorización de un conjunto de aparatos eléctricos del hogar, tales como luces, cámaras, sensores, y otros aparatos mediante una aplicación para dispositivos móviles Android. También se facilita una herramienta web para administrar dichos aparatos.

Nuestro sistema da soporte al manejo individual y manual de estos aparatos, por ejemplo encender una lámpara, así como la posibilidad de incluir programas que se ejecuten a una determinada hora, y con una determinada recurrencia. Por ejemplo, se puede programar el encendido de una luz todos los días a las 20:00. Además, también se pueden crear reglas que, dado una determinada situación, realicen una serie de acciones. Por ejemplo, si se detecta que la luminosidad baja de cierto valor (detectado mediante un sensor de luz), se encienda una lámpara.

Para conseguir esto se ha realizado una aplicación centralizada, a modo de servidor, que se comunica con los aparatos eléctricos mediante diferentes protocolos de comunicación, así como con el dispositivo móvil Android, y una aplicación web administrativa, a través de un portal de servicios web.

ABSTRACT

This Project's objective is to provide a centralized system from which a user can control and manage home electrical devices such as lamps, appliances, cameras or sensors. To provide these functionalities the system includes a centralized server, an Android tablet application, and an administration web application.

The system allows the user to control the devices directly by sending commands to them, e.g. turn on a lamp. The system also offers the user the ability to create automatic behaviors for the devices. These behaviors can be either schedules or rules. Schedules run automatically every certain time, e.g. every day at 11PM, while rules run when a series of conditions are met. These conditions can be either time or device status, including sensor readings.

The centralized server connects the electric devices to the PC, and offers an interface over the Internet via Web Services. This Web Service is used both by the web application and the Android application to manage the system remotely.

INDEX

1	INTRODUCTION	1
1.1	Motivation.....	1
1.2	Software Used.....	2
1.2.1	Visual Studio 2010.....	2
1.2.2	Eclipse and Android SDK.....	2
1.2.3	Crucible.....	2
1.3	Memory Structure	3
2	PROJECT'S OBJECTIVES DOCUMENT	5
2.1	Tasks List.....	5
2.2	Gantt Chart.....	5
3	SYSTEM DESCRIPTION.....	7
3.1	Terminology and generic description	7
3.1.1	Schedule.....	8
3.1.2	Rule.....	8
3.1.3	Scene.....	8
3.2	Centralized Unit.....	9
3.2.1	Objectives	9
3.2.2	Design	9
3.2.2.1	Centralized class	9
3.2.2.2	Device Management class	10
3.2.2.3	Location Management class.....	10
3.2.2.4	Schedule Management class	10
3.2.2.5	Rule Management class	10
3.2.2.6	Scene Management class	10
3.2.2.7	Database Controller classes	10
3.2.2.8	IDispatcher interface.....	10
3.3	Web application	11
3.3.1	Objectives	11
3.3.2	Design	11
3.3.2.1	Home Page.....	11

3.3.2.2	Device Management.....	11
3.3.2.3	Add Device.....	12
3.3.2.4	Edit Device.....	12
3.3.2.5	Add Location.....	12
3.3.2.6	Schedule Management	12
3.3.2.7	Add Schedule	12
3.3.2.8	Edit Schedule.....	12
3.3.2.9	Rule Management	12
3.3.2.10	Add Rule	12
3.3.2.11	Edit Rule.....	12
3.4	Android application.....	13
3.4.1	Objectives.....	13
3.4.2	Design.....	13
3.4.2.1	Welcome.....	13
3.4.2.2	Control.....	14
3.4.2.3	Schedule Management	14
3.4.2.4	Add Schedule	14
3.4.2.5	Rule Management	14
3.4.2.6	Add Rule	14
3.4.2.7	Security.....	14
3.4.2.8	Scenes.....	14
4	CENTRALIZED UNIT AND WEB SERVICE.....	15
4.1	Web Service	15
4.2	Design and Main classes	18
4.2.1	Centralized	19
4.2.2	Device Management.....	19
4.2.3	Schedule Management	20
4.2.4	Rule Management	20
4.2.5	Location Management.....	20
4.2.6	Scene Management	20
4.2.7	Security Camera Management (not implemented).....	20
4.2.8	IDispatcher	21

4.2.8.1	X10.....	21
4.2.8.2	Insteon.....	21
4.2.8.3	ZWave.....	21
4.2.9	SQLConnection – SQLQuery	22
4.3	Example of execution sequence.....	22
4.4	Database.....	24
5	WEB APPLICATION	27
5.1	Design.....	27
5.2	Main pages.....	27
5.2.1	Device Management	28
5.2.2	Add Device	29
5.2.3	Edit Device	30
5.2.4	Add Location	31
5.2.5	Schedule Management.....	32
5.2.6	Add Schedule.....	33
5.2.7	Edit Schedule	34
5.2.8	Rule Management.....	35
5.2.9	Add Rule	36
5.2.10	Edit Rule	37
5.3	Connectivity with the Web Service	38
6	ANDROID APPLICATION: USER INTERFACE	39
6.1	The layout system	39
6.2	Activity List	41
6.2.1	HomeAutomation Activity	41
6.2.2	Control	42
6.2.3	Rule Management.....	44
6.2.4	Add Rule	46
6.2.5	Schedule Management.....	48
6.2.6	Add Schedule.....	50
6.2.7	Security	52
6.2.8	Scenes	53
7	ANDROID APPLICATION: CLASSES AND CONNECTIVITY	55

7.1	Classes.....	55
7.2	Connectivity.....	55
7.2.1	Web Service methods used.....	56
7.2.2	Calling the Web Service methods.....	57
8	CONCLUSIONS.....	59
9	BIBLIOGRAPHY.....	61

Annex I	Project Environment
Annex II	Android Application's Owner's Manual
Annex III	Home Automation Protocols

FIGURE INDEX

Figure 2.1: Gantt Chart task list	5
Figure 2.2: Gantt chart	6
Figure 3.1: System overview	7
Figure 3.2: Centralized unit major class design	9
Figure 3.3: Web application design	11
Figure 3.4: Android app activity design	13
Figure 4.1: Centralized Unit modules	19
Figure 4.2: Execution sequence for edit schedule	23
Figure 4.3: Database design	25
Figure 5.1: Top menu	27
Figure 5.2: Device Management page	28
Figure 5.3: Add Device page	29
Figure 5.4: Edit Device page	30
Figure 5.5: Add Location page	31
Figure 5.6: Schedule Management page	32
Figure 5.7: Add Schedule page	33
Figure 5.8: Edit Schedule page	34
Figure 5.9: Rule Management page	35
Figure 5.10: Add Rule page	36
Figure 5.11: Edit Rule page	37
Figure 6.1: Layout system	39
Figure 6.2: Android activity list	41
Figure 6.3: Control activity	42
Figure 6.4: All the locations	43
Figure 6.5: Rule Management activity	44
Figure 6.6: Rule details	45
Figure 6.7: Add a Rule	46
Figure 6.8: Edit a Rule	47
Figure 6.9: Schedule Management	48
Figure 6.10: Schedule details	49
Figure 6.11: Add a Schedule	50
Figure 6.12: Edit a Schedule	51
Figure 6.13: Security activity	52
Figure 6.14: Scenes activity	53

1 INTRODUCTION

This Graduation Project (Proyecto Fin de Carrera) has been developed at Discover Lab (University of Ottawa). The project develops a home automation system to manage and control different devices.

After arriving at Ottawa I presented my experience to the laboratory staff. One of them, Bassem Barake, proposed to be my tutor for an Avaya Company project. He provided a document with the project's specifications and possible scenarios (included as Annex I). From there, we started designing the system, and after that developing the components.

Following subsections present: the motivation for the project, the software used and the memory structure.

1.1 Motivation

Smart home management has always seemed like a futuristic paradise. Lights that turn on when you enter the room, fans that activate when the temperature is too high, entire rooms changing its ambient at specific hours or when presence is detected...

Anyone you tell about these things will think it's a future, unreachable perfect home. But they are all wrong. This kind of devices has been in the market for decades. People have been building their own ideal home over the last years. Then, why is this still a "future" topic?

Probably, because of how hidden these devices are from the daily markets. A single look to any of these technologies' web page will show us how easy, and yet not expensive, we can customize our very own home, adding controllable elements to it. But there is still a big concern about these devices: how to control them.

Most of these technologies offer some way or another to control their own devices. Sometimes they will sell you a console from which you can monitor all the installed devices, or a remote control to turn devices on and off. But every technology has its own, and they don't work with other technologies' devices, so you are limited to buy only one kind of device, or have to buy another control console from another technology to control their devices. Even then, you cannot control behaviors that contain devices from different technologies. For example, even if you have a ZWave control console and an Insteon control console, you wouldn't be able to control a ZWave lamp that turns on when an Insteon motion sensor detects movement.

It's obvious that there is still a lot of work to be done in this area. The most important thing is to get the people know about these technologies, and make their control as simple as possible, while still offering a good amount of customization. Maybe one of the most interesting topics about these technologies is to get a centralized, technology-independent system from which anyone can control their home. It must be simple, intuitive, yet powerful and solid.

A fast look to all these technologies' web pages shows that all of them offer a way to connect the devices to a PC to further control their devices, via USB interfaces. So we

decided to get the most popular technologies' USB interfaces (X10, Insteon and ZWave), connect them to the same PC, and develop a server that allows the user to control all these technologies' devices, without having to care about technology-dependent behaviors. Just as if every device was from the same technology. Then, we discussed how we could create a user-friendly environment to control the server. We decided to use one of the most popular devices nowadays: tablets. Tablets offer a powerful and intuitive interface to make complex things look easy, which was our purpose.

1.2 Software Used

Following subsections present the software used for the development of this project. As this environment was selected previously to the definition of my project I will neither enter a deep discussion of this matter nor an analysis of choices. All the modules cited in this subsection will be introduced in chapter 3.

1.2.1 Visual Studio 2010

Visual Studio 2010 has been used to develop the centralized unit, the web service and the web application, all of them developed in C# language. This IDE includes the .NET framework, which has a wide choice of communication options between different applications, so the connectivity between the web application and the centralized unit, via the web service has been greatly simplified by using this framework.

1.2.2 Eclipse and Android SDK

Eclipse has been used to develop the tablet application using Java language and the Android SDK. In order to develop the Android application we have used the Android SDK plugin for Eclipse which also includes the Android Virtual Device Manager (AVD Manager). This manager offers a way to emulate an Android device, such as a tablet, in the PC, which was very useful in the early stages of the project, when we did not have access to a real tablet, or when the tablet was being used by another person.

1.2.3 Crucible

Crucible is a web-operated application that offers a simple way to monitor the evolution of the project when there is more than one person working on it. It saves the current state of the project and all the changes that have been made before. When someone changes the solution, before submitting the changes, (s)he can upload the changes to this application. Then, all the other people working on the same project will be able to see, comment and accept or reject the changes before submitting them.

1.3 Memory Structure

The second chapter will present the objectives and tasks of the project. The third chapter will enounce the description of the system as a whole, while the fourth, fifth, sixth and seventh chapters will define each of the system's components individually.

The annexes of this document include the project environment in the first annex, the Android application's owner's manual in the second annex, and the protocols used in the system in the third annex.

2 PROJECT'S OBJECTIVES DOCUMENT

This chapter describes the list of tasks planned in the project and a Gantt chart indicating the time distribution of these tasks.

2.1 Tasks List

This is the list of tasks that have been done during the Project.

- System design
- Centralized Unit
 - Design
 - Development
- Web Application
 - Design
 - Development
- Android Application
 - Design
 - User Interface
 - Business Logic

2.2 Gantt Chart

Figure 2.1 and Figure 2.2 show the Gantt chart of the Project.

	🕒	Nombre	Duración	Inicio	Terminado
1	✔	System Design	7 days	20/01/12 8:00	30/01/12 17:00
2	✔	☐Centralized Unit	100 days	30/01/12 8:00	15/06/12 17:00
3	✔	Design	14 days	30/01/12 8:00	16/02/12 17:00
4	✔	Development	98 days	1/02/12 8:00	15/06/12 17:00
5	✔	☐Web Application	85 days	20/02/12 8:00	15/06/12 17:00
6	✔	Design	5 days	20/02/12 8:00	24/02/12 17:00
7	✔	Development	85 days	20/02/12 8:00	15/06/12 17:00
8	✔	☐Android Application	75 days	5/03/12 8:00	15/06/12 17:00
9	✔	Design	10 days	5/03/12 8:00	16/03/12 17:00
10	✔	User Interface	70 days	12/03/12 8:00	15/06/12 17:00
11	✔	Business Logic	35 days	12/03/12 8:00	27/04/12 17:00

Figure 2.1: Gantt Chart task list

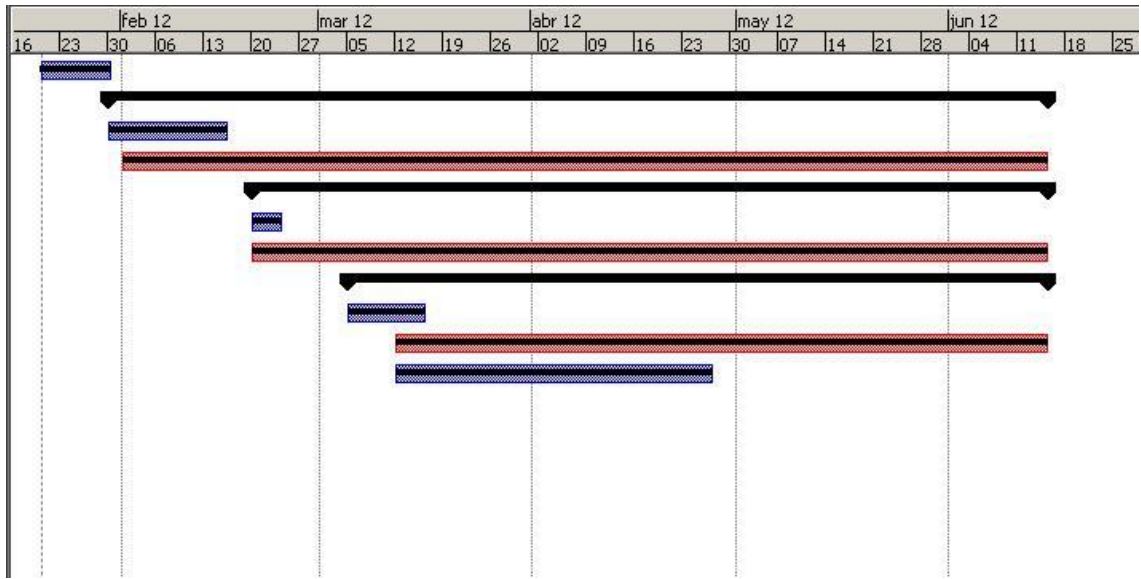


Figure 2.2: Gantt chart

3 SYSTEM DESCRIPTION

The system consists of a **centralized unit** that controls the electric devices attached to it, a **web service** that offers basic interactivity with these devices from the outside, an administration **web application** to manage the system, and an **Android tablet application** that offers the user-oriented functionalities of the system (see Figure 3.1).

Next subsection presents some terminology and generic descriptions. The following ones introduce the centralized unit, the web application and the Android application.

The Web Service and the Centralized unit must be hosted in the same PC, whereas the Database can be hosted either in the same PC or in an external PC.

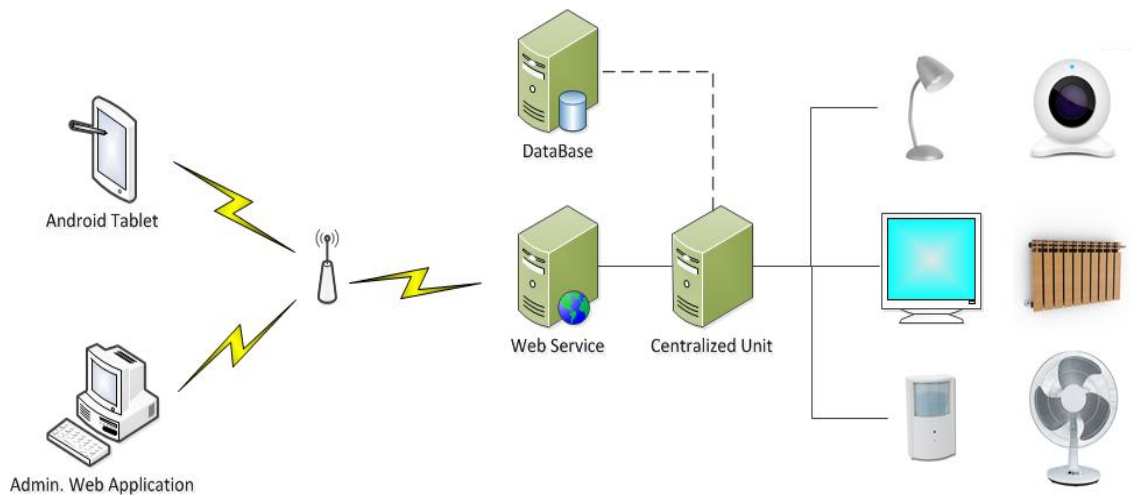


Figure 3.1: System overview

3.1 Terminology and generic description

The system offers the ability to manually control the devices. It also provides other ways to control the behavior of the devices under different conditions, such as time, sensor data, or other devices' status.

It also provides the ability to install, monitor and control live-streaming cameras.

To **manually control** the devices, the system provides functionalities like the following ones:

- Turn off/off appliances or lamp modules.
- Set a specific brightness level of a lamp module.
- Control the pan and tilt of live-streaming cameras.

The next subsections describe the system's provided ways to **adjust the behavior** of the devices.

3.1.1 Schedule

A Schedule consists of a date and an hour in which a series of actions must be executed. The user can also specify an occurrence for the schedule, so when it executes, it automatically refreshes the date and hour to execute in a future time. The system allows the user to configure a schedule to execute once, hourly, daily, weekly, monthly, or yearly.

An example of a schedule could be:

Turn off the hall lamp, adjust the level of the lobby lamp to 20% and turn off the fan every day at 11:00PM starting from tomorrow.

3.1.2 Rule

A Rule consists of a series of conditions, such as specific device status, or sensor readings. If all the conditions are met, a series of actions are executed. To adjust these rules in a more advanced way, the system also allows the user to specify a time period in which the rule can be executed, and the ability to refresh this period every time it ends. The user can also activate or deactivate a rule, which will prevent the rule from executing, even when the conditions are met.

An example of a rule could be:

Turn on the hall lamp and the fan daily between 1:00PM and 5:00PM if the motion sensor detects movement and the temperature is higher than 22 degrees.

3.1.3 Scene

A Scene consists of a series of manual commands stored in a single unit. It is useful to create an ambient for the whole room with a single click. When creating a new scene the user selects which devices will be stored and then the system will store the current state of the devices for future use.

An example of a scene could be:

Turn off the hall lamp, the fan and the lobby lamp, and set the bedroom lamp level to 20%.

3.2 Centralized Unit

3.2.1 Objectives

The Centralized Unit manages and controls all the devices attached to the system. This unit provides multiple methods that allow the component to control devices, add/edit/remove devices, locations, rules, schedules and scenes.

This component receives requests from the Web Service directly, process them, save the needed changes to a Database, send the corresponding commands, and return the result of the operations.

Because this unit directly controls the electric components, it has to be located close to the devices. Although the main appliances that send the commands to the devices can send wireless commands via radio-frequency, they have to be directly connected to the PC to be controlled. The Centralized Unit and the Web Service are the only units that need to be located near the devices.

3.2.2 Design

To manage all the requests the web service can send to the centralized unit, we have divided the unit in different modules, each one focuses on one major aspect of the system (see Figure 3.2).

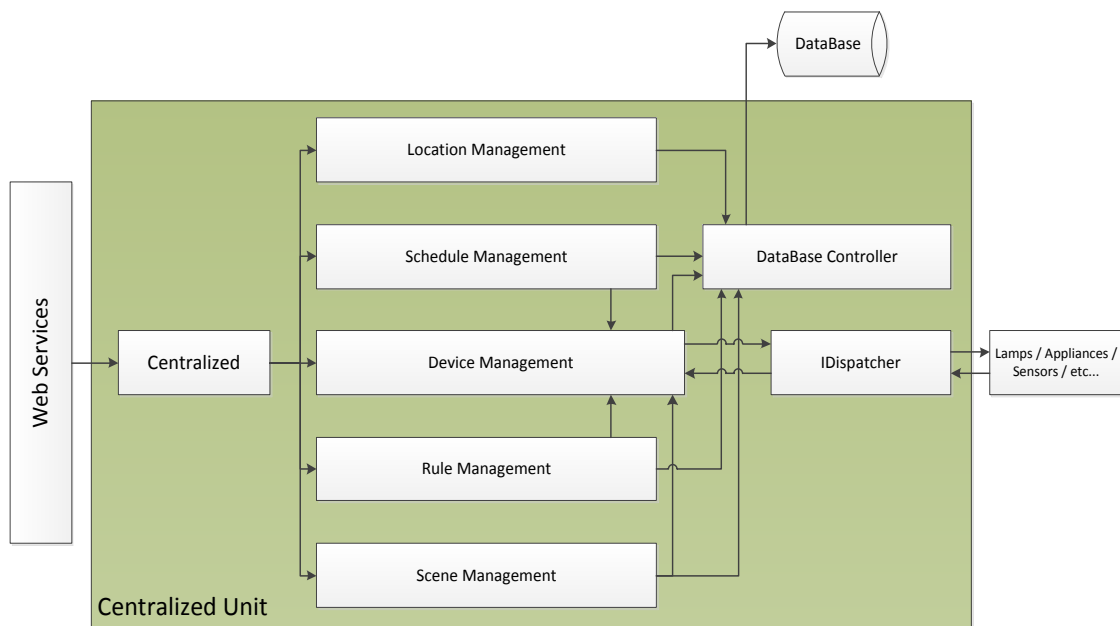


Figure 3.2: Centralized unit major class design

3.2.2.1 Centralized class

The Centralized class receives the requests from the Web Service, and sends them to the corresponding management class depending on the nature of the request.

3.2.2.2 Device Management class

The Device Management class controls all the device specific commands, such as turning on/off, setting the level of lamps, or capturing sensor readings.

3.2.2.3 Location Management class

The Location Management class provides useful methods to categorize the devices in different locations, to offer a more user-friendly environment. Locations are different rooms/areas of a house/office. It is used as a device sorting/grouping mechanism.

3.2.2.4 Schedule Management class

The Schedule Management class controls the schedule specific commands, such as adding, editing, removing Schedules, and sending the corresponding actions to the Device Management class in case a Schedule needs to be executed.

3.2.2.5 Rule Management class

The Rule Management class controls the Rule specific commands, such as adding, editing and removing Rules, checking condition requirements for the Rules, and sending the corresponding actions to the Device Management class in case a Rule needs to be executed.

3.2.2.6 Scene Management class

The Scene Management class controls the Scene specific commands, such as adding, editing and removing Scenes, and sending the corresponding actions to the Device Management class in case a scene is loaded.

3.2.2.7 Database Controller classes

The Database Controller classes (SQLConnection and SQLQuery) connect and manage the database operations.

3.2.2.8 IDispatcher interface

IDispatcher is an interface to control the different devices' protocols. Each device protocol should implement this interface in order to be supported by the system.

3.3 Web application

3.3.1 Objectives

The Web application must provide an advanced way to access and manage the system. It is built for a more advanced user than the Android application, as it requires the user to have some knowledge about how the devices and the system work.

It provides functionalities to add, edit, remove and control devices, as well as adding, editing and removing schedules and rules.

Camera management is considered at high-level architecture and user interface module design. However, this project does not include camera management.

3.3.2 Design

The web application is divided in three subcategories: Device Management, Schedule Management and Rule Management (see Figure 3.3).

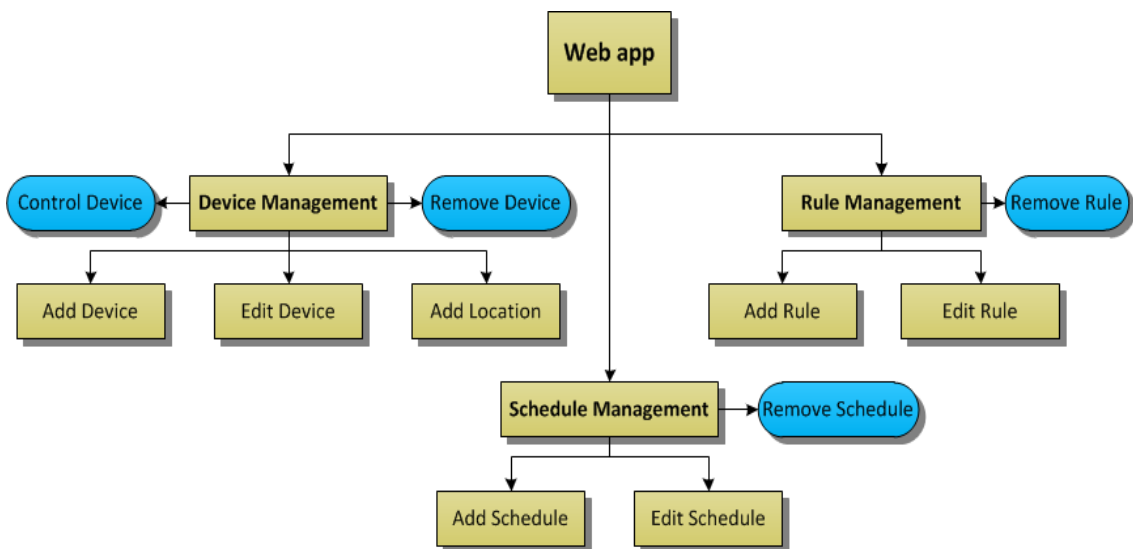


Figure 3.3: Web application design

The next subsections introduce each of the application's page.

3.3.2.1 Home Page

The Home Page shows a welcome message. The user can navigate through the application using the top menu bar.

3.3.2.2 Device Management

The Device Management page shows the list of all the devices in the system, and all their information, including name, location, category, protocol... etc.

From this page the user can access the Add location, Add Device and Edit Device pages, and directly remove and control the devices.

3.3.2.3 Add Device

From the Add Device page the user can add a new device. If there is no location available, or the user wants to put this device in a new location, there is a link to the Add Location page next to the location selector. Once the user adds the device, a new window will show the user the result of adding the device.

3.3.2.4 Edit Device

Similar to the Add Device page, the Edit Device page allows the user to edit the details of an existing device. The user can access this page either by clicking the Edit link of one of the devices in the Device Management page, in which case the device name will be already written, or by going directly from the top menu, in which case the user should enter the name of an existing device.

3.3.2.5 Add Location

The Add Location page allows the user to add a new location. The image of the location must be an Internet URL that points to an image.

3.3.2.6 Schedule Management

The Schedule Management page shows the list of all the Schedules in the system, including the name, date, occurrence, and the list of actions. From this page, the user can access the Add Schedule and the Edit Schedule page, as well as directly removing any Schedule.

3.3.2.7 Add Schedule

The Add Schedule page allows the user to add a new Schedule to the system.

3.3.2.8 Edit Schedule

Similar to the Edit Device page, the Edit Schedule page lets the user edit an existing Schedule.

3.3.2.9 Rule Management

Analogous to the Schedule Management page, the Rule Management page shows the Rule list.

3.3.2.10 Add Rule

Analogous to the Add Schedule page, the Add Rule page lets the user add a new Rule.

3.3.2.11 Edit Rule

Analogous to the Edit Schedule page, the Edit Rule page lets the user edit an existing Rule.

3.4 Android application

3.4.1 Objectives

The Android application offers the user an intuitive and simple interface to access the functionalities of the system. The objective of this application is to cover all the common utilities of the system while not overwhelming the user with complex controls or excessive information.

To avoid having too many buttons and windows in the application, we decided to exclude the following functionalities from the application:

- Add, Edit and Remove devices.
- Activate and deactivate sensors.
- Add, Edit and Remove locations.

3.4.2 Design

The application is divided in 5 main windows: Control window, Schedule Management window, Rule Management window, Security window and Scenes window. There are also two additional windows to add a Rule and to add a Schedule (see Figure 3.4).

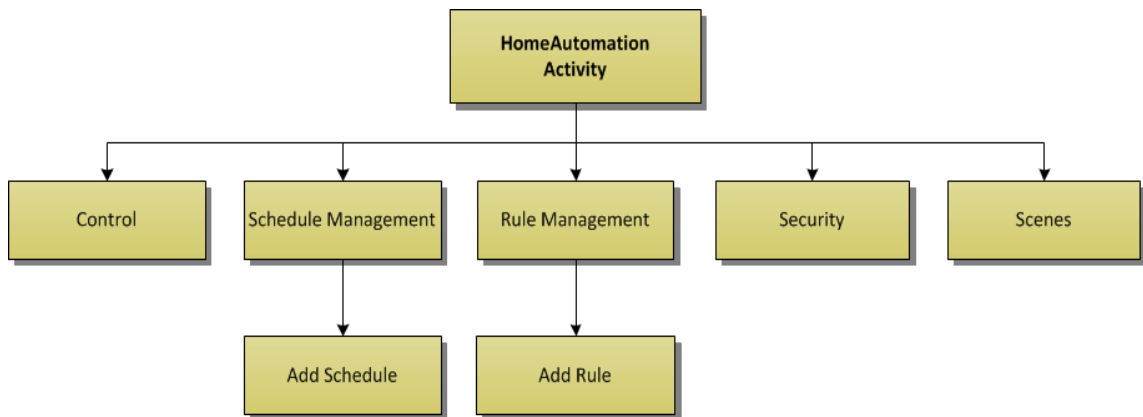


Figure 3.4: Android app activity design

The next subsections introduce each of the application's activities.

3.4.2.1 Welcome

The Welcome window shows a welcoming message, and lets the user connect to the desired centralized unit by providing a username, a password, and a URL.

3.4.2.2 Control

The Control window provides all the manual controls of the devices, as well as offering a categorized view of the devices installed in the system, sorted by their location and their category.

It also shows some information about the system, the locations, the devices, and the weather of a selected city.

3.4.2.3 Schedule Management

The Schedule Management window provides a list of all the installed Schedules in the system, information about each Schedule, and the possibility of removing the Schedules.

From this window the user can also add or edit a Schedule, in which case, the application will lead him/her to the Add Schedule window

3.4.2.4 Add Schedule

The Add Schedule window provides all the fields and options necessary to add, or edit a Schedule. If the user wants to add a Schedule, all the fields will be empty for the user to fill them in. If the user wants to edit a Schedule, all the fields will be filled with the actual schedule's info.

3.4.2.5 Rule Management

Similar to the Schedule Management window, the Rule Management window shows a list of all the installed Rules in the system, information about each Rule, and the possibility of removing a Rule.

It also provides the ability to activate or deactivate a Rule, as well as adding a new Rule, or edit an existing one, via the Add Rule window.

3.4.2.6 Add Rule

This window will provide all the fields and options to add, or edit a Rule, depending on where the user came from. If the user wants to add a new Rule, all the fields will be empty for the user to fill. If the user wants to edit an existing Rule, all the fields will be filled with the Rule's info.

3.4.2.7 Security

This window provides the interface that will be used to control live-streaming cameras, such as viewing the cameras, and controlling their direction via panning and tilting.

3.4.2.8 Scenes

This window provides a list of all the installed scenes in the system, information about each scene, and a way to add, remove or load scenes.

4 CENTRALIZED UNIT AND WEB SERVICE

The Centralized Unit provides direct control over the electric devices and the ability to create, store and process automated actions such as Schedules and Rules over them. It is the main component of the system and the heaviest resource processing unit.

This Centralized Unit operates individually by receiving device events and time events and acting accordingly depending on the stored Schedules and Rules. It can also receive actions from the web service.

All the devices, Schedules, Rules, and Scenes and their data are stored in local lists accessible from the entire component, to avoid having multiple instances of the same information. In order to keep an alternative storage system in case of failure or shutdown, a database has been created in which we store these lists and update them whenever there is any change. So in case of failure all the local lists can be regenerated from the database.

In order to keep the system as simplest as possible, the Web Service has been included in the same *Visual C# solution* as the Centralized Unit. This way, the communication between the Web Service and the Centralized Unit is direct.

All the Web Service calls will go through the Centralized main class and from here to the specific management classes, depending on the method called. Once the specific management class processes this call, it can call the Database Controller classes to update the data in the database, or call another management class to have access to other areas of the system. For example, a call to create a new Scene will go to the Scene Management class, and from here it will access both the Device Management class to ask for the current status of the devices involved in the Scene, and the Database Controller classes to add the Scene.

This chapter will explain the functionalities of the Web Service, the design and main classes of the Centralized unit, an example of execution sequence, and the database design.

4.1 Web Service

The Web Service was initially designed as a separate entity from the Centralized Unit, resident in the same PC. Later on, we decided to make it as a part of the Centralized Unit to keep the system simple. The Web Service provides all the functionalities that can be accessed from outside platforms, such as the Android application.

As a public interface, this unit must offer an authentication system to prevent the access to non-authorized users. However, though planned, it hasn't been implemented yet, and will be added in the future.

The functionalities this Web Service offers are represented by 21 methods:

- **XmlDocument GetDeviceList();**
Returns an XML with the list of the installed devices.
- **XmlDocument GetLocationList();**
Returns an XML with the list of locations.
- **XmlDocument GetScheduleList();**
Returns an XML with the list of Schedules.
- **XmlDocument GetRuleList();**
Returns an XML with the list of Rules.
- **XmlDocument GetSceneList();**
Returns an XML with the list of Scenes.
- **XmlDocument GetConditionList(String type);**
type: a String containing the type of the device you want the valid conditions from. It can be Lamp, Appliance, Luminance, Temperature, or Motion Detection.

Return an XML with the list of conditions available for the type of devices specified in the **type** attribute.

- **XmlDocument GetActionList(String type);**
type: a String containing the type of the device you want the valid actions from. It can be either Lamp or Appliance.

Returns an XML with the list of actions available for the type of devices specified in the **type** attribute.

- **Int SwitchStatus(String name);**
name: name of the device (not display name, the unique name)

Turns on/off the device. Returns an integer specifying if the operation succeeded or not.

- **Int SetLevel(String name, int level);**
name: name of the device (not display name, the unique name)
level: desired level for the device, from 0 to 100

Sets the level specified in the **level** attribute to the device specified by the **name** attribute. Returns an integer specifying if the operation succeeded or not.

- **Int AddDevice(String sxml);**

sxml: A String, with XML format, that contains the data of the new device.

Inserts the device in the system and the database. Returns an integer specifying if the operation succeeded or not.

- **Int AddLocation(String sxml);**

sxml: A String, with XML syntax, that contain the data of the new location.

Inserts the location in the system and the database. Returns an integer specifying if the operation succeeded or not.

- **Int AddRule(String sxml);**

sxml: A String, with XML syntax, that contains the data of the new Rule.

Inserts the Rule in the system and the database. Returns an integer specifying if the operation succeeded or not.

- **Int AddSchedule(String sxml);**

sxml: A String, with XML syntax, that contains the data of the new Schedule.

Inserts the Schedule in the system and the database. Returns an integer specifying if the operation succeeded or not.

- **int createScene(string sxml);**

sxml: A String, with XML syntax, that contains the data of the new Scene.

Inserts the Scene in the system and the database. Returns an integer specifying if the operation succeeded or not.

- **Int EditDevice(String sxml);**

sxml: A String, with XML syntax, that contains the data of the device.

Edits the existing device updating the fields specified in the **sxml** xml object. Returns an integer specifying if the operation succeeded or not.

- **Int EditRule(String sxml);**

sxml: A String, with XML syntax, that contains the data of the Rule.

Edits the existing Rule updating the fields specified in the **sxml** xml object. Returns an integer specifying if the operation succeeded or not.

- **Int EditSchedule(String sxml);**

sxml: A String, with XML syntax, that contains the data of the Schedule.

Edits the existing Schedule updating the fields specified in the **sxml** xml object. Returns an integer specifying if the operation succeeded or not.

- Int **RemoveSchedule(long ID);**
ID: ID of the Schedule.

Removes the Schedule specified in the **ID** attribute. Returns an integer specifying if the operation succeeded or not.

- Int **RemoveRule(long ID);**
ID: ID of the Rule

Removes the Rule specified in the **ID** attribute. Returns an integer specifying if the operation succeeded or not.

- Int **RemoveScene(long ID);**
ID: ID of the Scene

Removes the Scene specified in the **ID** attribute. Returns an integer specifying if the operation succeeded or not.

- Void **executeAScene(long ID);**
ID: ID of the Scene.

Executes the actions associated with the Scene specified in the **ID** attribute.

4.2 Design and Main classes

The classes in the centralized unit are divided in four modules (three interfaces and the business logic). The interfaces are with the Web Service (Centralized class), with the electric devices (IDispatcher and its implementations), and with the database (SQLConnection and SQLQuery). The business logic is divided in six management classes (Device Management, Location Management, Schedule Management, Schedule Management, Rule Management, Scene Management and Security Camera Management) (see Figure 4.1).

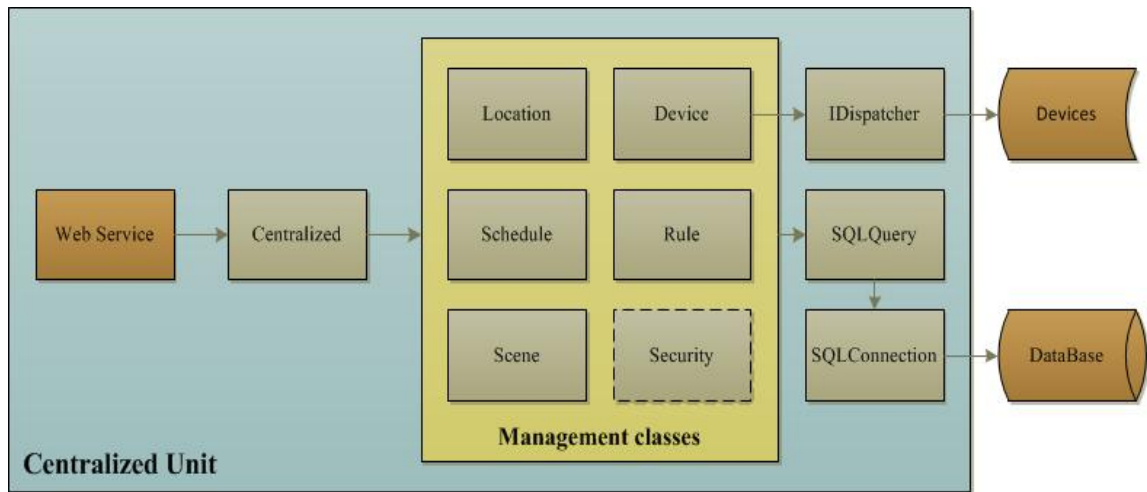


Figure 4.1: Centralized Unit modules

Every management class stores a local copy of the corresponding list from the database. For example, the Schedule Management class stores a local copy of the list of Schedules of the database. Each time a method has to modify the list of Schedules, it will modify both the local copy and the database list in order to maintain synchronous copies. The initial copy from the database to the local list is done when the Centralized Unit is started, so there is only one database query to get the Schedule list.

The next subsections describe each of the Centralized Unit's main classes.

4.2.1 Centralized

The Centralized class communicates the web service with the business logic of the Centralized Unit. For this purpose, the Centralized class has a method for every web service method offered. Each web service method will call the corresponding method of the Centralized class.

After this call, the Centralized class acts like a dispatcher. Depending on the nature of the call, it will wrap all the information of the call in an instance of the corresponding object (Device, Rule, Schedule, Location, or Scene) if necessary, and call the corresponding management class.

4.2.2 Device Management

The Device Management class includes the business logic corresponding to device specific actions, such as adding, editing and removing devices, and also controlling them. After receiving a call from either the Centralized class or any other Management class, this class will process the call and call either the IDispatcher class to control a device, or the SQLQuery class to access the database, or both.

This is the only management class that calls to the IDispatcher interface, and thus the only class that can control the devices. Any other class that needs to control a device needs to call the Device Management class first. This has been done to create a coherent responsibility for each of the management classes.

4.2.3 Schedule Management

The Schedule Management class includes the business logic corresponding to the Scheduling system. It offers the methods to add, edit, and remove Schedules.

It also has the ability to act independently upon external events, such as time passing. If there is a specific Schedule that has been activated in the current time, it will parse the actions associated to the Schedule, and call the Device Management class to control the required devices. These executions are sent using a new thread for every action. This way, the performance of the system is improved if the list of actions is relatively big.

4.2.4 Rule Management

The Rule Management class includes the business logic corresponding to the Rule system. It offers the methods to add, edit, and remove Rules.

Every 30 seconds a timer will wake up, and check the condition requirements for every Rule that is active (including scheduling requirements). If every single condition is met, the list of actions of the Rule is parsed, and the corresponding actions sent to the Device Management class to control the corresponding devices. As with the Schedules' list of actions, these actions are sent using new threads to improve the performance.

4.2.5 Location Management

The Location Management class includes the business logic corresponding to the locations of the devices. It offers a method to add locations to the system and the database.

4.2.6 Scene Management

The Scene Management class includes the business logic corresponding to the Scene system. It offers the methods to add, edit, remove, and execute Scenes.

The execution of Scenes is done similarly to the execution of Schedules and Rules. Whenever there is a call to execute a Scene, the list of devices, stored as a list of actions, is parsed and all the corresponding actions are sent to the Device Management class to execute them. As usual, these executions are sent via new threads, to improve the performance of the system if the list of actions is relatively big.

4.2.7 Security Camera Management (not implemented)

The Security Camera Management class will include the business logic corresponding to the management of the live-streaming cameras. As this a future work there is still no specifications designed, but it will include any method necessary to manage and control the live-streaming cameras.

Depending on how the cameras will be stored in the system, it might also call the Device Management class to control them.

4.2.8 IDispatcher

The IDispatcher interface offers a list of methods that all the different device protocol-specific classes must implement in order to be supported by the Centralized Unit. It also offers some methods that not all the protocols support, but can be implemented if a specific protocol supports them, like adding and removing associations between devices, enabling/disabling the polling system, and getting device information such as status and level.

For now, the system does not support associations between devices, so those methods are not implemented yet and are only used for testing purposes. The currently implemented protocol classes are explained next.

4.2.8.1 X10

As it is a very simple protocol and the SDK provided by X10 is very limited, the X10 class can only implement the methods to switch status, set level, and receive device events. Every other method from the interface is empty as there is no way to provide that functionality.

4.2.8.2 Insteon

A more advanced protocol than X10, it offers the methods to switch status, set level, get device events, and also asking for device information such as status and level.

The way to communicate the system with Insteon devices is by a RS232 communication interface specifically developed to control Insteon device, using the documentation provided by the Insteon SDK web page.¹ As it is merely an auxiliary class to support this protocol, it would not be covered in this document.

4.2.8.3 ZWave

This is the most advanced protocol. The ZWave class implements the basic methods to switch status, set device level, receive device events, ask for device information, and also enabling and disabling the polling mechanism. The ZWave polling mechanism offers a way to receive periodic events indicating the status of every device. It is also a way to detect device faults such as disconnections or malfunctions. Thus there is no support to detect malfunctioning in this document's writing date, it is a future work project.

The way to communicate the system with ZWave devices is using the open-source OpenZWave library². As it is an open-source project, we could modify some of the code in the library to adapt it to our needs. In this case, we modified the polling mechanism inside the library to lighten the processing time to check the devices' status, as it was using too many resources for the polling mechanism.

¹ <http://code.insteon.net>

² <http://code.google.com/p/open-zwave/>

4.2.9 SqlConnection – SQLQuery

The SqlConnection and SQLQuery classes manage all the database queries. The database connection is defined in an external XML file so it is easy to change the database location without changing the code. The system supports both SQL and MySQL databases.

This class is only accessible from the management classes, and not from the Centralized class, in order to separate the layers of the system. It acts like a logical interface that separates the database from the management classes so they don't have to deal with SQL sentences.

As databases are normally very time-consuming resources, the system tends to use the database only to modify data, while it only asks for stored information at the start of the system, and then keeps a local list that updates at once with the database.

4.3 Example of execution sequence

This subsection describes a possible execution sequence. If the Web Service receives a call to edit an existing schedule, the Web Service will parse the received xml object, and call the Centralized method EditSchedule with all the necessary fields. The Centralized class will then wrap all this fields in a Schedule object, and send the object to the ScheduleManagement class. The ScheduleManagement class will check if the schedule exists, and then call the SQLQuery class to edit the corresponding fields of the schedule in the database. If all goes right, the ScheduleManagement class will update the info in the local list and return an integer indicating the success of the operation (see Figure 4.2).

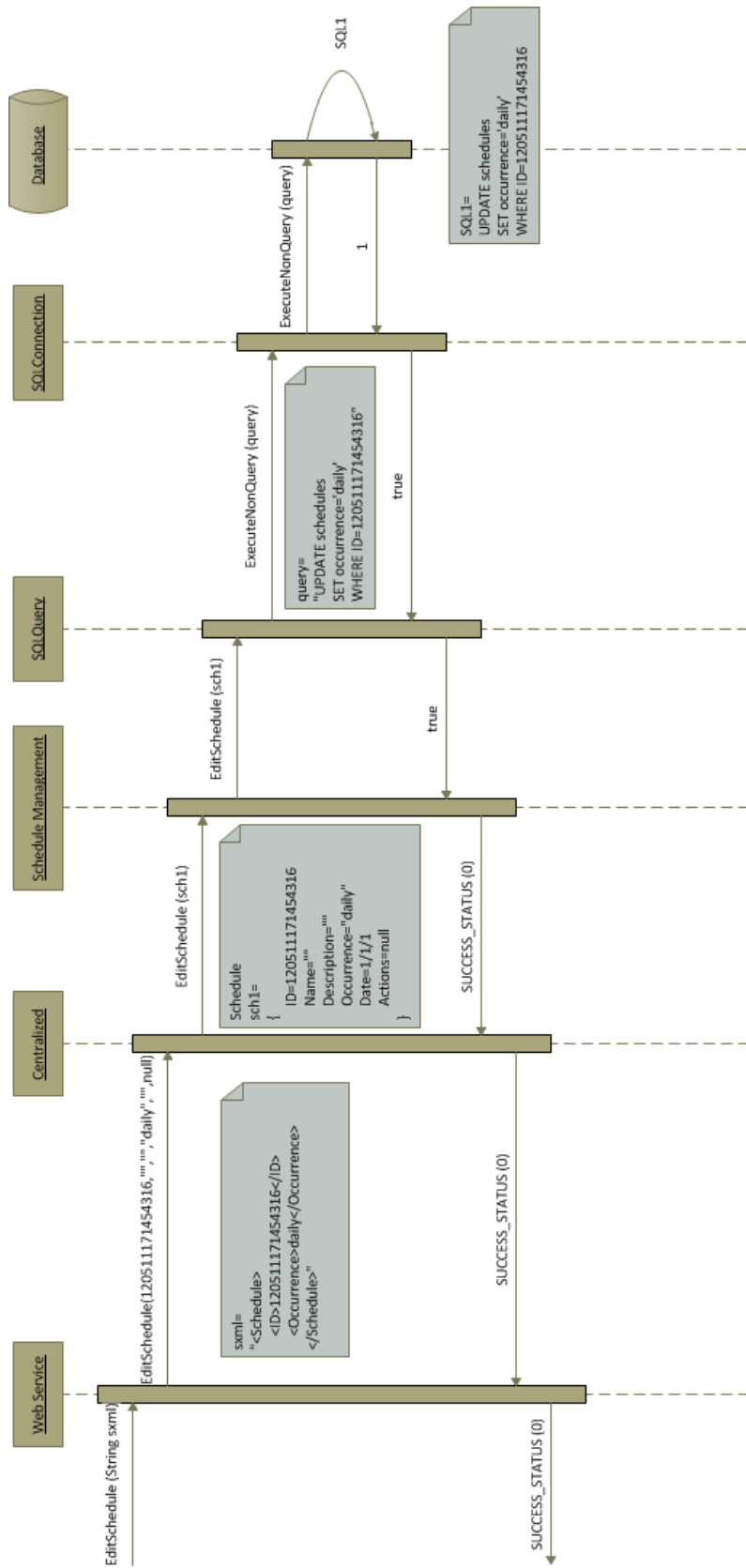


Figure 4.2: Execution sequence for edit schedule

4.4 Database

In order to store the information in a persistent and safe environment we decided to create a MySQL database. In this database we store all the installed devices, locations, Rules, Schedules and Scenes in order to keep a backup storage system.

Databases offer a great way to store data, but can be very time consuming to constantly read or update them, so we decided to avoid using the database for the most common and used operations, like asking for the device list.

For that purpose we store the device list (and the Schedule, Rule, location and Scene lists) in a local volatile object in the Centralized Unit that we populate when the Centralized Unit starts. This reduces the amount of database calls extremely. This way, whenever an external application needs the device list, we will directly return this object, instead of having to access the database.

In order to connect the Centralized Unit to the database we use the MySQL Connector for ADO.NET provided by MySQL³.

The database stores the information of all the devices, locations, Schedules, Rules and Scenes. Because the action for the Schedules are the same for Rules, and can also be used as a way to store the Scenes' actions, we decided to create a different table to store this data. To maintain an analog structure, we also store the Rules' conditions in a separate table.

Devices and locations are identified by their names, while Schedules, Rules and Scenes, are identified by a dynamically generated unique number ID. We decided to create a separate table (RuleScheduleMap) to store the ID and the kind of object it represents, so it's easier to remove an existing object.

Actions and Conditions will be identified by their linked Schedule, Rule, or Scene's ID.

Figure 4.3 represents the design of the database.

A Rule can either have conditions or be scheduled for a fixed period of time, or both, but not none. An action must always be related to a Rule, a Schedule or a Scene, and only to one of these. A single Rule cannot contain duplicate conditions or actions. Analogous, a Schedule and a Scene cannot contain duplicated actions.

³ <http://www.mysql.com/products/connector/>

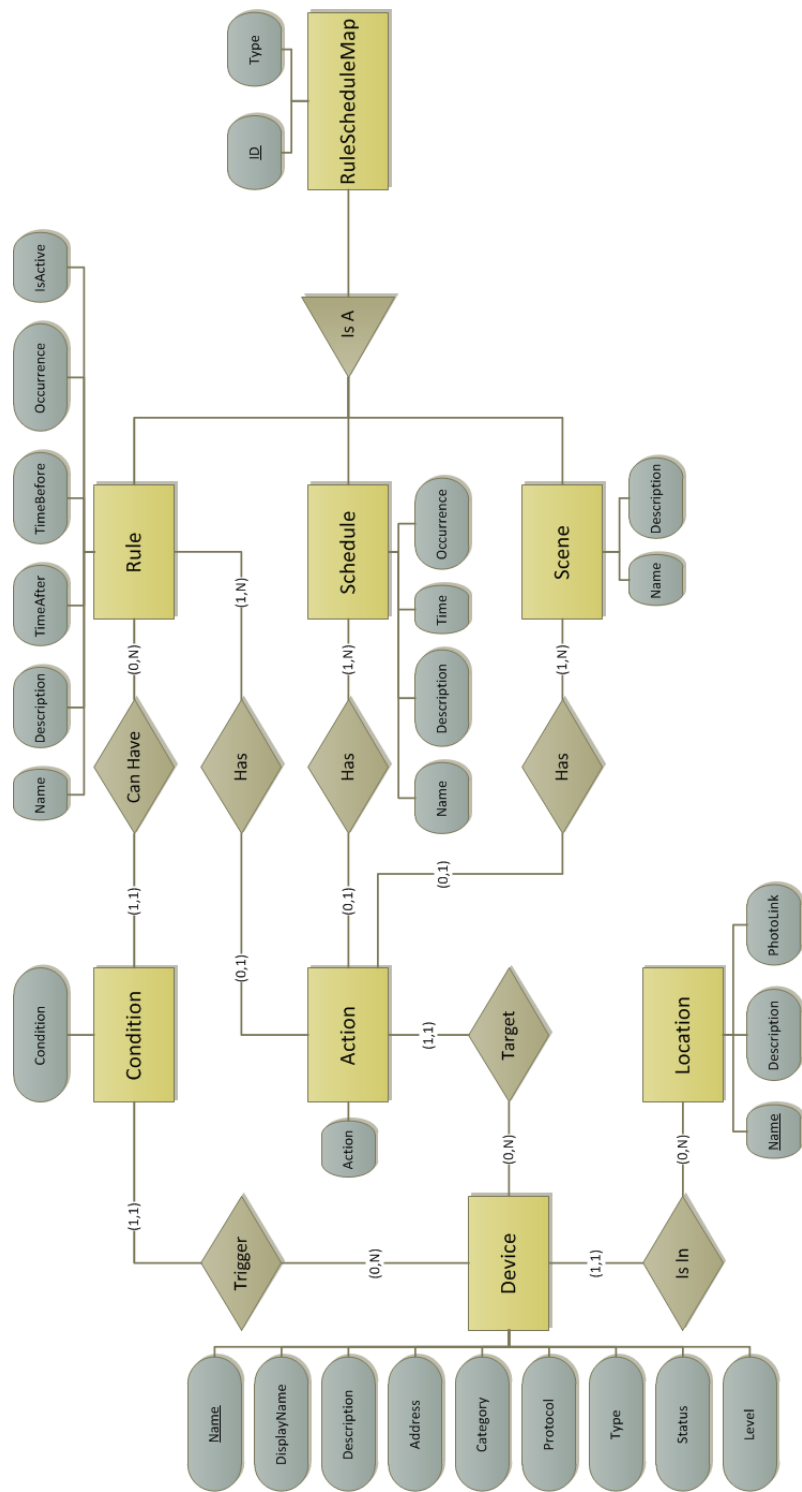


Figure 4.3: Database design

5 WEB APPLICATION

The Web Application's objective is to provide a more advanced way to manage the Home Automation System. By providing more specific functionalities, the user can have more control over the devices, and also an extra platform in case he doesn't have access to the tablet application.

This platform includes functionalities that are not available in the tablet application, such as adding, editing and removing devices, or adding new locations. We decided to remove these functionalities from the tablet application in order to keep the tablet application simple and accessible.

By doing a web application, the user only has to have a web browser installed in order to have access to the system, greatly simplifying the resources necessary to control the system.

This chapter explains the design of the application, the main web pages, and the connectivity with the Web Service.

5.1 Design

The web application has been designed in three modules: Device Management, Schedule Management and Rule Management. In the future, it will also contain modules to control the security cameras and the Scenes.

The access to these modules and the pages associated with them is a horizontal menu in the top side of the web page. This menu bar is accessible from every page of the application (see Figure 5.1).

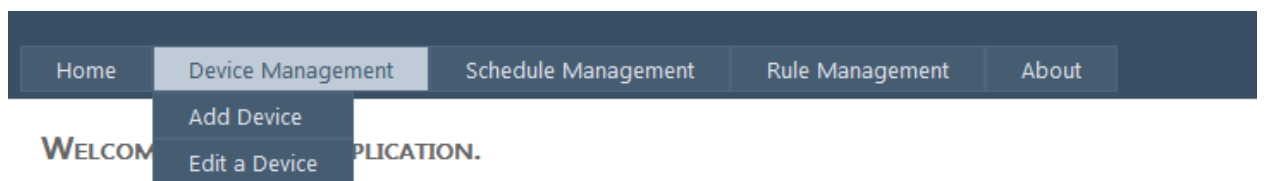


Figure 5.1: Top menu

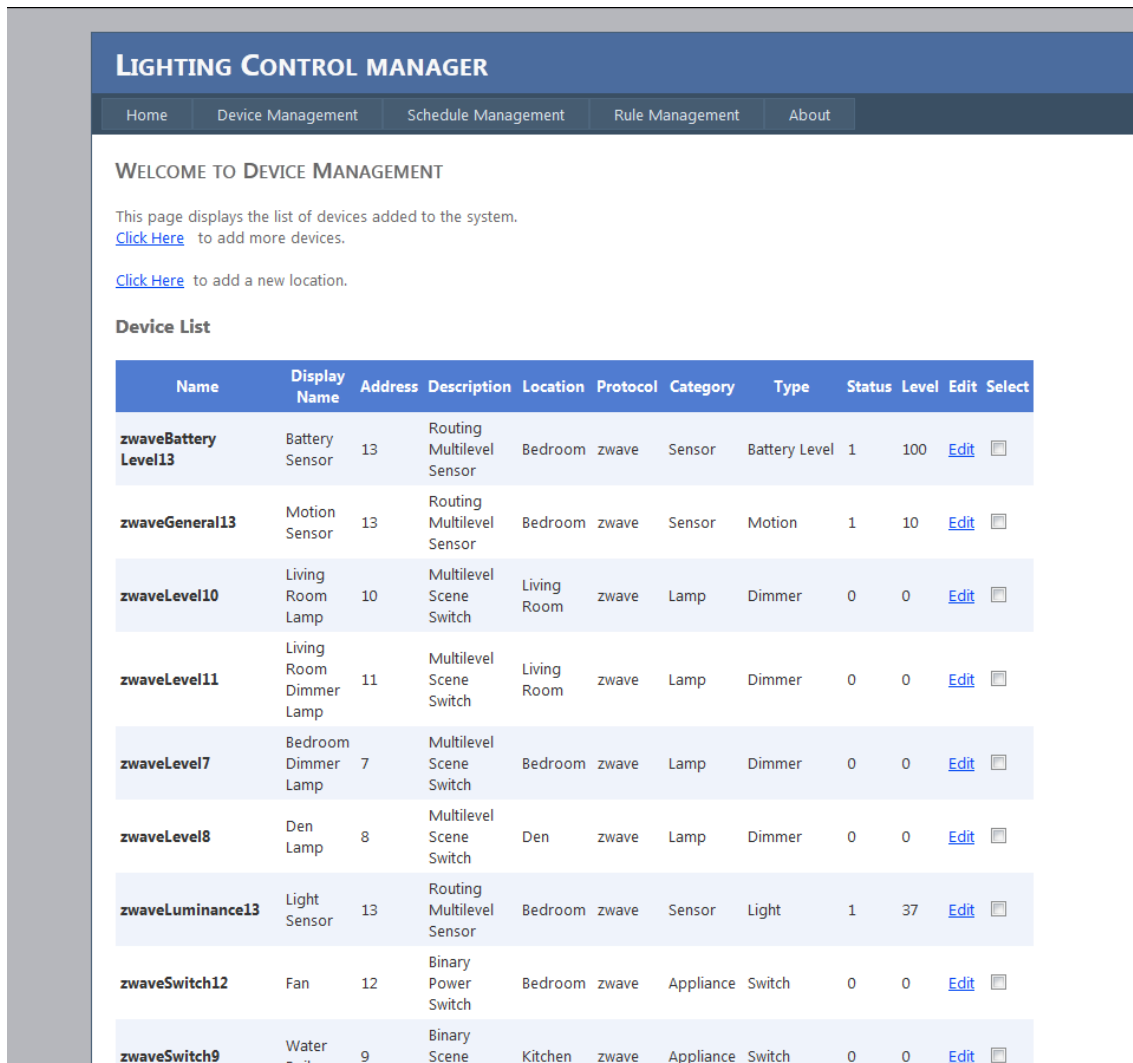
5.2 Main pages

Each of the three modules has a main page. This page shows a table with the contents of either the list of devices, Schedules or Rules. From this page, the user can manage the objects in the table, or access other pages to add new objects to the table.

The next subsections describe each of the Web Application's pages.

5.2.1 Device Management

The Device Management page shows a table containing all the devices installed in the system, as well as all their information. This list is updated every 30 seconds to show status changes (see Figure 5.2).



Name	Display Name	Address	Description	Location	Protocol	Category	Type	Status	Level	Edit	Select
zwaveBatteryLevel13	Battery Sensor	13	Routing Multilevel Sensor	Bedroom	zwave	Sensor	Battery Level	1	100	Edit	<input type="checkbox"/>
zwaveGeneral13	Motion Sensor	13	Routing Multilevel Sensor	Bedroom	zwave	Sensor	Motion	1	10	Edit	<input type="checkbox"/>
zwaveLevel10	Living Room Lamp	10	Multilevel Scene Switch	Living Room	zwave	Lamp	Dimmer	0	0	Edit	<input type="checkbox"/>
zwaveLevel11	Living Room Dimmer Lamp	11	Multilevel Scene Switch	Living Room	zwave	Lamp	Dimmer	0	0	Edit	<input type="checkbox"/>
zwaveLevel7	Bedroom Dimmer Lamp	7	Multilevel Scene Switch	Bedroom	zwave	Lamp	Dimmer	0	0	Edit	<input type="checkbox"/>
zwaveLevel8	Den Lamp	8	Multilevel Scene Switch	Den	zwave	Lamp	Dimmer	0	0	Edit	<input type="checkbox"/>
zwaveLuminance13	Light Sensor	13	Routing Multilevel Sensor	Bedroom	zwave	Sensor	Light	1	37	Edit	<input type="checkbox"/>
zwaveSwitch12	Fan	12	Binary Power Switch	Bedroom	zwave	Appliance	Switch	0	0	Edit	<input type="checkbox"/>
zwaveSwitch9	Water Boiler	9	Binary Scene	Kitchen	zwave	Appliance	Switch	0	0	Edit	<input type="checkbox"/>

Figure 5.2: Device Management page

One of the columns of this table is a checkbox. The user can select all the devices (s)he wants, and after that, click the Delete button below the table to remove all the selected devices. Another column of the table is a link to the Edit Device page, in which the user can change the information of the device.

There is also a link to add a new device to the system and another link to add a new location, as well as a pair of drop down menus that let the user control the devices, by selecting the device in the first drop down menu, and then selecting the action in the

second drop down menu. Once both items have been selected, the user can press the Accept button to send the action to the system.

5.2.2 Add Device

The Add Device page is a normal form page that lets the user fill the details of the new device. If this is the first device, and there are no locations added to the system yet, the page will redirect to the Add Location page to add a new location before adding the device. If there are available locations in the system, the user will be able to select a location from the drop down list of locations, in the corresponding form field. There is also a permanent link in this page that redirects the user to the Add Location page if the user wants to create a new location, even if there are existing ones (see Figure 5.3).

LIGHTING CONTROL MANAGER

Home Device Management Schedule Management Rule Management About

ADD DEVICE

You can add devices to the system, such as lamp modules, appliance modules, cameras, and sensors

Name

Device Address

Description

Location [Click Here](#) to create a new location

Type

Protocol

Category

Figure 5.3: Add Device page

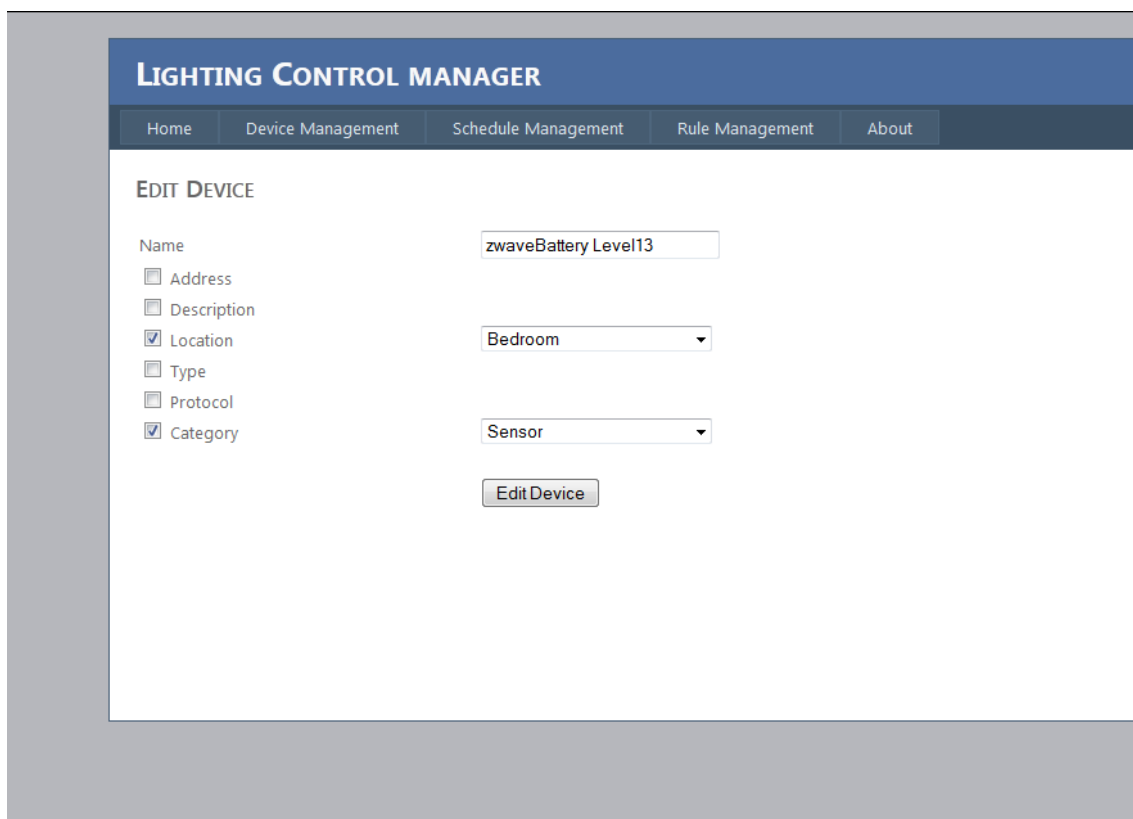
This is the only platform from which the user can add a device. This is done to prevent having complex functions in the tablet application, because the user has to know some technical details about the device in order to fill all the fields in the form, like the Address field, or the Type field, as well as the Protocol field. In order to maintain a simple tablet application, we decided to keep these functionalities off the tablet.

5.2.3 Edit Device

The user can either come to the Edit Device page from the top menu, or by clicking an Edit Device link in the device list table in the Device Management page.

This page will show the name of the selected device if the user came from the Device Management page. If the user came from the top menu, the device name field will be empty.

In either case, apart from the field for the name, the page will show a list of checkboxes, one for each of the fields that can be edited in the device. This includes the Address, Description, Location, Type, Protocol, and Category fields. If the user checks one of these checkboxes, the corresponding field will become visible, allowing the user to change it. The user can change all the fields (s)he wants with a single Edit Device operation, by checking multiple checkboxes and changing the corresponding fields (see Figure 5.4).



The screenshot displays the 'EDIT DEVICE' page within the 'LIGHTING CONTROL MANAGER' application. The page features a navigation bar with links for Home, Device Management, Schedule Management, Rule Management, and About. The main content area is titled 'EDIT DEVICE' and contains a form with the following elements:

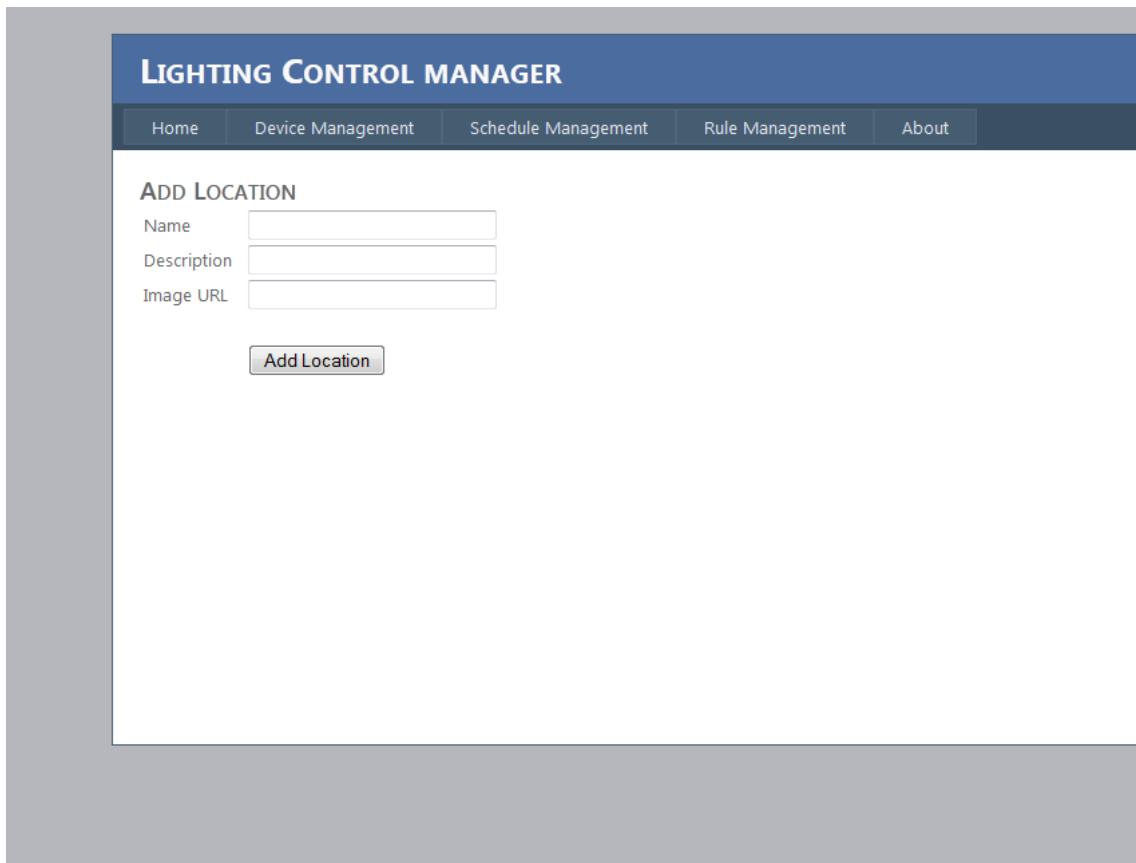
- Name:** A text input field containing 'zwaveBattery Level13'.
- Address:** A checkbox that is unchecked.
- Description:** A checkbox that is unchecked.
- Location:** A checkbox that is checked, with a dropdown menu showing 'Bedroom'.
- Type:** A checkbox that is unchecked.
- Protocol:** A checkbox that is unchecked.
- Category:** A checkbox that is checked, with a dropdown menu showing 'Sensor'.
- Edit Device:** A button located at the bottom of the form.

Figure 5.4: Edit Device page

Once the user finishes editing the device, a button in the bottom side of the page will let the user submit the changes. A message will appear indicating the success or failure of the operation, and a brief error message if something went wrong.

5.2.4 Add Location

The Add Location page is a simple form that lets the user select a name, a description, and a URL that links to an image for a new location in which the user can add devices (see Figure 5.5).



The screenshot shows a web application interface for a lighting control system. At the top, there is a dark blue header with the text "LIGHTING CONTROL MANAGER" in white. Below the header is a navigation menu with five items: "Home", "Device Management", "Schedule Management", "Rule Management", and "About". The main content area is titled "ADD LOCATION" and contains a form with three input fields: "Name", "Description", and "Image URL". Each field is followed by a text input box. Below the input fields is a button labeled "Add Location".

Figure 5.5: Add Location page

5.2.5 Schedule Management

The Schedule Management page shows a table with the list of all the Schedules in the system. For every Schedule it will show the name, the date, the occurrence and the list of actions of the schedule. Analogous to the device management page, it will let the user select the schedules (s)he wants to remove by checking the checkbox in the table, and then pressing the Remove button below the table. It will also have a link to edit each one of the schedules in the table (see Figure 5.6).

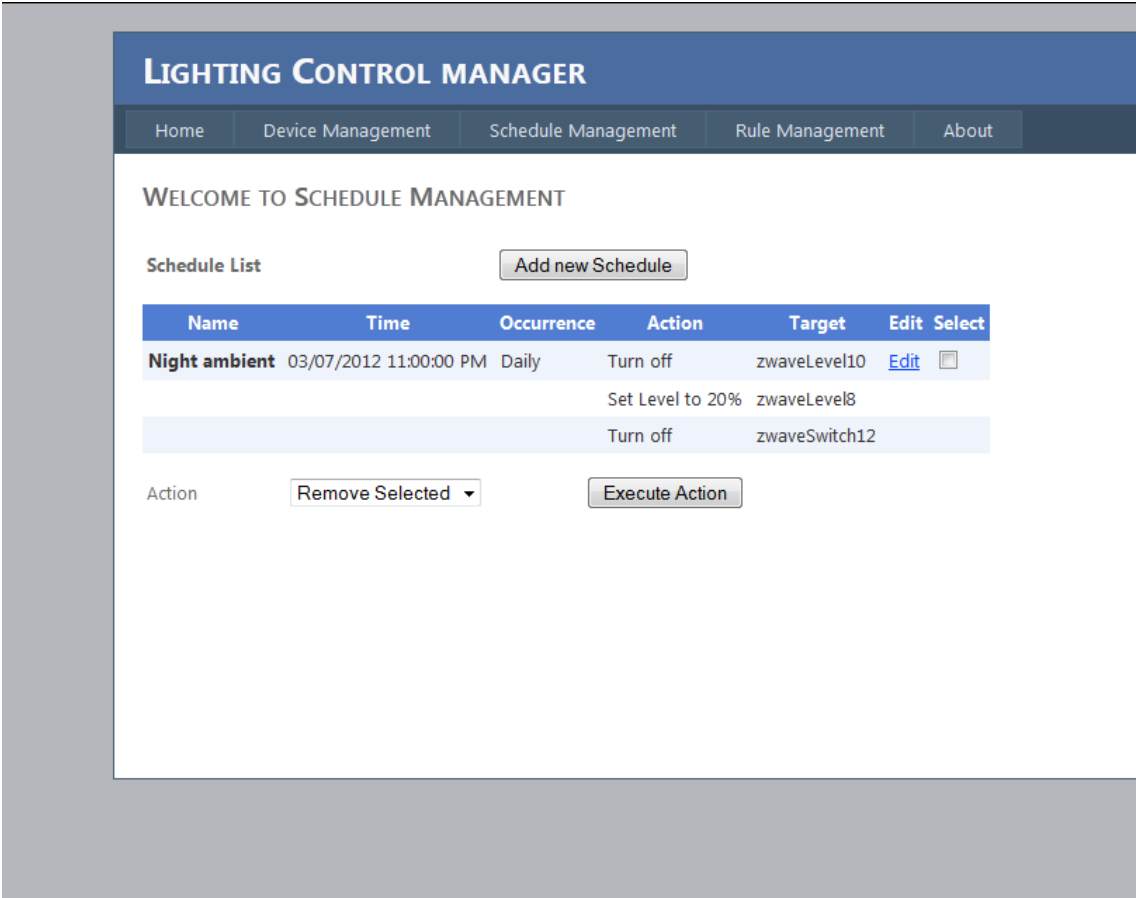


Figure 5.6: Schedule Management page

It also shows a link to add a new schedule in the top side of the table.

5.2.6 Add Schedule

The Add Schedule page is a simple form to allow the user to add a new Schedule in the system. It will show all the required fields to add a new Schedule, such as the name, occurrence, date and time. It will also let the user add actions to the Schedule. Each action is determined by two drop down lists, one for the action itself, and one for the targeted device. The user can add more actions to the schedule by clicking the Add Additional Actions link, below the actions (see Figure 5.7).

LIGHTING CONTROL MANAGER

Home Device Management Schedule Management Rule Management About

Create New Schedule

Fill out the information below to add a new schedule to the system

Name

Occurrence **Once** ▼

Effective Date

Hour **00** ▼ Min **00** ▼

Actions List Perform Action **Select Action** ▼
To device **Select device** ▼

[Add Additional Actions](#)

Figure 5.7: Add Schedule page

Once the user has filled all the fields and added at least one action to the schedule, the user can submit the schedule to the system by clicking the Add Schedule button in the bottom side of the page. A message will appear indicating the success or failure of the operation, showing a brief error message in case something went wrong.

5.2.7 Edit Schedule

Similar to the Edit Device page, the Edit Schedule page is accessible from the schedule list table's Edit Schedule link, or from the top menu of the application. It shows a field with the selected schedule's name, or blank if the user came from the top menu. Below this field, a list of checkboxes lets the user select which fields of the schedule wants to edit. Once a checkbox is checked, the corresponding field will become visible. There is a checkbox for the schedule's occurrence, date, and actions (see Figure 5.8).

The screenshot displays the 'EDIT SCHEDULE' page within the 'LIGHTING CONTROL MANAGER' application. The page features a navigation bar with links for 'Home', 'Device Management', 'Schedule Management', 'Rule Management', and 'About'. The main content area is titled 'EDIT SCHEDULE' and contains the following form elements:

- Name:** A text input field containing 'Night ambient' and a button labeled 'Edit Schedule' to its right.
- Occurrence:** A checked checkbox followed by a dropdown menu set to 'Daily'.
- Date:** A checked checkbox followed by three dropdown menus for month ('June'), day ('19'), and year ('2012'). Below these are three more dropdown menus for time, set to '23', ': 0', and ': 0'.
- Actions:** An unchecked checkbox.

Figure 5.8: Edit Schedule page

Once the user finishes editing the schedule, a button in the right side of the page will let the user submit the changes. A message will appear indicating the success or failure of the operation, and a brief error message if something went wrong.

5.2.8 Rule Management

Very similar to the schedule management window, the Rule Management page shows a table with the list of rules in the system. The table shows their information, a checkbox to select and remove the rules, and a link to edit a single rule (see Figure 5.9).

LIGHTING CONTROL MANAGER

Home Device Management Schedule Management Rule Management About

WELCOME TO RULE MANAGEMENT

Rule List [Add new Rule](#)

Name	Effective from	Effective to	Occurrence	Conditions	Actions	Edit Select
Summer Weather	19/07/2012 01:00:00 PM	19/07/2012 05:00:00 PM	Daily	zwaveGeneral13 - Motion is Detected zwaveTemperature13 - Temp above 22°C	Turn on - zwaveLevel10 Turn on - zwaveSwitch12	Edit <input type="checkbox"/>

Action [Remove Selected](#) [Execute Action](#)

Figure 5.9: Rule Management page

The page also shows a link to add a new rule in top of the table.

5.2.9 Add Rule

Very similar to the Add Schedule page, the Add Rule page lets the user fill all the required fields to add a new Rule to the system. In this case, the user has to enter both conditions and actions to the Rule. The way to add conditions to the Rule is similar to the way of adding actions. Each condition is represented by two drop down lists, one to select the device, and one to select the condition the device has to meet.

In the case of Rules, the user can select whether the Rule will be scheduled or not by checking or unchecking the Scheduled Rule checkbox. If the user wants to schedule the Rule, the fields to select the period of time in which the Rule is active will become visible, as well as the occurrence field (see Figure 5.10).

LIGHTING CONTROL MANAGER

Home Device Management Schedule Management Rule Management About

Create New Rule

Fill out the information below to add a new rule to the system

Name

Occurrence **Once** ▼

Scheduled Rule

Effective Date From

Hour **00** ▼ Min **00** ▼

To

Label **00** ▼ Min **00** ▼

Condition(s) [Add Condition](#)

Action(s) Perform Action ▼

To device

[Add Additional Actions](#)

Figure 5.10: Add Rule page

5.2.10 Edit Rule

Under development at this document's date, the Edit Rule page will allow the user to edit the selected rule.

Very similar to the Edit Schedule page, accessible both from the rule list table in the Rule Management window or from the top menu of the application, it will show the selected rule name and a series of checkboxes allowing the user to select the fields to be edited.

In the case of rules, the checkboxes will include the Description, whether the rule is scheduled or not, the period of time and its occurrence if the rule is scheduled, and the list of conditions and actions (see Figure 5.11).

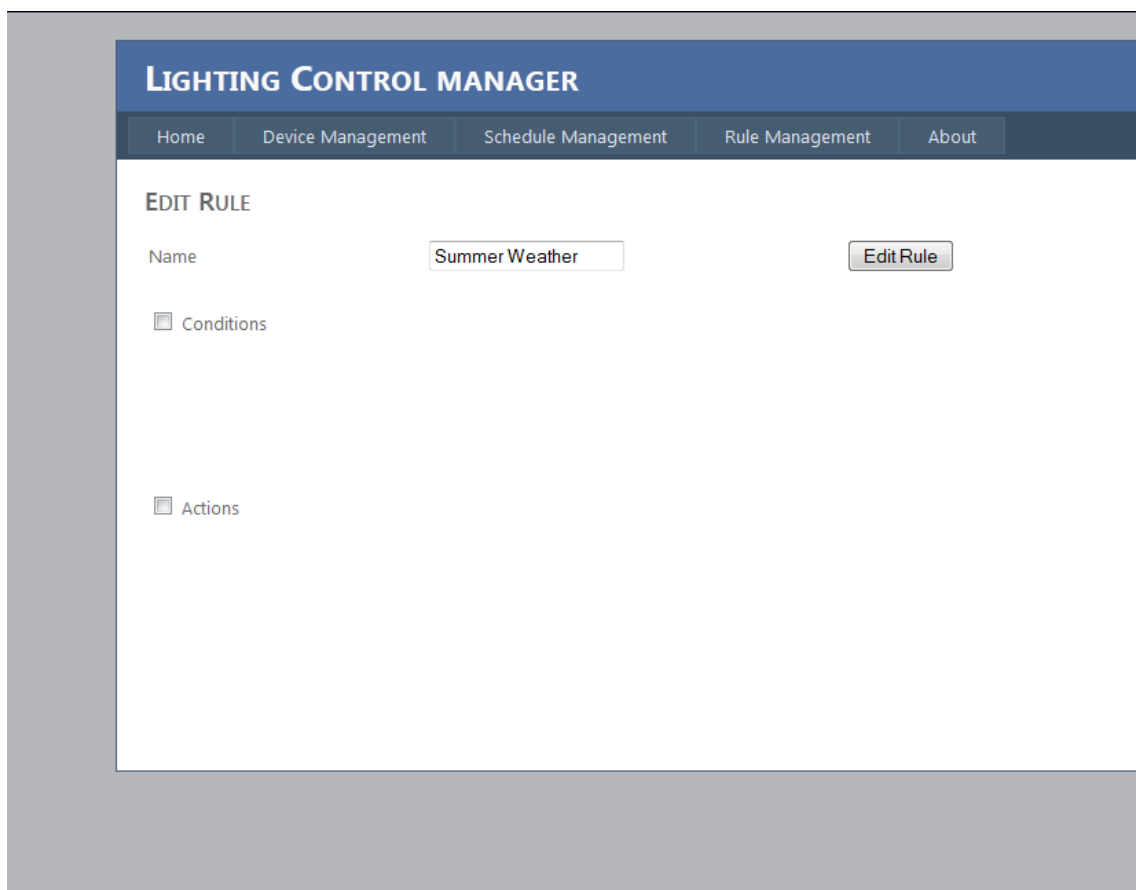


Figure 5.11: Edit Rule page

5.3 Connectivity with the Web Service

The web application has been built using the Visual Studio software which includes the .NET framework, as well as the Web Service, so the connectivity between them is easy to configure. The way to access a .NET web service from a .NET application is by creating a web reference in the application's solution to the web service's WSDL file. This file is created automatically by .NET when the web service is launched. The .NET framework then downloads the specification of the web service and creates an object that contains all the methods of the web service. By instantiating this object in our application, we have direct access to the web service methods, without having to configure any kind of socket connection between the two endpoints.

6 ANDROID APPLICATION: USER INTERFACE

The Android application is focused towards a common user, so it is based on simplicity and intuitiveness.

The main color of the application is a variety of dark grey and black, easy for the eyes, with some green elements indicating activated modules, like devices or rules.

The first subsection describes the Android SDK's layout system, and a second one describes the list of activities of the application. Activity is the Android name for an interface window.

6.1 The layout system

Android SDK offers a very large list of elements to use when building the user interface. As in most common interface developing kits, they are divided in two groups: Views (individual elements) and Viewgroups (containers of Views and/or other Viewgroups).

The way to build a user interface is by creating a Viewgroup structure (see Figure 6.1).

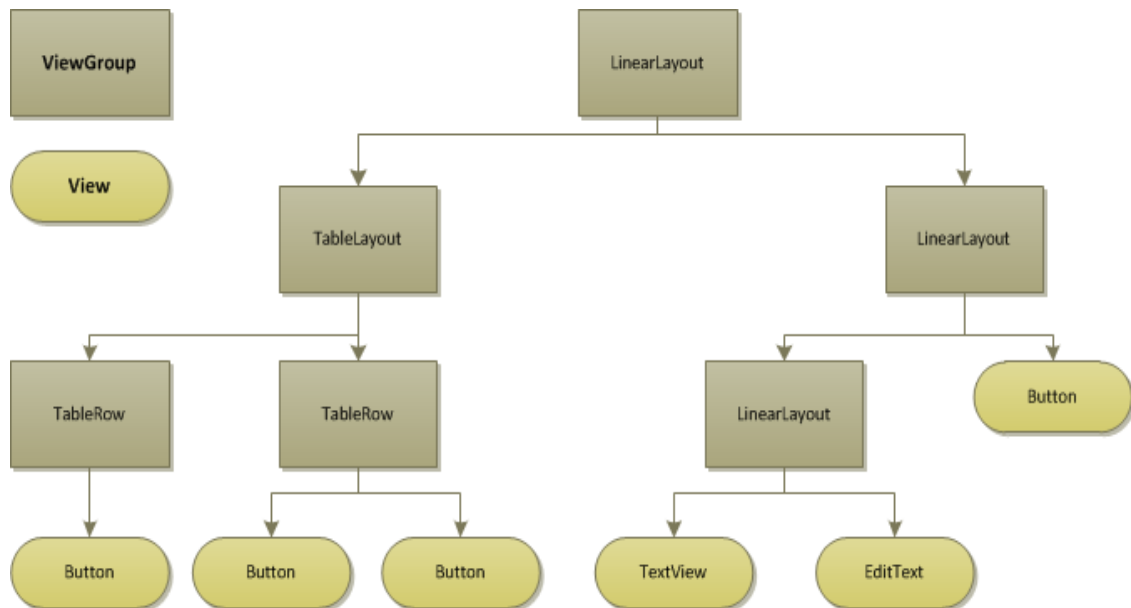


Figure 6.1: Layout system

These layouts can be created dynamically in the code, or stored in a layout XML file.

Android SDK also allows the developer to create their own Views and Viewgroups by implementing their interfaces (or extending existing Views and Viewgroups).

In our Android application, the most used Views and Viewgroups have been:

Views:

- **TextView:** Used to show text information. It can also implement listeners to click events, such as buttons.
- **ImageView:** Used to show pictures and icons, such as the location photo, or the weather icon.
- **EditText:** Used to get text input from the user, such as the rule name when adding a new rule.
- **CheckBox:** Used to get Boolean input from the user, such as the activation of a new rule.
- **Spinner:** Also known as drop-down menus, used to select an item from a list, such as selecting a Device when adding a condition to a rule.
- **Button:** Used to get direct actions from the user, such as turning devices on and off, or adding a rule.
- **SeekBar:** Used to get the brightness level change from the user for a lamp device.

ViewGroups:

- **LinearLayout:** Simple layout that sorts the underneath Views and ViewGroups in a linear way, horizontally or vertically.
- **GridLayout:** Sorts the underneath Views and Viewgroups in a grid of the selected column number. It behaves differently as it only accepts children Views from an Adapter class, not directly.
- **TableLayout:** Sorts the underneath Views and Viewgroups in a table. There has to be a TableRow ViewGroup for each of the desired rows of the table. Each TableRow behaves as a horizontal LinearLayout.
- **ScrollView:** Creates a scrollable area with the underneath Views and Viewgroups.

Each View and Viewgroup has a LayoutParams attribute in which the size of the view is stored.

Android SDK offers two ways to assign screen space to a view.

Direct sizing: You can assign a static size (e.g. 100 pixels) for both the width and the height of a View. You can also set the View to match the parent ViewGroup's space, or let the system assign the view all the area it needs (for instance, size may depend on the number of characters and font for a TextView).

Weight sizing: In the LayoutParams attribute there is a field called “weight”. This weight directly affects the amount of space used by that view compared to other views’ weight. A View with a weight of 2 will have double the size than a View with a weight of 1, while all the Views with the same weight will have the same size.

In order to use this feature, the dimension of the view (width and/or height) that has to be affected by the weight must be set to 0 pixels.

Both size assigning features have their well known advantages and drawbacks. We have made use of both of them.

6.2 Activity List

The application is divided in five main activities (Control, Schedule Management, Rule Management, Security and Scenes). From the Schedule Management activity the user can access the Add Schedule activity, and from the Rule Management activity, the user can access the Add Rule activity. The main activity is the HomeAutomation activity (see Figure 6.2). This scheme was introduced in chapter 3.

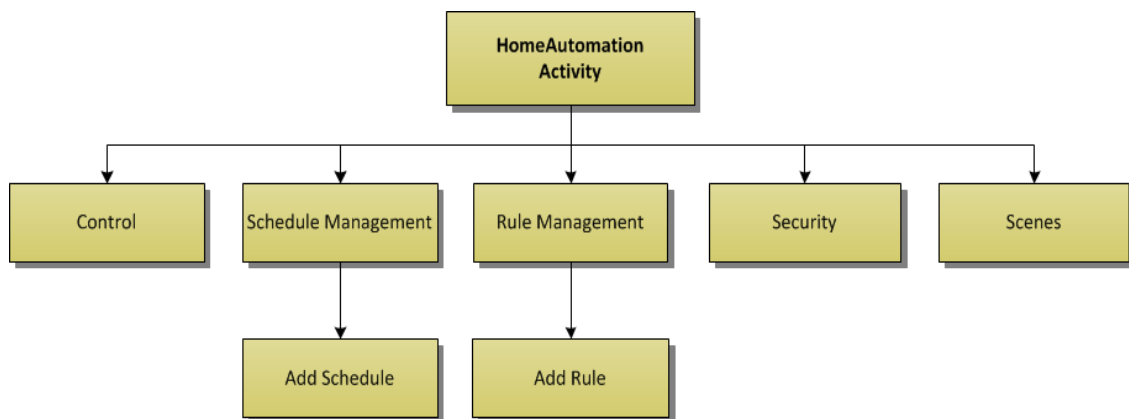


Figure 6.2: Android activity list

The next subsections describe each of the activities.

6.2.1 HomeAutomation Activity

The HomeAutomation activity shows the welcome message and shows a horizontal menu on the top that lets you navigate through the different activities of the app (Control, Schedule Management, Rule Management, Security and Scenes).

6.2.2 Control

The Control activity includes all the manual control abilities of the system.

At first, it will show a list of the available locations on the left panel. When the user selects one of the locations, the main panel will populate with the information and device list of the selected location.

This includes the location name, location image, humidity, light and temperature levels of the location (if there are sensors located in the selected location), and the device list of that location (see Figure 6.3).



Figure 6.3: Control activity

The devices are grouped in three different **categories**: lamp modules, appliance modules and sensor modules. Any of these categories can be expanded or collapsed. Inside these categories the device names and descriptions are shown. The user can also click the device name in order to hide/show the device description. In addition to the devices' names, the interface also shows different buttons to manually control the devices. These buttons are different for each category (sensors don't have any control buttons):

- For **lamp modules**, the interface shows a brightness level indicator and a scrollable bar that lets the user select the desired level for the device. The

level indicator is updated in real time when the user moves the scroll. The interface also shows a button to turn on or off the device.

- For **appliance modules**, the interface shows the button that lets the user turn on or off the desired appliance.

The full list of locations, devices and their status is updated every three seconds.

One of the locations will always be “**All the locations**”, which will show all the devices. This option is selected by default whenever the user enters the Control activity. In this case, the user will be able to select how the devices are grouped, either by their category (lamp, appliance, and sensor) or by location. The main panel will show some system statistics, such as how many devices are installed, how many lamps, appliances and sensors, and how many of these devices are not responding. If the user clicks in this last message, it will show the list of devices that are not responding (see Figure 6.4).

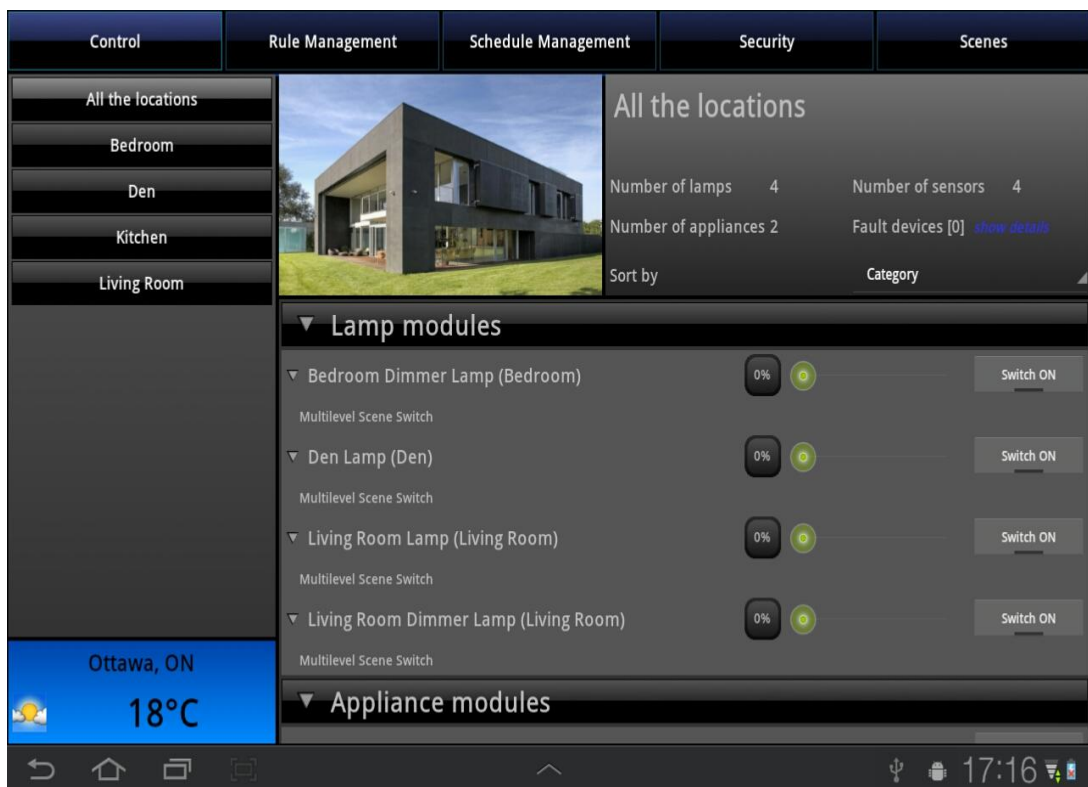


Figure 6.4: All the locations

The Control activity also offers a weather view that shows the current temperature and the weather status. If the user clicks on the weather widget, a popup will ask the user to insert the name of the city he wants to see the weather of.

If the user changes the orientation of the tablet to portrait mode, the left panel will hide to offer the user a better view of the device list and the location info.

6.2.3 Rule Management

The Rule Management activity shows a list of all the Rules added to the system. For each Rule in the system this window will show a container. The background color of this container can be either black if the Rule is not active, or green if the Rule is active (see Figure 6.5).

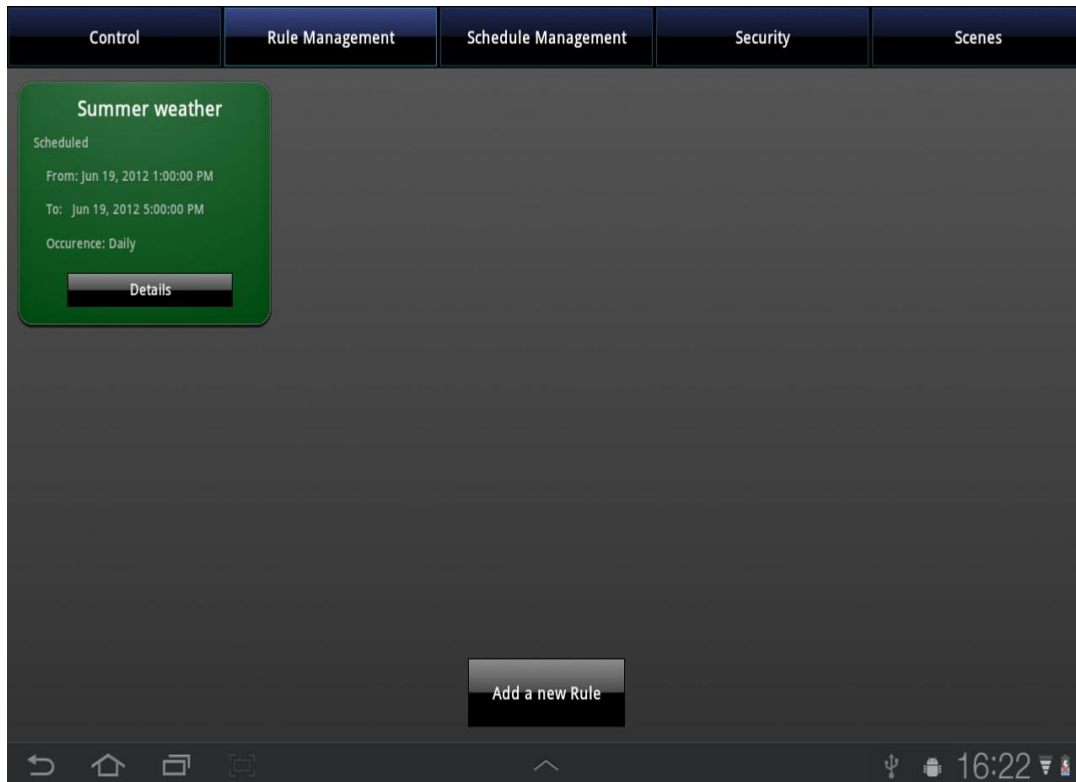


Figure 6.5: Rule Management activity

This container shows the Rule's name, whether the Rule is scheduled or not, and the period of time and occurrence of the Rule if it is scheduled. In addition to that, each container has a button to show the details of the Rule. After pressing this button, a pop-up window will show all the Rule's details (see Figure 6.6).

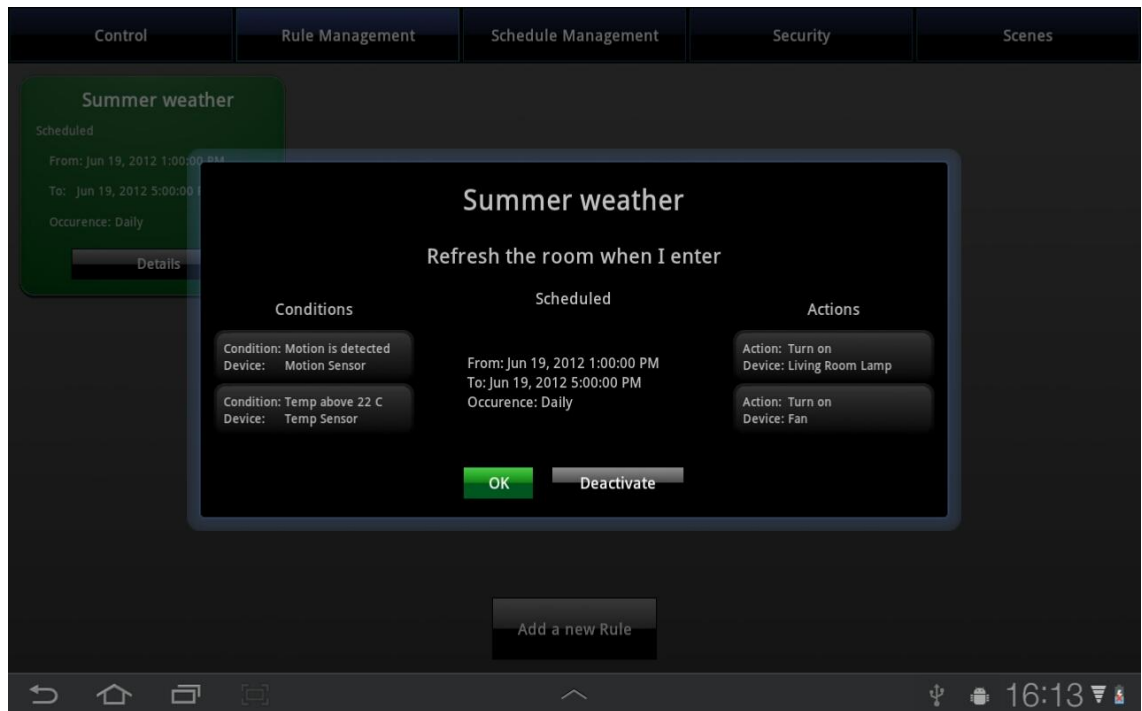


Figure 6.6: Rule details

If the user long-presses a container, this will enter the edit mode. A red-cross button will appear in the right top corner of the container to let the user remove the Rule, and the 'Details' button will change to 'Edit Details'. This button will let the user edit that Rule's details. The detail operation is done via the Add Rule activity, explained next.

6.2.4 Add Rule

The Add Rule activity lets the user add new Rule to the system. It will show a field to select the Rule's name and description, as well as two columns to add conditions (left side of the window) and Actions (right side of the window) to the Rule. The user can choose whether the Rule is constantly checking the conditions, or only during a period of time. The user can also choose if this period is refreshed daily when it ends, and if the Rule is going to be activated from the beginning or not.

If the user comes from the Add a new Rule button from the Rule Management activity, all the fields will be empty (see figure 6.7).

The screenshot shows a mobile application interface for adding a rule. At the top, there are five tabs: Control, Rule Management (selected), Schedule Management, Security, and Scenes. Below the tabs, the interface is split into three main vertical sections. The left section contains two buttons: 'Add condition' and 'Remove condition'. The middle section contains four input fields: 'Name', 'Description', 'Scheduled' (with a checkbox), and 'Activate' (with a checkbox). Below these fields is an 'Add Rule' button. The right section contains two buttons: 'Add action' and 'Remove action'. At the bottom of the screen, there is a navigation bar with icons for back, home, recent apps, and a home indicator, along with system status icons and the time 15:40.

Figure 6.7: Add a Rule

If the user comes from the Edit Rule button it will show all the fields already filled with the Rule's information (see Figure 6.8). After editing the desired fields the user can click on the 'Save Changes' button to submit the changes to the system.

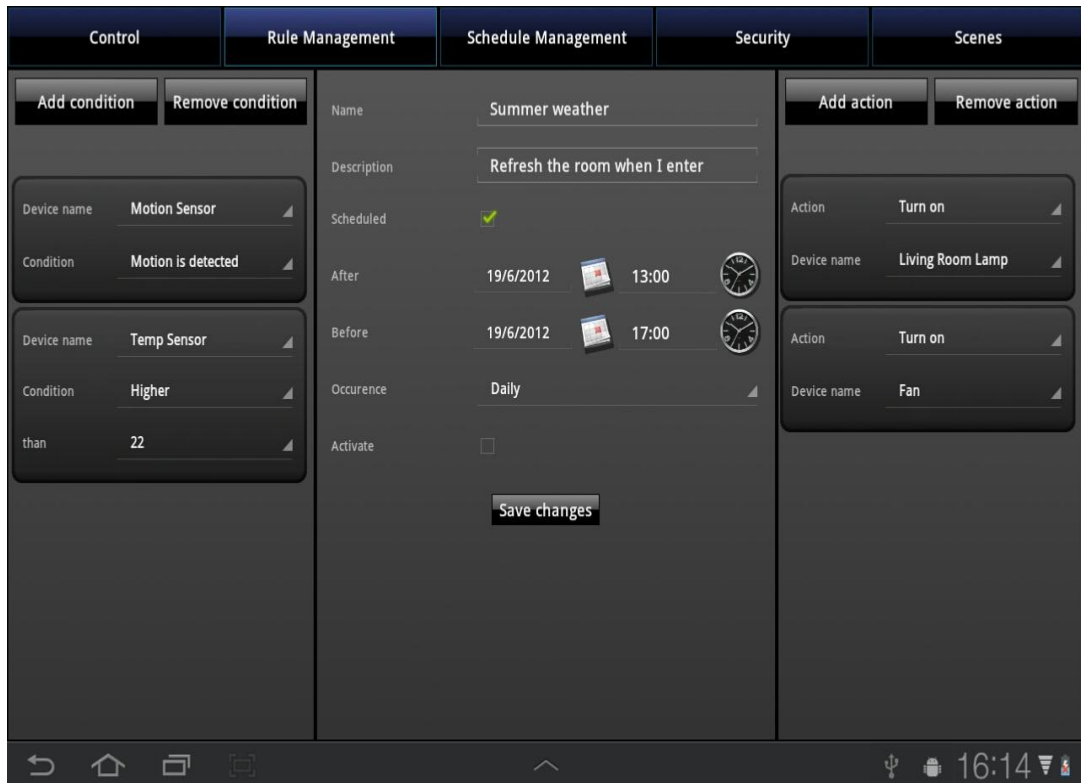


Figure 6.8: Edit a Rule

6.2.5 Schedule Management

The Schedule Management activity shows the list of Schedules and lets the user add or edit Schedules.

The activity shows all the Schedules that exist in the system. For each Schedule in the system, it shows a container with the Schedule's name, the date of execution and the occurrence (see Figure 6.9).

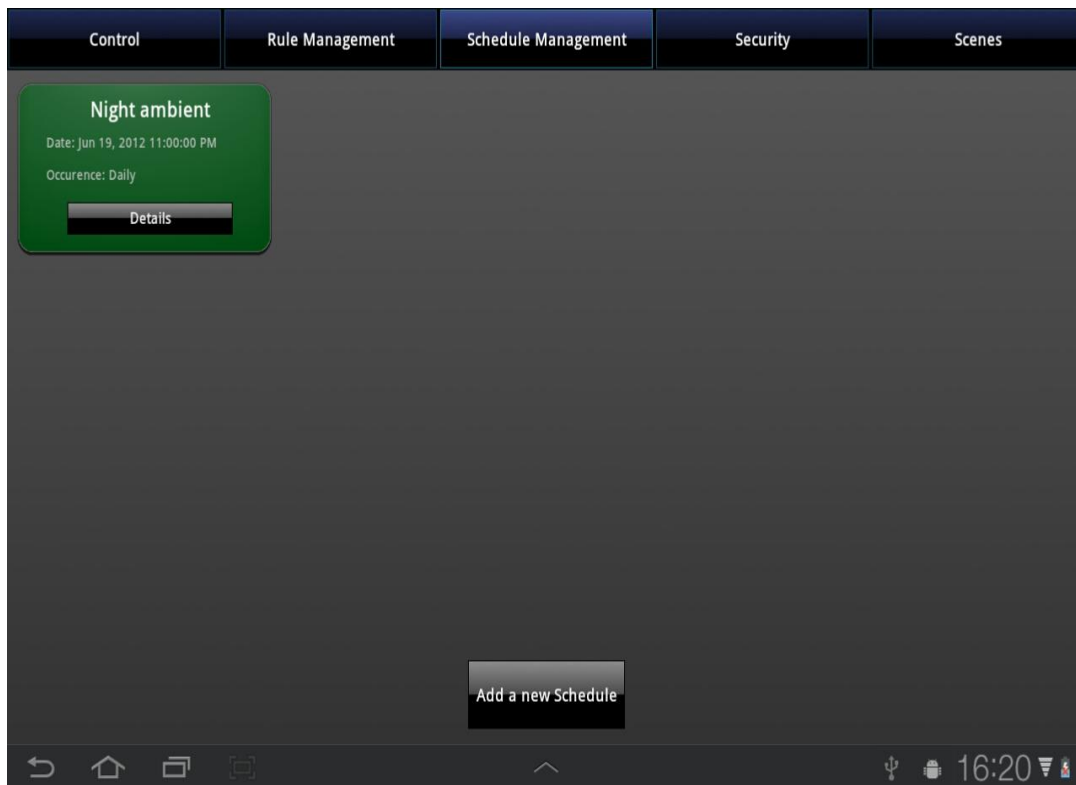


Figure 6.9: Schedule Management

It also contains a button that will show a popup message with the detailed information of the Schedule, including all the information from the container, plus the Schedule description, and the full list of actions it executes (see Figure 6.10).

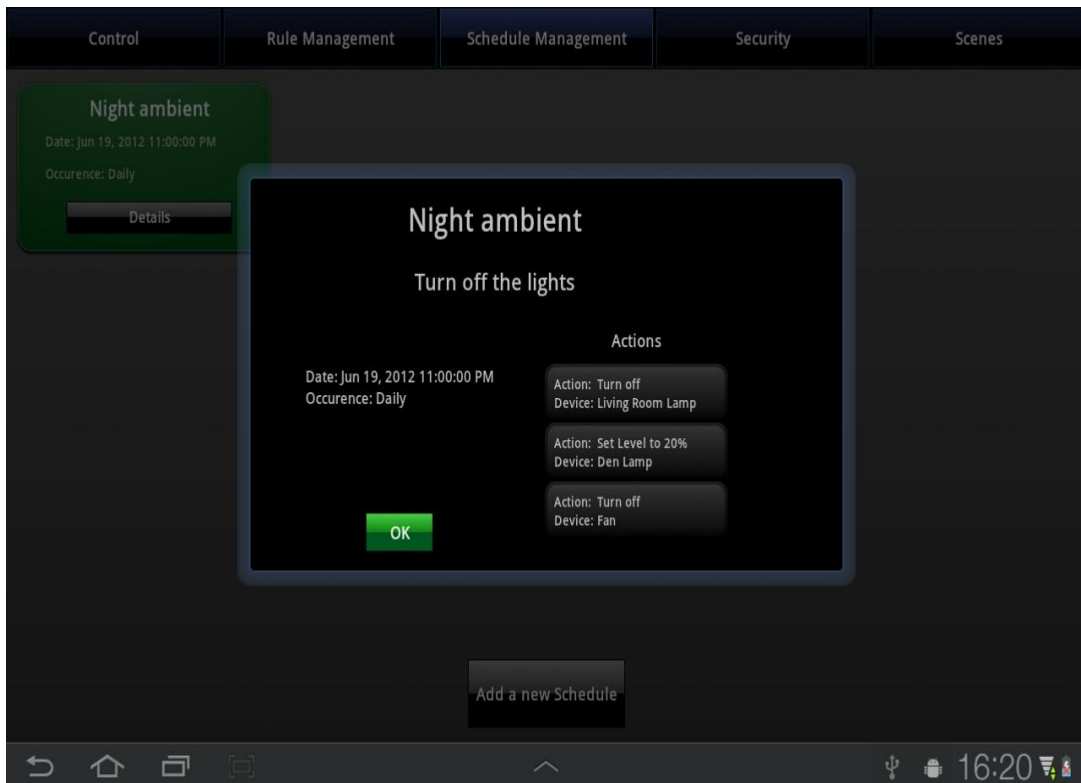


Figure 6.10: Schedule details

If the user long-clicks the container it enters the edit mode. It shows a button that lets the user delete the Schedule from the system (after a confirmation popup), and the ‘Show Details’ buttons transforms to ‘Edit Details’. This button will lead to a new activity that lets the user change any part of the Schedule. This is done via the Add Schedule activity.

In addition to the Schedule list, in the bottom of the interface there is a button to add a new Schedule that leads to the Add Schedule activity.

6.2.6 Add Schedule

The Add Schedule activity offers two different functionalities depending on the button that led the user there.

If the user clicked the Add Schedule button from the Schedule Management activity, it will show a list of fields to enter before submitting the Schedule to the system (see Figure 6.11).

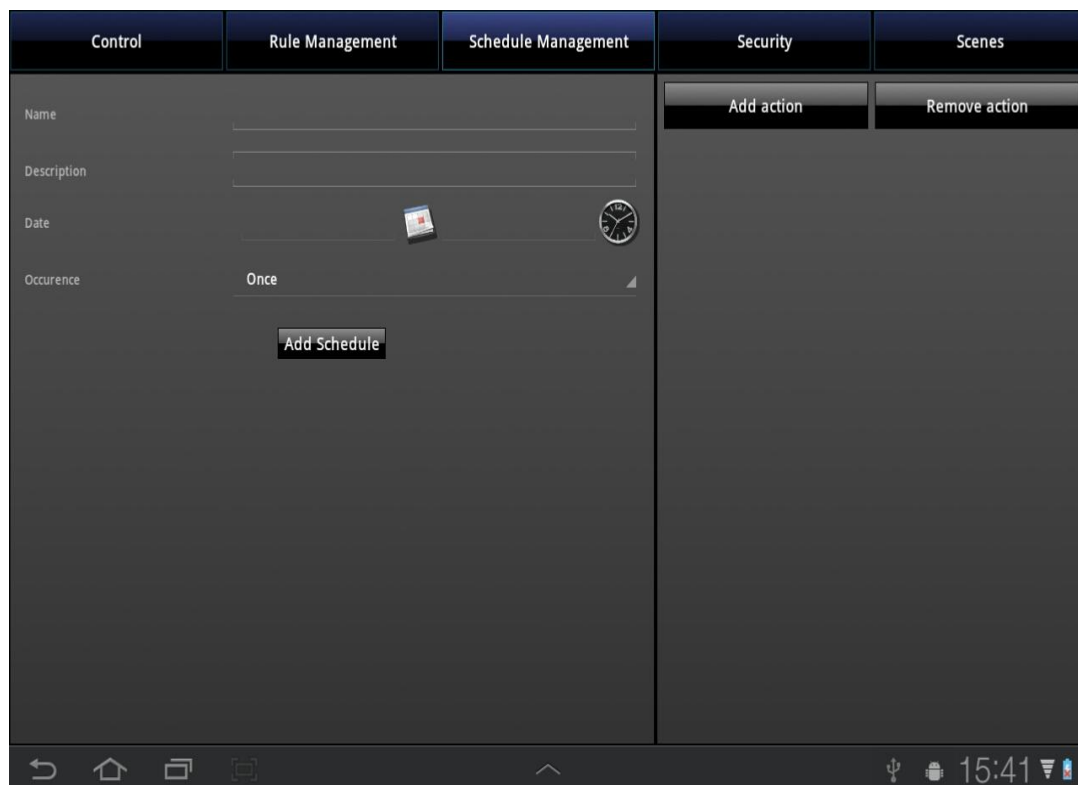


Figure 6.11: Add a Schedule

This includes a name field, a description field, a date and time selector (via a calendar view and an hour view), and an occurrence drop-down selector. On the right side of the screen there is a panel that lets the user add or remove actions to the Schedule. Each action has an action selector indicating the action (turn on, turn off, etc.) and a device selector.

Once all the fields have been filled, the user can click on the “Add Schedule” button on the bottom side of the screen to add the Schedule to the system.

If the user came to this activity via the “Edit Details” button from one of the Schedules container in the Schedule Management activity, it will show the same interface, but all the fields will be already filled with the information of the actual Schedule, so the user only has to modify the desired fields and press the “Edit Schedule” on the bottom to edit the Schedule (see Figure 6.12).

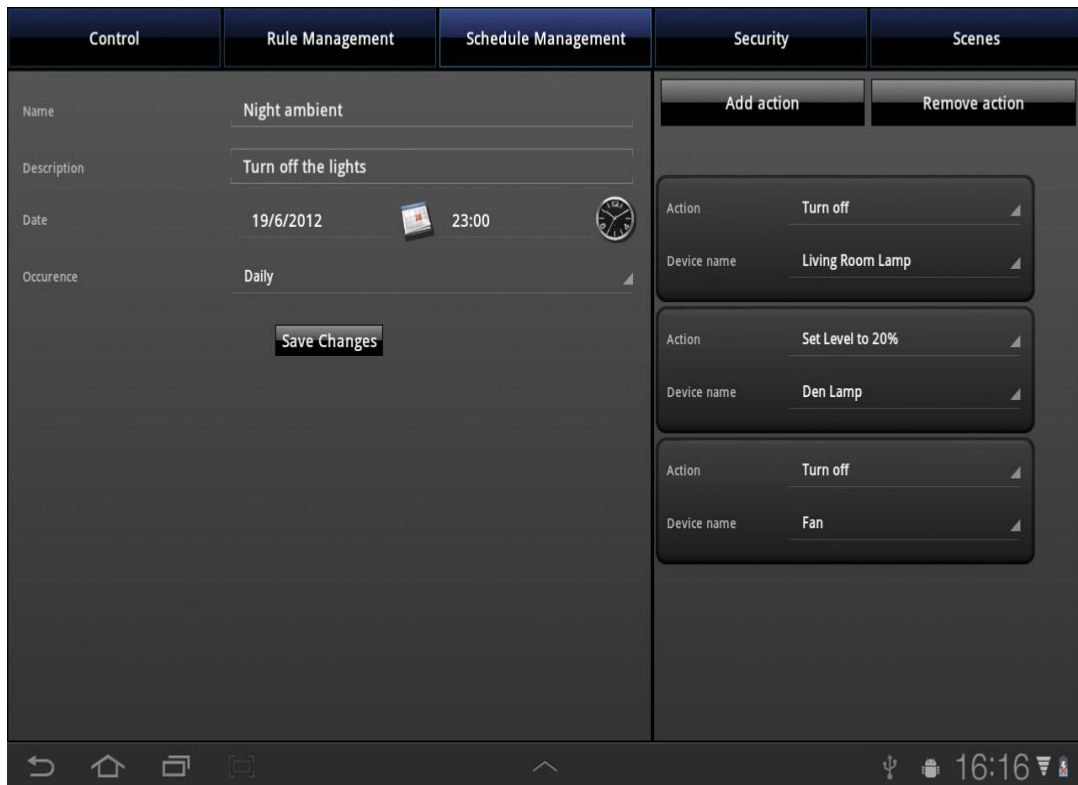


Figure 6.12: Edit a Schedule

In any of these cases, if there is a problem adding or editing the Schedule, a warning message will appear. If the Schedule is correctly added or edited, it will automatically return to the Schedule Management activity.

If the user changes the table orientation to portrait mode, instead of dividing the interface horizontally, it will be divided vertically, so the action list panel will be on the bottom side, and the information panel on top of it. Also, the action list will be horizontally scrollable instead of vertically scrollable.

6.2.7 Security

The Security activity will let the user watch and control the live-streaming cameras installed in the system. It will show a group of four cameras on the right side, while on the left side it will show a list of all the cameras installed and a panel from where the user can control the selected camera. Once the cameras are loaded, the user will be able to select one of the cameras from the left menu, and from the control panel below the cameras, control the direction of the camera. Because of the heavy data communication, a certain delay between the real image and the camera image is expected.

The framework for this functionality is part of the future work of the architecture and is not available at the moment of this document's writing, but the base user interface design has been made (see Figure 6.13).

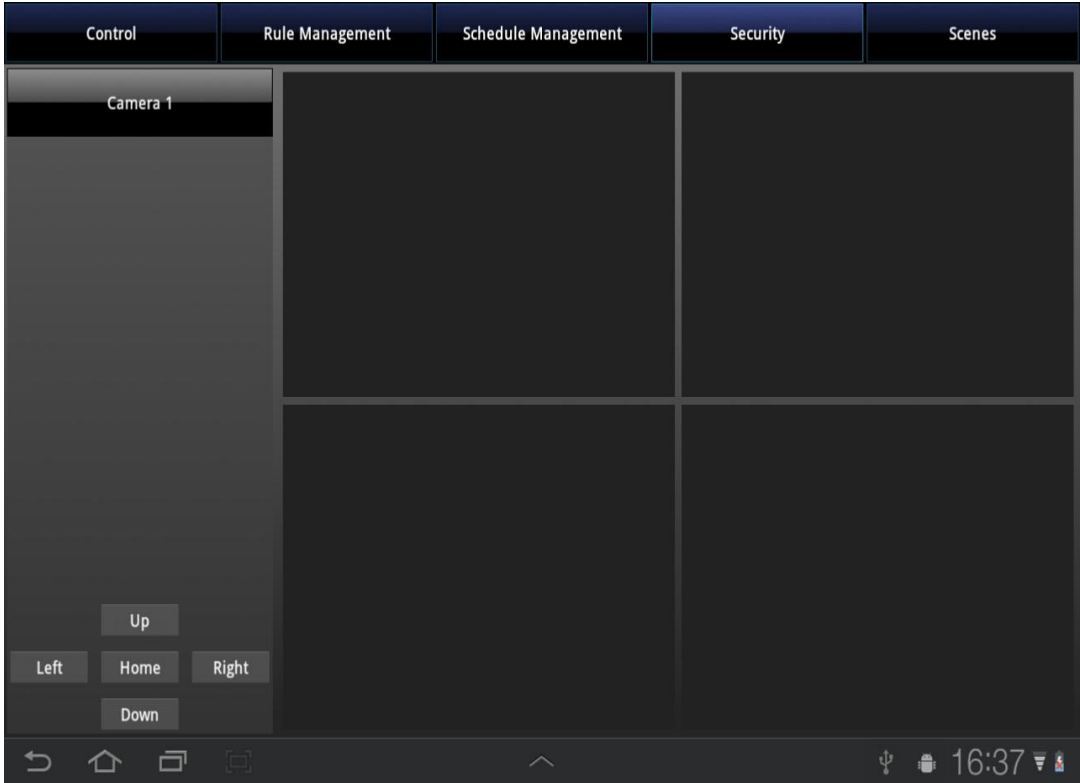


Figure 6.13: Security activity

6.2.8 Scenes

The Scenes activity is similar to the Schedule and Rule Management activities. It will show a list of all the Scenes added to the system. In this case, the user can single-touch any of the Scenes to load them. There is also a button in each of the containers that lets the user instantly load the Scene (see Figure 6.14).

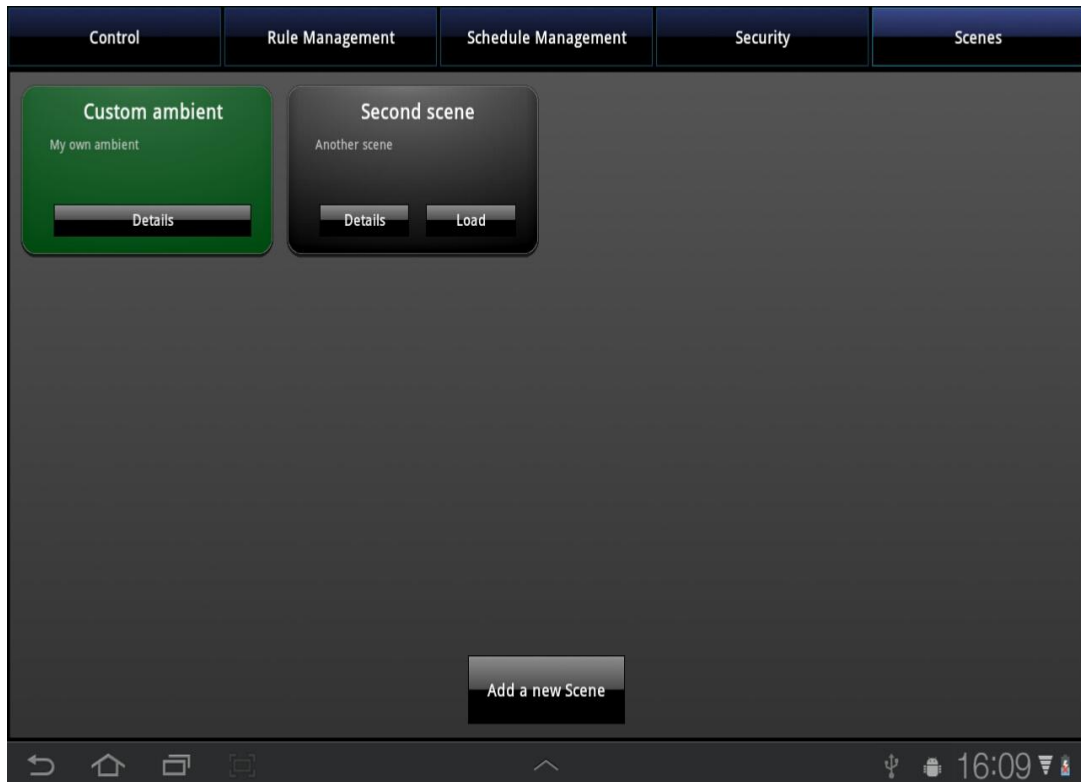


Figure 6.14: Scenes activity

If there is a Scene currently loaded, its background will be green instead of black, and the Load button will become invisible. From this activity the user can also add a Scene, by either pressing the Add new Scene button in the bottom side of the window.

7 ANDROID APPLICATION: CLASSES AND CONNECTIVITY

This chapter explains the classes used to represent the system objects, and the connection between the Android application and the Web Service.

The design order of the application doesn't match the order of the chapters in this document. This chapter was designed before the previous one (user-interface), but in order to ease the reading of this document, their order has been reversed.

7.1 Classes

As there is too much data to manage directly, we have defined a series of classes to manage different information.

- **Device:** for each device we store the name, display name, description category, location, type, status, level, and protocol.
- **Location:** for each location we store the name, description, image link, the list of devices installed in that location, and also the temperature, humidity and lighting levels, when an appropriate sensor is installed in that location.
- **Schedule:** for each Schedule we store the id, name, description, scheduled date and time, and the list of actions. Each action is stored in an Action object, explained below.
- **Rule:** for each Rule we store the id, name, description, whether it is scheduled or not, the date, time and occurrence if it is scheduled, the lists of conditions and actions, and whether it's activated or not. Each condition is stored in a Condition object, and each action is stored in an Action object, both explained below.
- **Condition:** for each condition we store the device name, and a String defining the condition, such as "Level below 80%".
- **Action:** for each action we store the device name, and a String defining the action, such as "Set Level to 50%".
- **Scene:** for each Scene we store the id, name, description, and the list of devices with their saved statuses and levels. This list is stored as a list of actions.

7.2 Connectivity

One of the Android's features is that all the networking processes must be done in a separate thread from the main one, in which the user interface operations run. This is done not to block the interface from user action if the networking processes fail or get blocked.

Android SDK offers an easy way to create a new thread for networking purposes by extending the AsyncTask interface. This interface offers a method (doInBackground) that we can implement to do the networking process.

Another feature of the Android devices is that any user interface access or modification has to be done in the Main thread, so it's not possible to modify the interface in the networking thread. Again, the Android SDK offers an easy way to work this around, by sending a Runnable object to the runOnUiThread method, available anywhere inside an Activity, even in a networking thread.

7.2.1 Web Service methods used

From the list of methods the Web Service offers, our Android application makes use of the following ones:

- GetDeviceList
- GetScheduleList
- GetLocationList
- GetRuleList
- GetSceneList
- GetConditionList
- GetActionList
- SwitchStatus
- SetLevel
- AddRule
- AddSchedule
- CreateScene
- EditRule
- EditSchedule
- RemoveSchedule
- RemoveRule
- RemoveScene
- ExecuteAScene

7.2.2 Calling the Web Service methods

In order to send requests to the Web Service and receive responses from the Web Service we had to use an external library to create Soap Objects in order to encapsulate the parameters of the Web Service methods.

The external library we used is Ksoap2-Android⁴. The way to call a Web Service method is by creating an Envelope (SoapSerializationEnvelope class), in which we add a SoapObject that includes the Web Service method name and the required parameters. After filling the envelope, we send it using the HttpTransportSE class.

After calling the Web Service, the response is added to the envelope, so we can extract that response by calling the method `getResponse` from the SoapSerializationEnvelope class. This method returns a SoapObject with the response from the Web Service. If the call failed, this method will return a SoapFault object instead, and if the response is just a single primitive value, like an integer, it returns a SoapPrimitive object instead.

The next sample code presents the main structure to call a Web Service.

```
HttpTransportSE androidHttpTransport = new HttpTransportSE(WEBSERVICE_URL);
SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(
    SoapEnvelope.V11);

SoapObject request = new SoapObject(WEBSERVICE_NAMESPACE,
    WEBSERVICE_METHOD);
request.addProperty(PARAMETER_NAME, PARAMETER_VALUE);
envelope.setOutputSoapObject(request);
envelope.dotNet = true;

try {
    androidHttpTransport.call(WEBSERVICE_NAMESPACE + "/"
        + WEBSERVICE_METHOD, envelope);

    SoapObject resultsRequestSOAP = (SoapObject) envelope.getResponse();
} catch (Exception e) {
    e.printStackTrace();
}
```

⁴ <http://code.google.com/p/ksoap2-android/>

8 CONCLUSIONS

The work done in this project has proven to be very successful, as both my tutor and the Company we were collaborating with were very interested in the results achieved.

I am also quite happy with the outcome. I have learned many new and hot technologies and my English communication skills have improved a lot. I have proved that I can work in an international environment.

I have gotten to learn a lot about tablet application development, web services and home automation protocols.

The project itself is very extensive and it has a lot of possible improvements. Because this is a long term project, some of the functionalities of the system are not yet implemented, but we have built a solid basis from which all these functionalities can grow.

The system has proven to be exciting, interesting, challenging and useful. We hope that with some additional investment it will make a useful and profitable product.

9 BIBLIOGRAPHY

- Android Developers page - <http://developer.android.com>
- StackOverflow page - <http://stackoverflow.com/>
- CodeProject page - <http://www.codeproject.com/>

ANNEX I

PROJECT ENVIRONMENT

This annex includes the initial description of the system my tutor presented to me at my first day in Ottawa.

1 HOME AUTOMATION

Nowadays, there are some systems that exist which automatically control the lighting system in apartments, residential, and commercial buildings. However, these systems offer basic functionalities, are inconvenient, and consume extra energy. Usually, the user has to configure the system to automatically switch on/off the lights. Since the times for sunset and sunrise change during the course of the year, it makes it harder for the users who have to adjust and configure the system regularly.

Our goal is to provide a solution, which automatically control the lighting by schedule, sunrise and sunset sensor activation, or could be customized based on users' preferences. The system could be then extended to comprise heating and ventilation, security system, fire alarm, to achieve a complete Building Management System BMS. Our solution should cover the following scenarios and meet other requirements described below.

1.1 Scenario 1: Manual Control

It is the end of the day, and Joe wants to read his favorite book before he goes to sleep. He wants to manually control the lights in his bedroom to create a lighting scene for his reading activity. Using his portable device, he switches on one of the light, and sets another dimmable light to a different level. Since Joe reads every night before he goes to sleep, he saves the lighting settings so he can create the same lighting scene with one single button using his portable device. Now Joe is asleep, so he selects "sleep mode" lighting scene that turn off all the lights in the bedroom.

1.2 Scenario 2: Automatic Control by Schedule

Joe wants to configure the lighting system installed in his backyard to be automatically controlled. Using his portable device, he configures the system to turn off one of the lights and set another dimmable light to a lower brightness level from midnight to 6 am. He also configures the system to automatically increase the brightness level in case a motion is detected in the backyard. The system then automatically resets the dimmable light's brightness level to its previous state when there is no activity in the backyard.

1.3 Scenario 3: Automatic Control by sunrise/sunset sensor activation

Adam, Joe's older brother, is not a techy guy and does not have the ability to configure the system himself. However, he wants the lighting of his living room to be automatically controlled. During installation, the system is configured to automatically control the lights using a light sensitivity sensor. Based on the brightness level, the system automatically dims the lights to a lower or higher level. For instance during sunset, the level of lights increases until it reaches 100% of its capacity.

1.4 Other Requirements

- The system should support the following components: motion detection sensor, light sensitivity sensor, Lamp/appliance modules and the main component.
- A centralized unit, which is hooked to the components listed above, can be accessed using an application running on iOS or Android operating system.
- The system should support different types of control and automation networks such as X10, Insteon, and ZWave.
- The system provides the flexibility to add components into the system whenever needed, such as an additional motion sensor, or additional X10 lamp module component.
- The system provides the functionality to manually control the lights, or automatically by schedule, by sunset sunrise sensor activation, or combination of both.

1.5 Simple solution to kick off

We must start by building a simple system hooked to 1 motion detection sensor, 1 light sensitivity sensor, 1 X10 main component that is connected to 2 lamp modules. We must also build an application running on iOS and Android in order to access the system and control the lighting.

2 SURVEILLANCE SYSTEM

Many businesses and homeowners are recently increasing their security level by installing security cameras. However, these solutions offer to the customers some basic functionality for controlling and monitoring their premises.

Our goal is to provide a solution allowing the user to easily control, monitor, or configure the cameras using their portable device wherever the user is located. The solution offers some basic functionality such as changing cameras view, and controlling each camera installed. The solution offers additional features such as notification messages, which notify the user when an incident take place and records its events. Our solution should cover the following scenarios and meet other requirements described below.

2.1 Scenario 1: Monitoring, and Controlling Security Cameras

Joe, who owns a villa, is going for a vacation for few weeks. An IP camera operates each room in his villa. While he is away, Joe is able to access the list of the cameras installed in his villa using his portable device. He selects one of the cameras from the list, and he is able to get a real view of the camera that Joe picked. Joe has the flexibility to pan, tilt, and zoom each of the cameras using the application installed on the portable device.

2.2 Scenario 2: Notification message

Since Joe is on vacation, he does not have the time to check regularly the cameras installed in his villas. Therefore, he configures the system to automatically notify him by sending a text message on his portable device when an incident takes place. While Joe is having dinner with his wife, a notification message is received on Joe's portable device. Joe checks the details of the message and he is notified that there is an activity detected in the living room. Immediately, Joe launches the application on his portable device and checks the saved picture of the incident. Fortunately, it was his brother Adam who wanted to get the DVD movie that Joe forgot to give it back before he went for his vacation.

2.3 Other requirements

- The system should support the following components: IP camera, and motion detection sensor.
- A centralized unit hooked to different type of components (described above), can be accessed using portable device on iOS or Android
- The system provides the flexibility to add additional camera into the system, or additional component such as motion detection sensor.

ANNEX II

ANDROID APPLICATION'S OWNER'S MANUAL

The Android application is divided in five tabs (Control, Schedule management, Rule management, Security and Scenes). All these tabs can be accessed through the top menu buttons.

This manual will guide you through all the functionalities available through the Android application, which includes:

1	Control	2
1.1.	View the locations and the devices	2
1.2.	Manually control a device.....	5
2	Schedule Management	8
2.1	What is a schedule?.....	8
2.2	View the available schedules	8
2.3	Add a Schedule	10
2.4	Edit a Schedule	12
2.5	Remove a Schedule.....	13
3	Rule Management	14
3.1	What is a rule?	14
3.2	View the available rules	14
3.3	Add a Rule	16
3.4	Edit a rule.....	20
3.5	Remove a Rule.....	21
3.6	Activate/Deactivate a Rule	22
4	Security Cameras	24
4.1	Watch the cameras	24
4.2	Control a camera.....	25
5	Scenes	26
5.1	What is a Scene?	26
5.2	View the available scenes	26
5.3	Add a Scene	27
5.4	Remove a Scene.....	28
5.5	Load a Scene	29

1 CONTROL

1.1. View the locations and the devices

In the Control window you will see a menu on the left showing all the locations available in the system. One of the locations will always be “All the locations”, selected by default, which contains all the devices. Once you select a location, the right panel will be populated with the location information (such as image, humidity, temperature or light, if sensors are available) and the list of devices installed in that location. The list of devices will be grouped by category (lamps, appliances, and sensors) (see Figure 1).

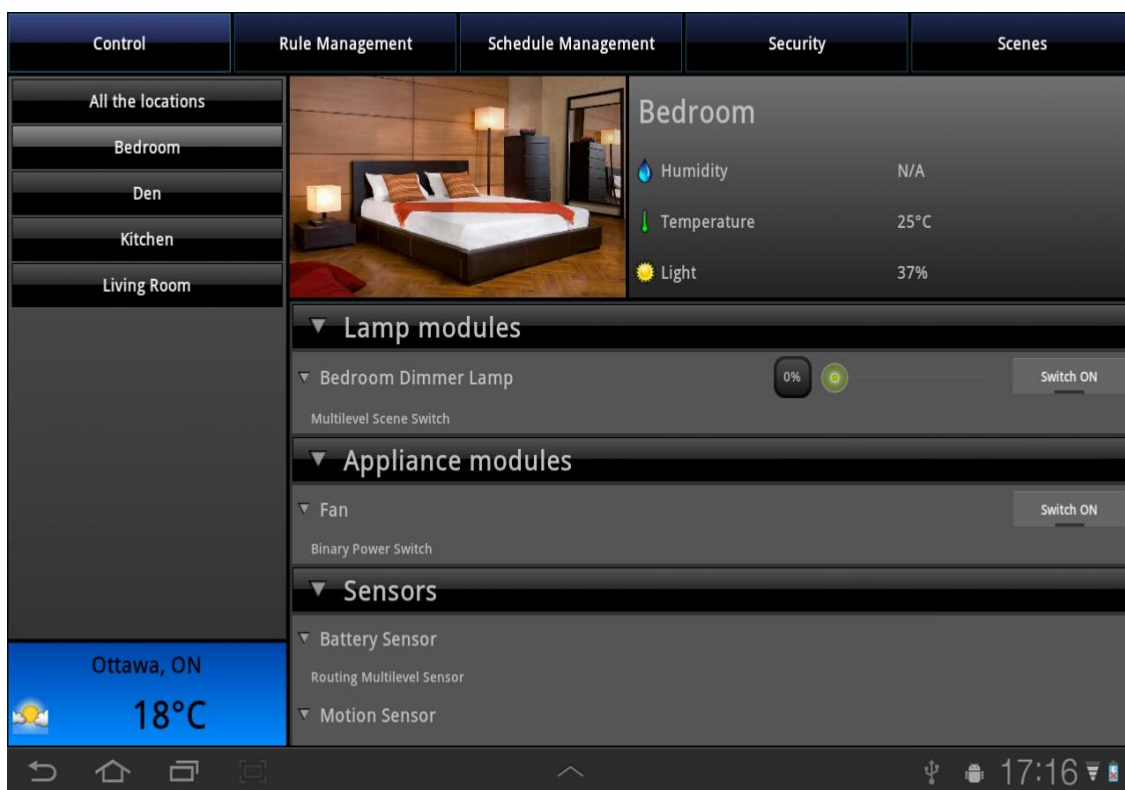


Figure 1 : Location details

You can expand and collapse every category by pressing on the category tab. You can also show and hide the device description by clicking on the device name.

If “All the locations” is selected, instead of the location information, the right panel will show statistics about the system, such as number of lamps, appliances and sensors, and number of faulting devices. The list of devices will now be grouped by either their category or their location, controlled by the drop-down menu just above the list (see Figure 2).

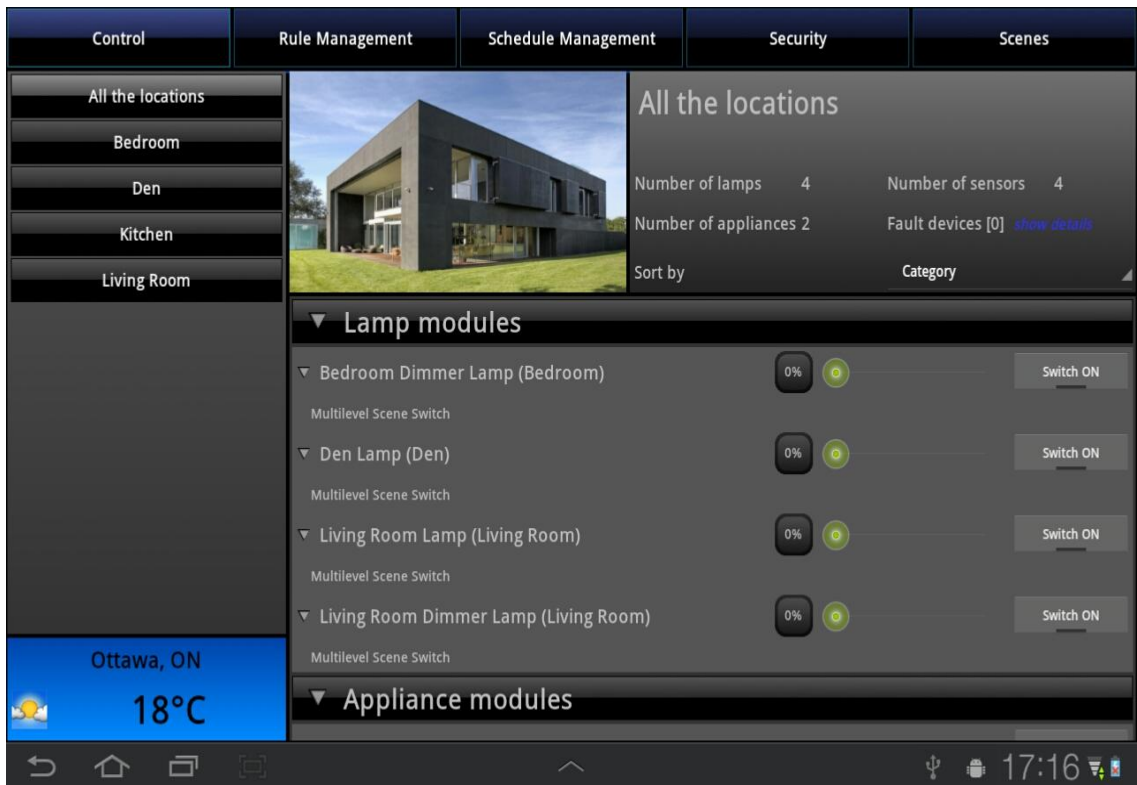


Figure 2: All the locations

You will also notice that there is a small blue tab in the left bottom corner showing the actual weather of a city. You can change this city by pressing the tab. It will ask you to enter a new city (see Figure 3).

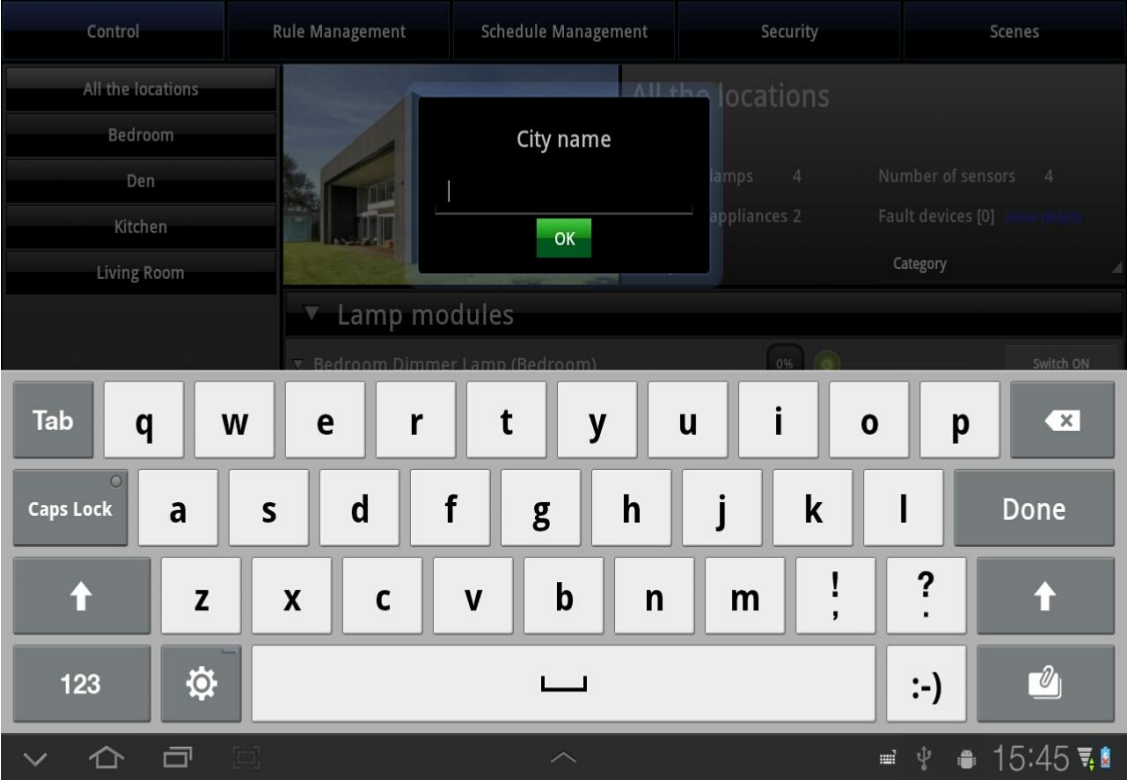


Figure 3: Change weather tab city

1.2. Manually control a device

The controls of the devices vary depending of the category of the device.

For lamp modules there are two options, you can either change the brightness level by moving the green circle across the line (see Figure 4), or switch the lamp on and off by clicking the button (see Figure 5).

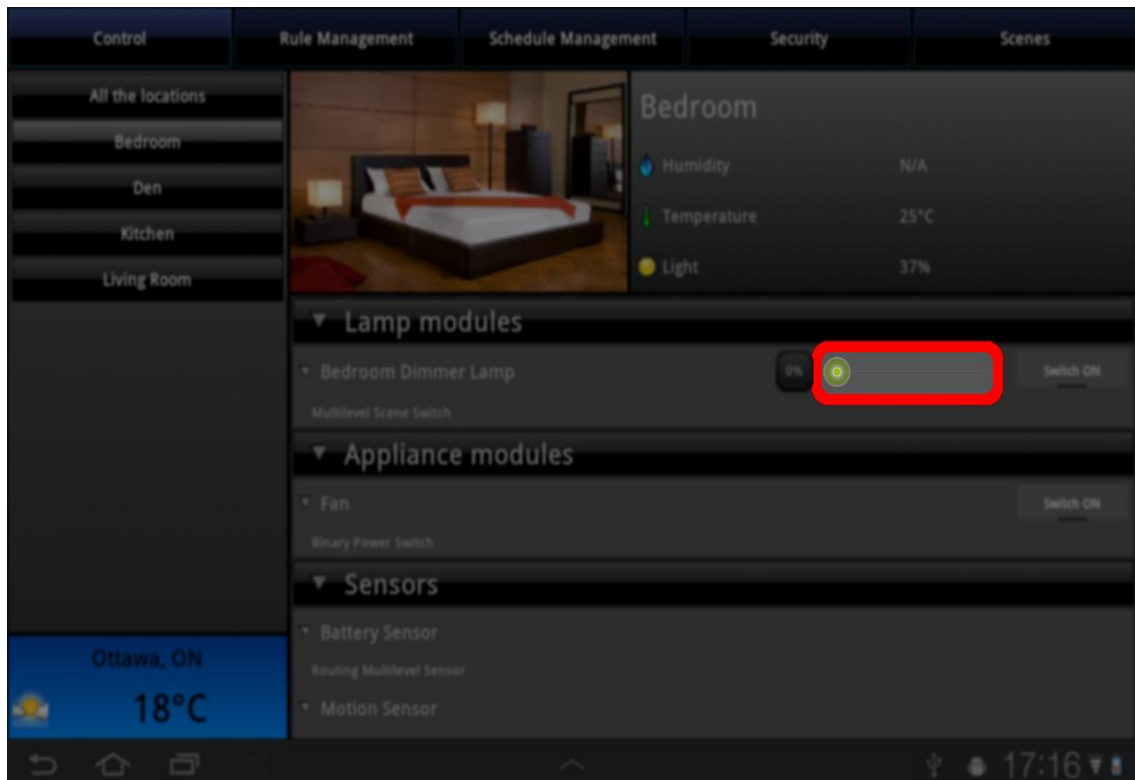


Figure 4: Set lamp brightness level

If you switch off a lamp after assigning a level, once you click the button again to switch it on, it will automatically set the brightness to that level. If the level is set to 0 and you switch the lamp on, it will automatically assign the level to 50%.

For appliance modules, you can switch on and off the appliance by clicking the switch button (see Figure 6).

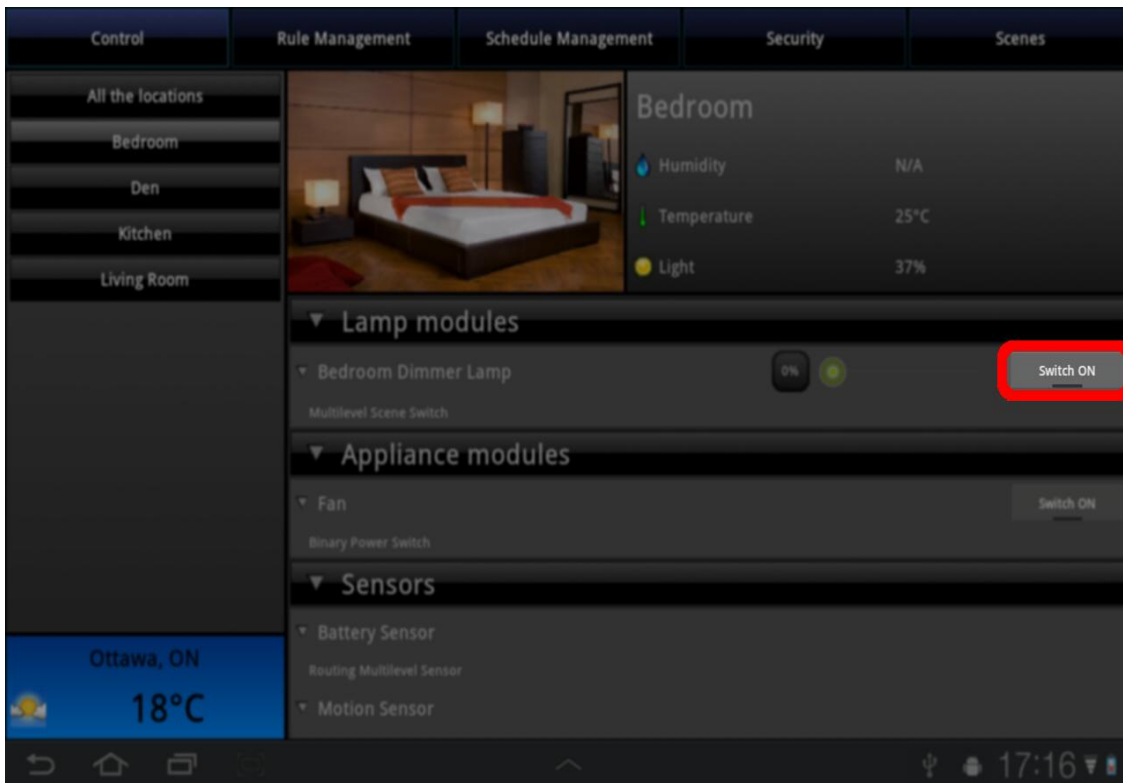


Figure 5: Switch a lamp

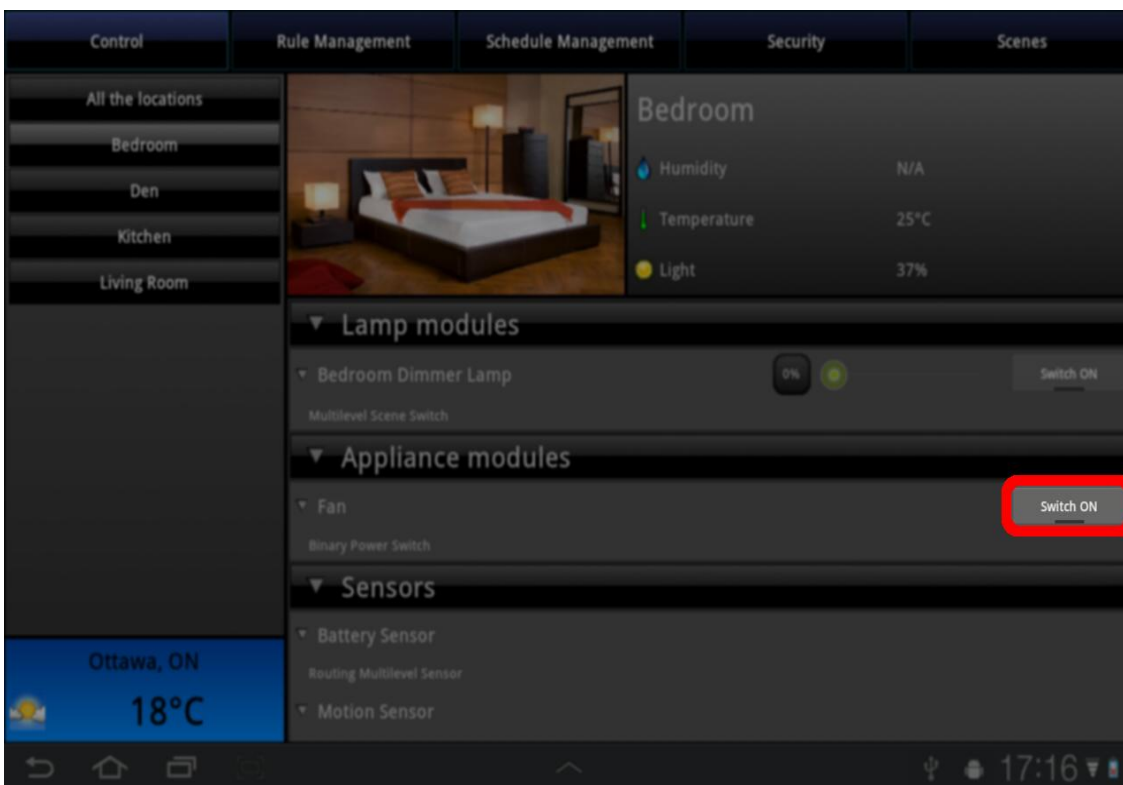


Figure 6: Switch an appliance

Note that once you click on a switch button, it will not let you click it again for a few seconds. This is because there is a certain delay between the button click and the actual switch action. There is also a delay before the system acknowledges the status change of the device and shows it in the screen.

To avoid overloading the system and the device, the switch button will be disabled for five seconds after every click.

Normally it will take the system around 4-5 seconds to acknowledge the status change and show it in the screen.

All the devices and locations are being refreshed every three seconds while on the Control tab.

2 SCHEDULE MANAGEMENT

2.1 What is a schedule?

A schedule consists of a date and an hour in which a series of actions must be executed. The schedule also specifies an occurrence, so when it executes, it automatically refreshes the date and hour to execute in a future time. The system allows the schedule to be configured to execute once, hourly, daily, weekly, monthly, or yearly.

2.2 View the available schedules

In the Schedule Management window, the application will show the list of schedules actually installed in the system. For each schedule installed you will see a green container including the schedule name, the date and time of the schedule activation, and the occurrence of the schedule (see Figure 7).

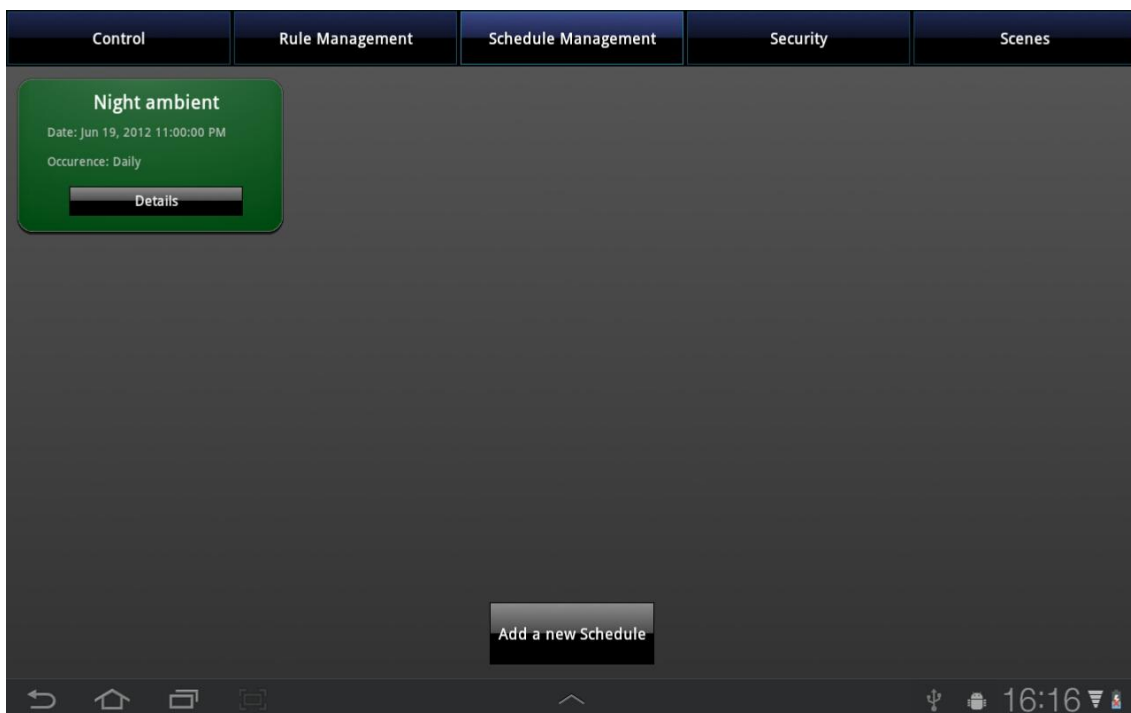


Figure 7: Schedule list

It will also contain a button that will show you all the details of the schedule: name, description, date, time, occurrence and the list of actions it executes (see Figure 8).

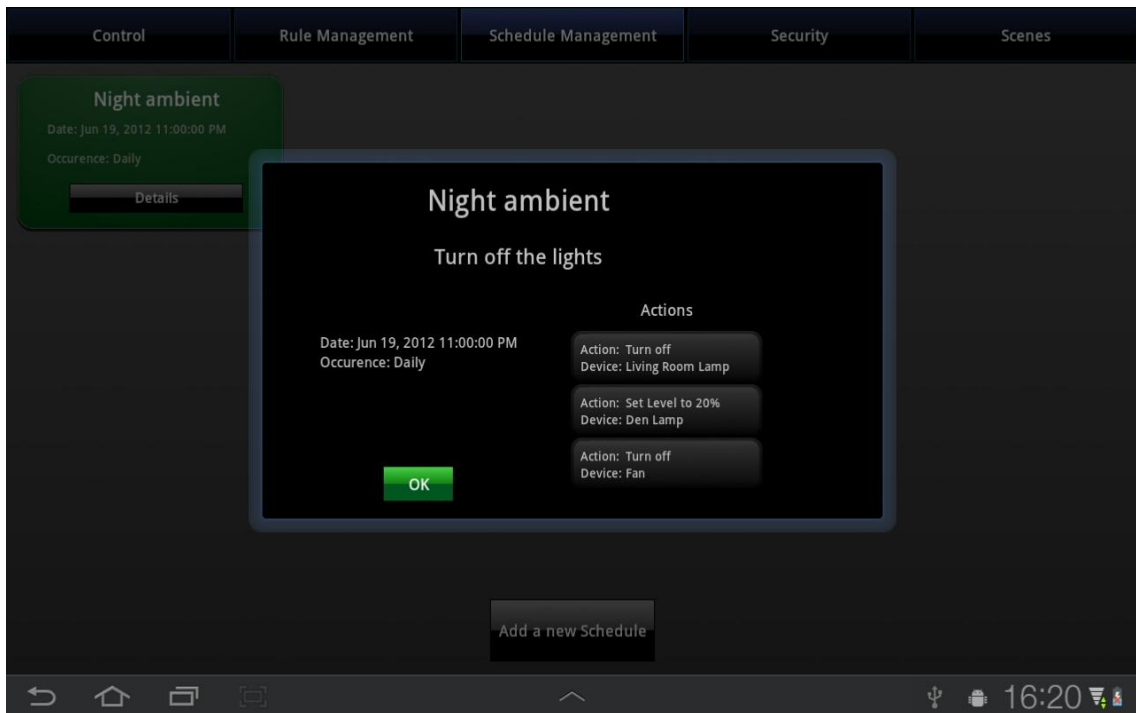


Figure 8: Schedule details

The schedule list and details are being updated every 10 seconds while in the Schedule Management window.

2.3 Add a Schedule

In the Schedule Management window you will see a button in the bottom side of the window. If you click this button, it will lead you to a new window where you can add a new schedule (see Figure 9).

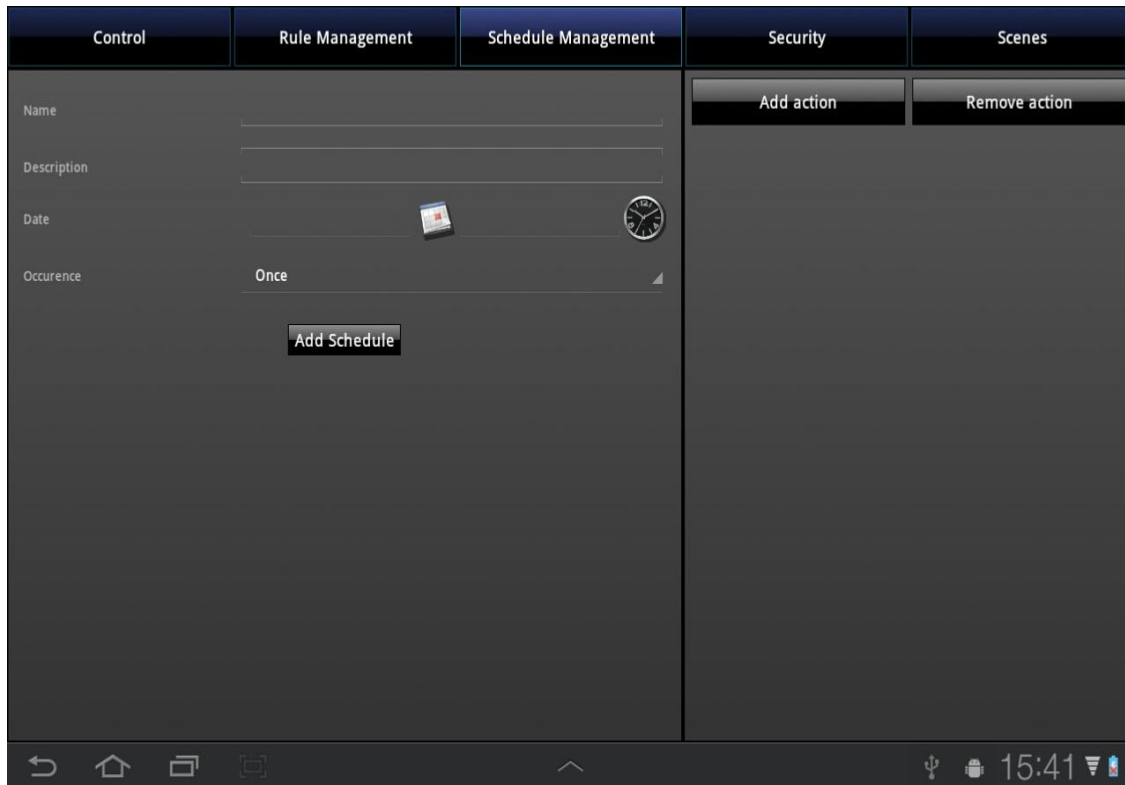


Figure 9: Add a Schedule

You will have to enter the new schedule name, description, date, time occurrence and a list of actions to be executed. You can add actions to the schedule by pressing the “Add action” button. A new container will appear, asking you to specify the action and the device. Once you select an action, the available devices that can execute that action will appear in the Device drop menu. For example, if you select the action “Set level to 40%”, only the lamp modules will be available in the Device drop menu (see Figure 10).

You can remove an action by long-pressing the container and clicking the red cross button, or you can remove the last added condition by clicking the “Remove Action” button (see Figure 11).

Once you finish adding all the details, you can add the schedule to the system by pressing the “Add Schedule” button. This will automatically take you back to the Schedule Management window.

Note that you have to at least specify a name, a date, a time, an occurrence and at least one action in order to the schedule to be valid.

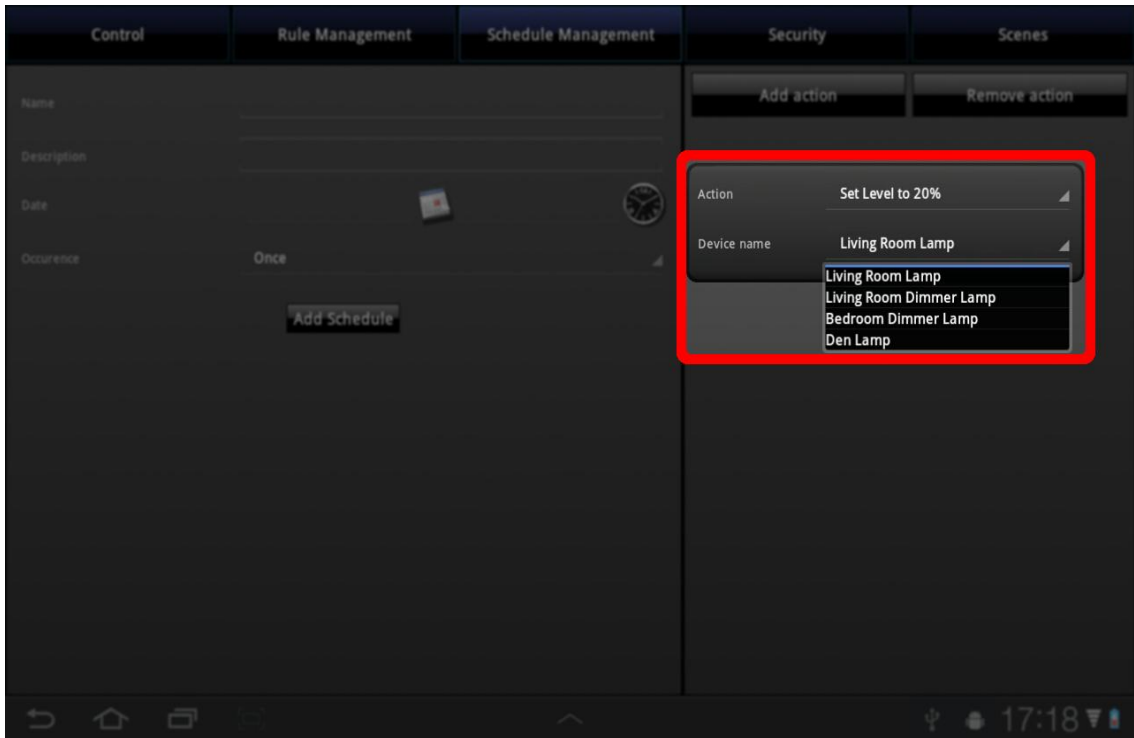


Figure 10: Device specific actions

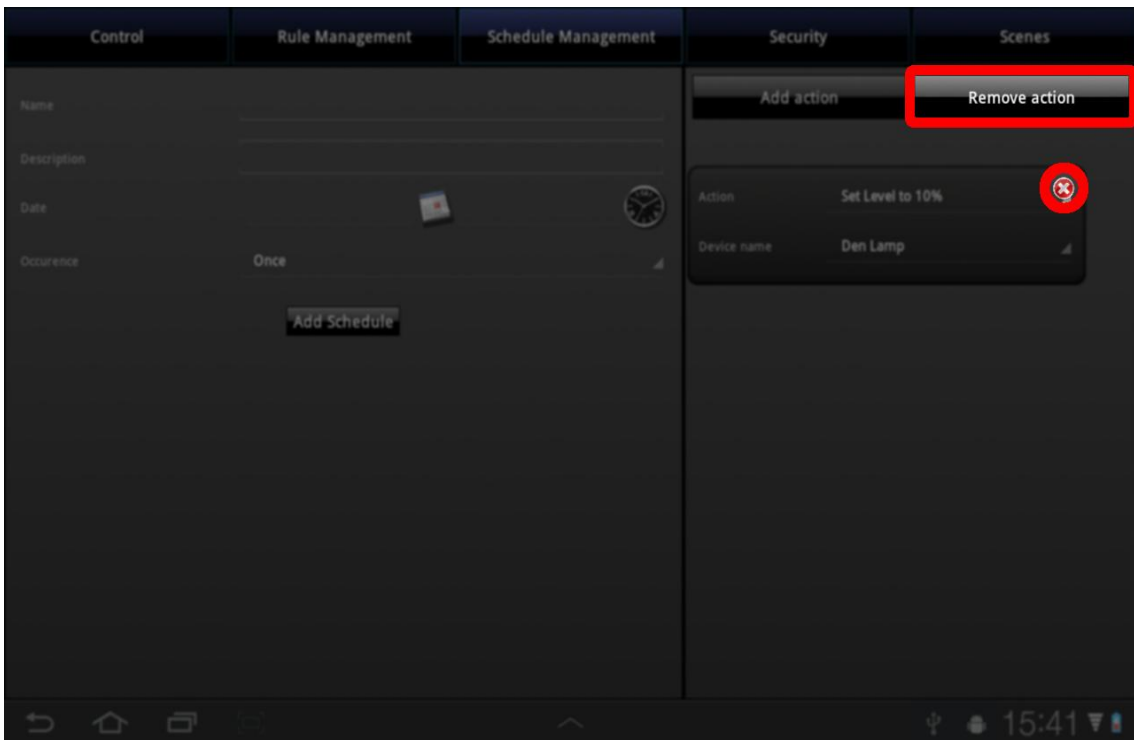


Figure 11: Remove an action

2.4 Edit a Schedule

Back in the schedule management window, if you long click a schedule container, the Details button will change to Edit Details. Once you click this button, it will lead you to the Edit Schedule window (see Figure 12).

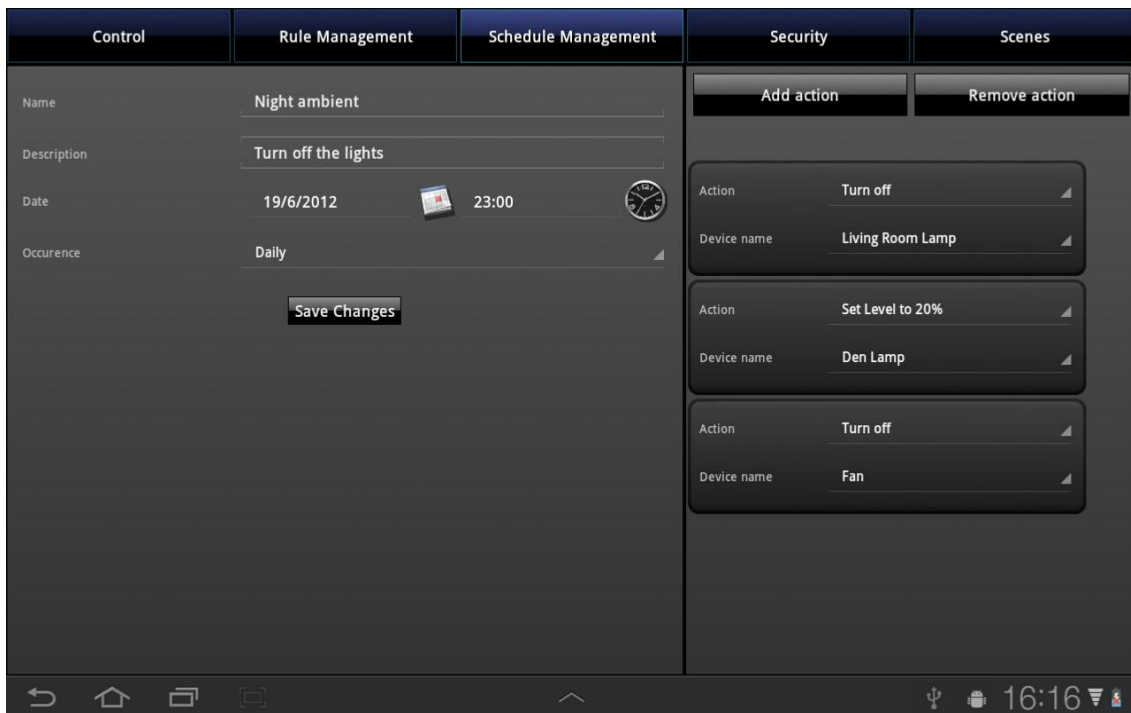


Figure 12: Edit a Schedule

This window is very similar to the Add Schedule window. In this case, all the fields have been already filled with the selected schedule details, so you easily change the details. Refer to the Add Schedule chapter to see the behavior of this window.

Once you finish editing the details, you can submit the changes to the system by pressing the “Save changes” button. This will automatically take you back to the Schedule Management window.

2.5 Remove a Schedule

In the schedule management window, if you long click a schedule container, a red cross button will appear in the right top corner of the container. You can remove the selected schedule if you press this button (a confirmation dialog will show) (see Figure 13).

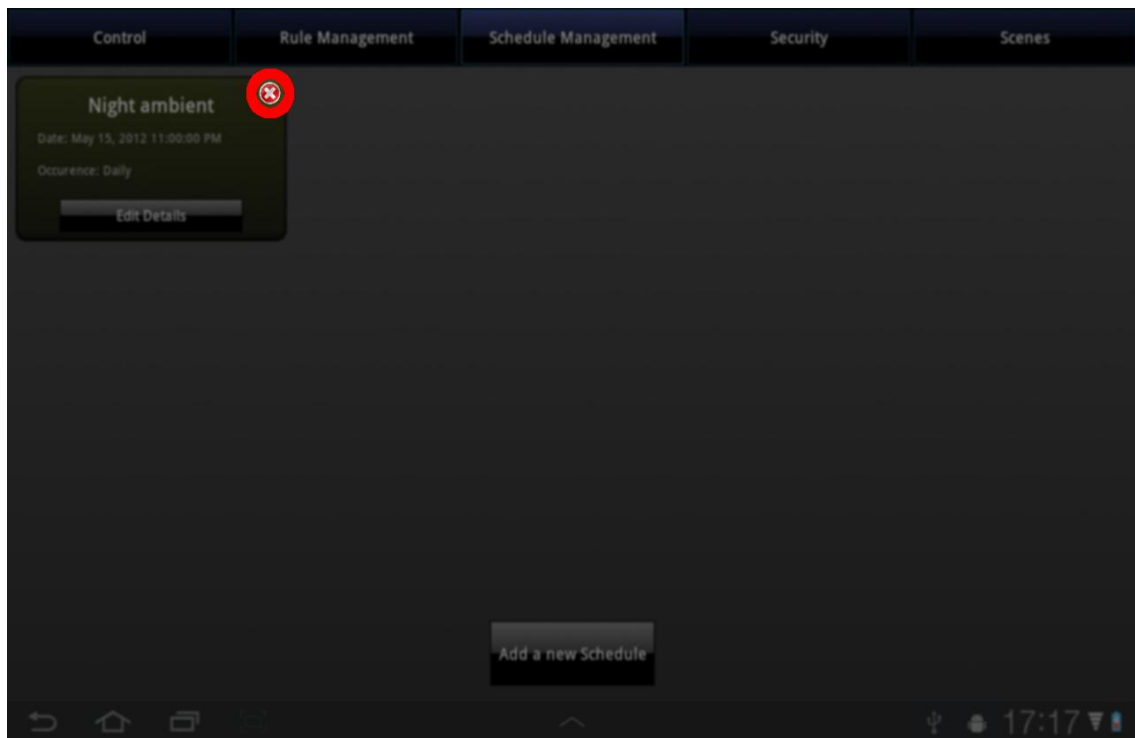


Figure 13: Remove a Schedule

3 RULE MANAGEMENT

3.1 What is a rule?

A rule consists of a series of conditions, such as specific device status, or sensor readings. If all the conditions are met, a series of actions are executed. To adjust these rules in a more advanced way, the system also allows the rule to be executed only during a period of time, and the ability to refresh this period every time it ends. The rule can also be activated or deactivated, which will prevent the rule from executing, even when the conditions are met.

3.2 View the available rules

Once in the Rule Management window, the application will show a list of the available rules. For each rule it will show a container with the rule name, whether the rule is scheduled or not, and the scheduled period of time and occurrence if it is scheduled. The container background color will be green if the rule is activated, or black if it's not (see Figure 14).

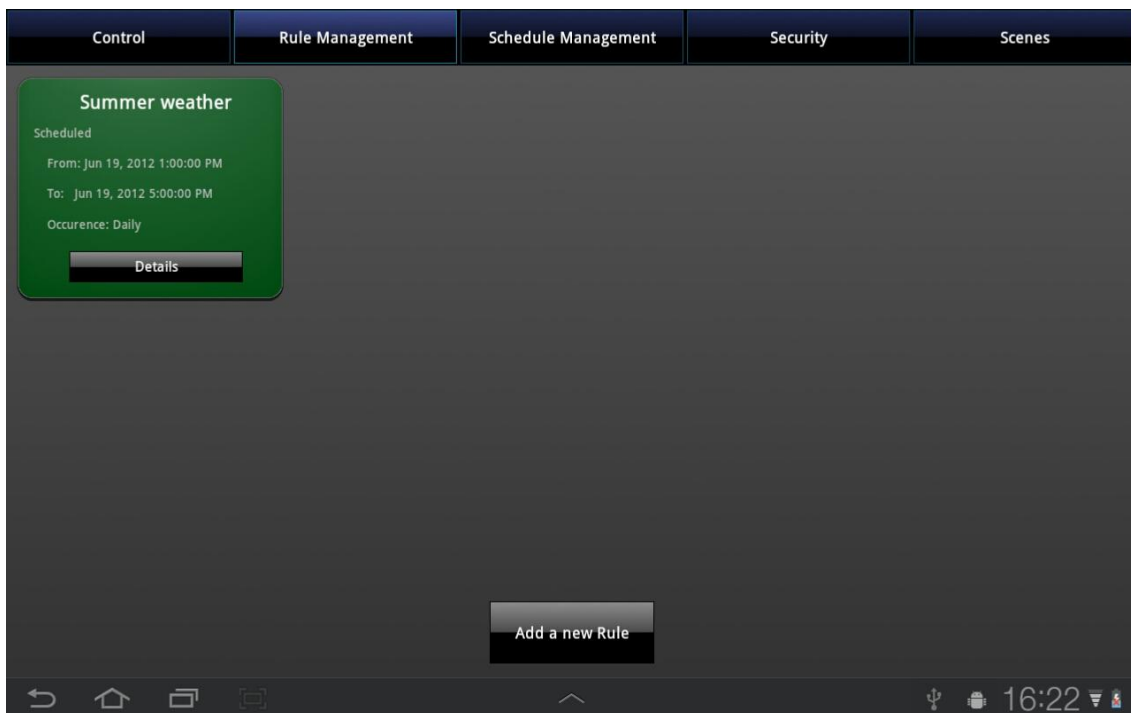


Figure 14: Rule list

It will also contain a button that will show you all the details of the rule, including the name, description, the scheduled period of time and occurrence if the rule is scheduled, the list of conditions, and the list of actions of the rule (see Figure 15).

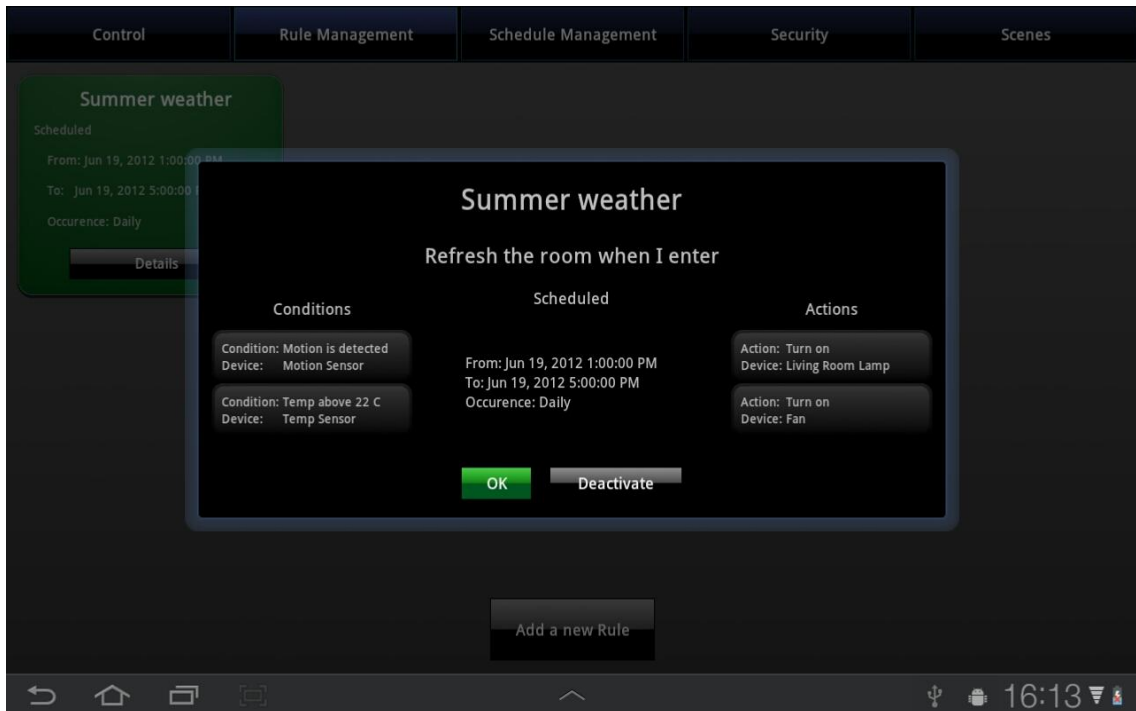


Figure 15: Rule details window

The rule list and details are being updated every 10 seconds while in the rule management window.

3.3 Add a Rule

In the rule management window you can add a new rule by pressing the “Add a new Rule” button in the bottom side of the window.

This button will lead you to a new window in where you can insert all the rule details (see Figure 16).

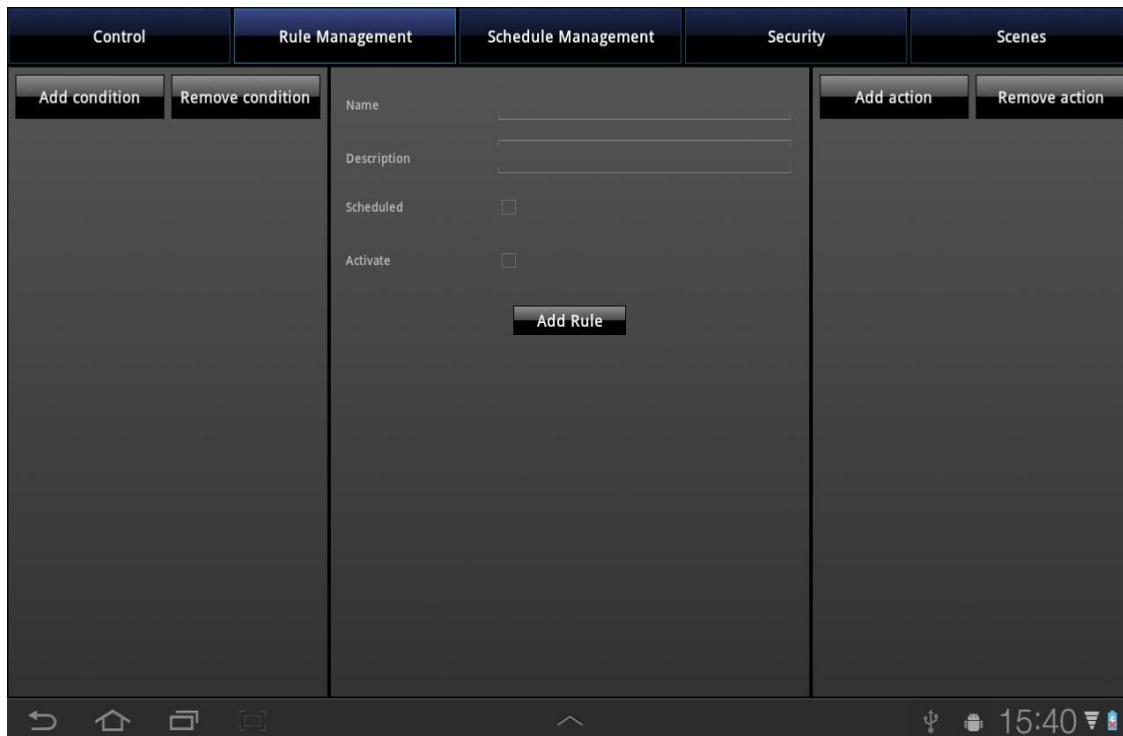


Figure 16: Add a Rule

You can select whether the rule is scheduled or not by activating or deactivating the “Scheduled” checkbox. If you activate the checkbox, new controls will appear to select the period of time in which you want the rule to be executed. It will also appear a drop menu to let you select the occurrence of this period of time.

You can also select whether the rule is going to be activated or deactivated by the “Activate” checkbox.

To add a condition to the rule you have to click the “Add Condition” button in the left side. A container will appear where you have to specify the condition. Once you select the device, a list of possible conditions of that device will show in the Condition drop menu. For example, if you select a lamp module, it will show conditions for its status or brightness level, for example “Level below 20%” (see Figure 17).

You can remove a condition by long-pressing the container and click the red cross button, or you can remove the last added condition by clicking the “Remove Condition” button (see Figure 18).

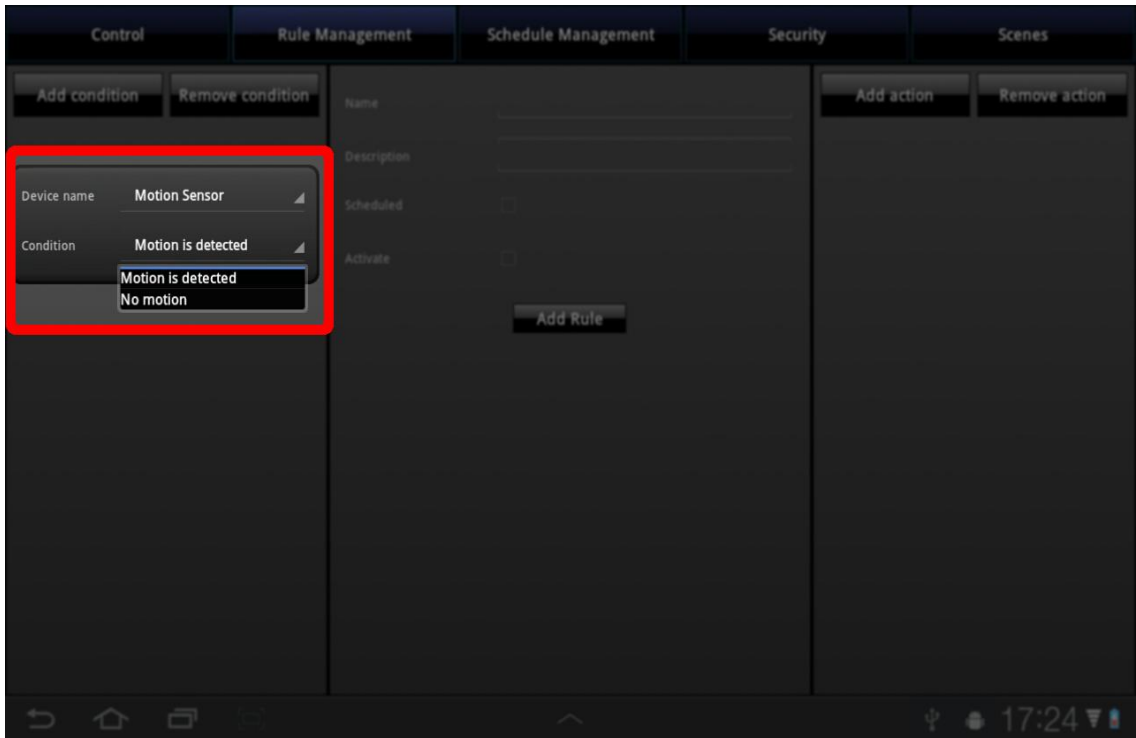


Figure 17: Device specific conditions

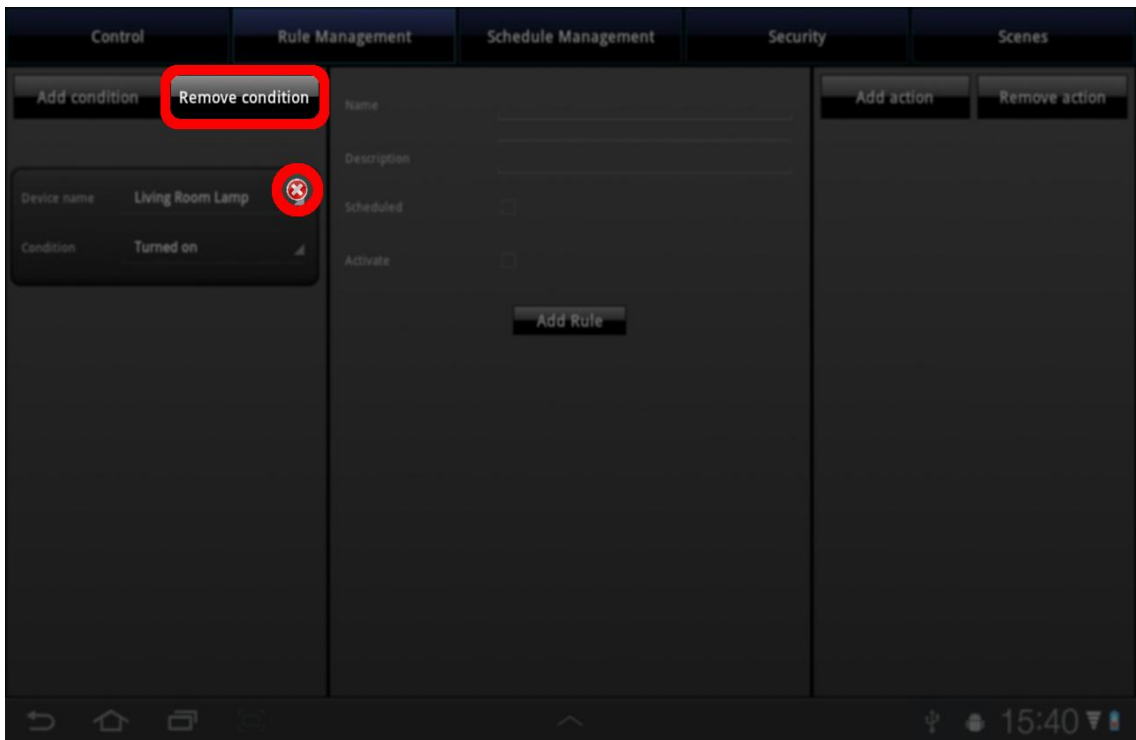


Figure 18: Remove a condition

You can add actions to the rule by pressing the “Add action” button. A new container will appear, asking you to specify the action and the device. Once you select an action, the available devices that can execute that action will appear in the Device drop menu. For example, if you select the action “Set level to 40%”, only the lamp modules will be available in the Device drop menu (see Figure 19).

You can remove an action by long-pressing the container and clicking the red cross button, or you can remove the last added action by clicking the “Remove Action” button (see Figure 20).

Once you finish adding all the details, you can add the rule to the system by pressing the “Add Rule” button on the bottom side of the window. This will automatically take you back to the Rule Management window.

Note that you must at least specify a name, a condition and an action in order for the rule to be valid.

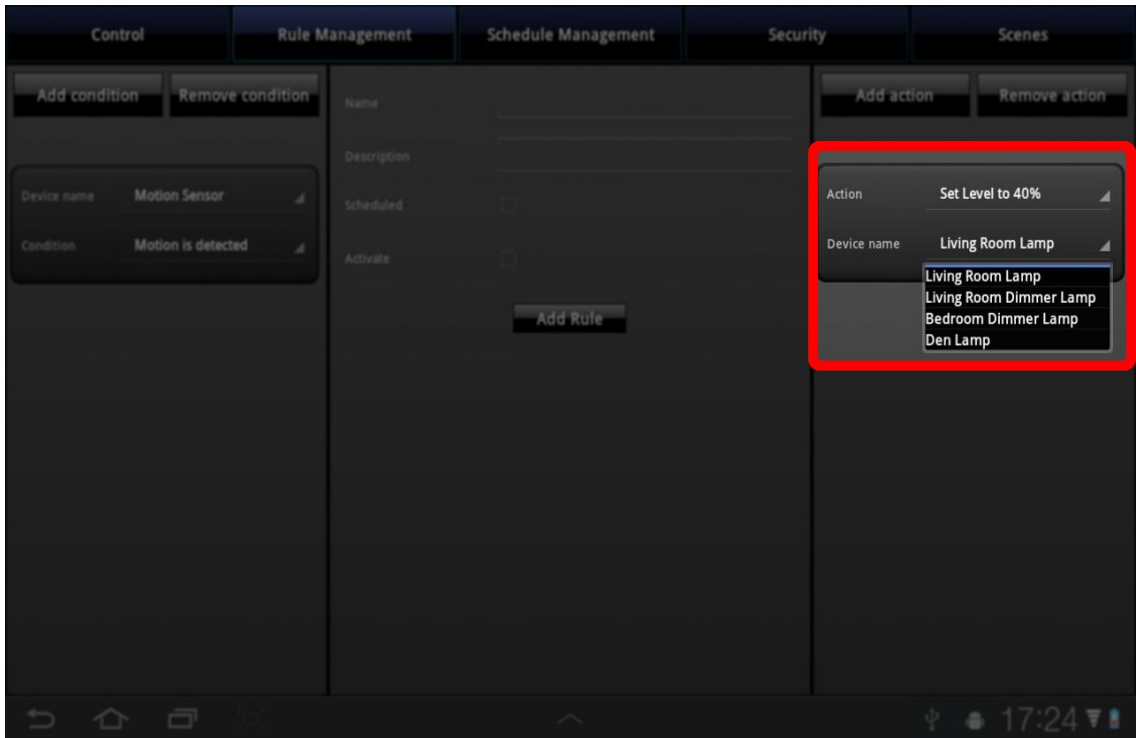


Figure 19: Device specific actions

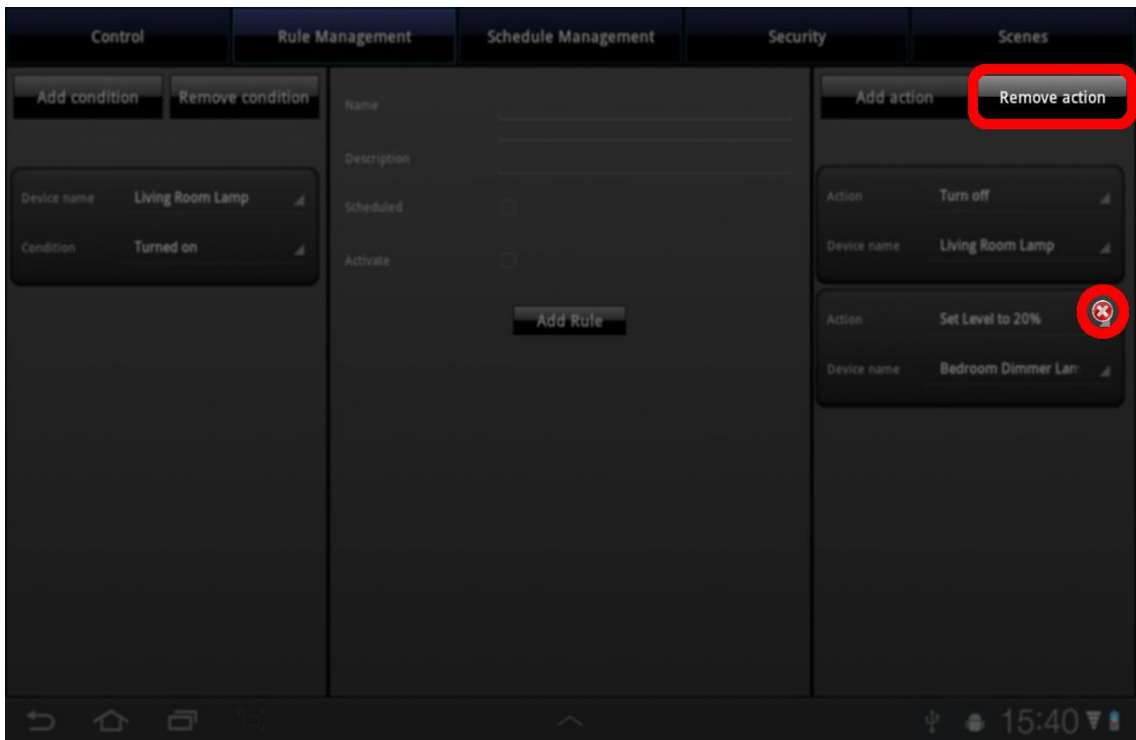


Figure 20: Remove an action

3.4 Edit a rule

In the rule management window, if you long-press a rule container, you will enter the edit-mode for that rule. The “Details” button will change to “Edit Details”.

This button will lead you to a new window, similar to the Add Rule window.

In this case, all the fields have been already filled with the details of the rule, so you can easily change whatever you want to change (see Figure 21). You can refer to the Add a Rule chapter to see the behavior of this window.

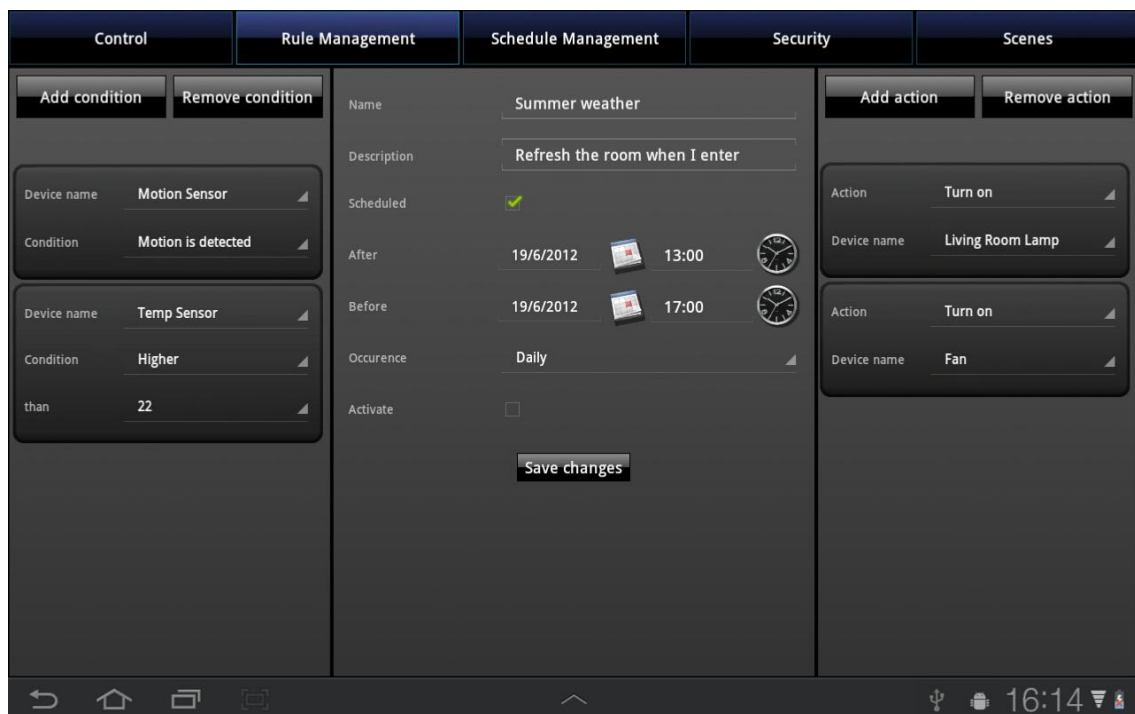


Figure 21: Edit a Rule

Once you finish editing the rule, you can submit the changes to the system by pressing the “Edit Rule” button in the bottom side of the window. This will automatically take you back to the Rule Management window.

3.5 Remove a Rule

In the Rule Management window, if you long-press the container of a rule, you will enter the edit mode for that rule. You can remove that rule from the system by pressing the red cross button in the top right corner of the container (see Figure 22).

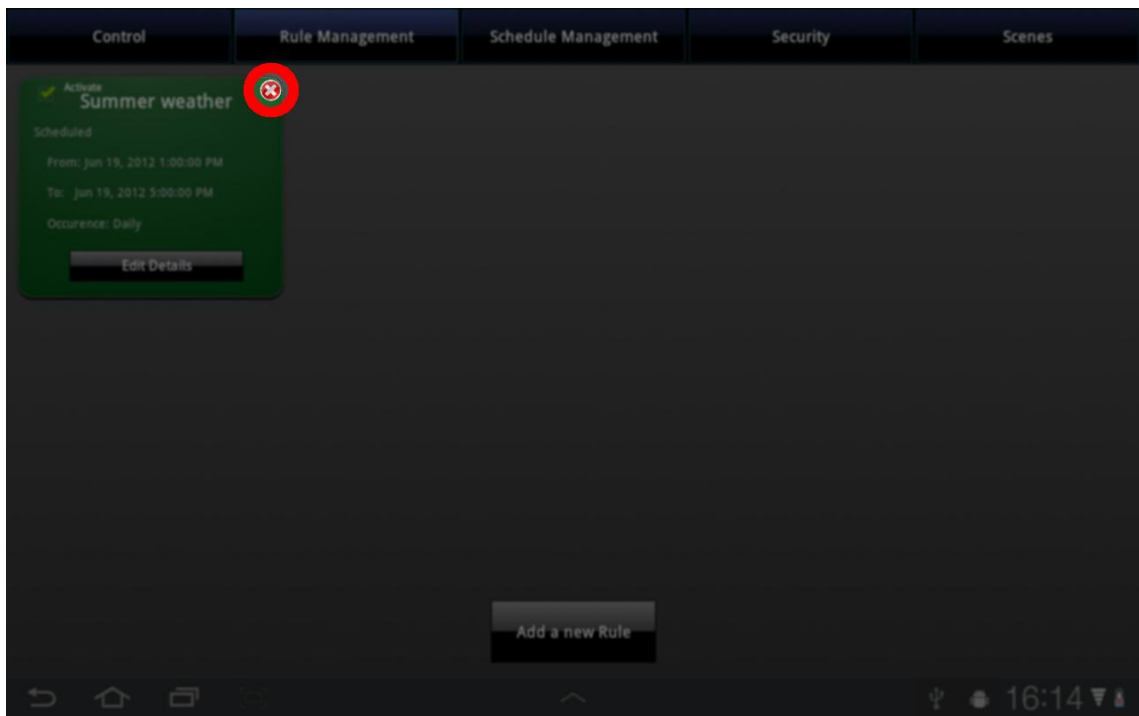


Figure 22: Remove a Rule

3.6 Activate/Deactivate a Rule

To activate or deactivate a rule, you have many options:

- In the Rule Management window, if you long-press the container of a rule, you enter the edit mode for that rule. You can activate or deactivate the rule by checking or unchecking the activation checkbox in the top left corner of the container (see Figure 23).
- In the Rule Management window, if you press the “Details” button of a rule, you will see a button that lets you activate or deactivate the rule (see Figure 24).
- In the Rule Management window, if you long-press the container of a rule and click the “Edit Details” button, you will enter the Edit Rule window. From here you can check or uncheck the “Activate” checkbox. Once you click the “Save changes” button, the changes will be saved (see Figure 25).

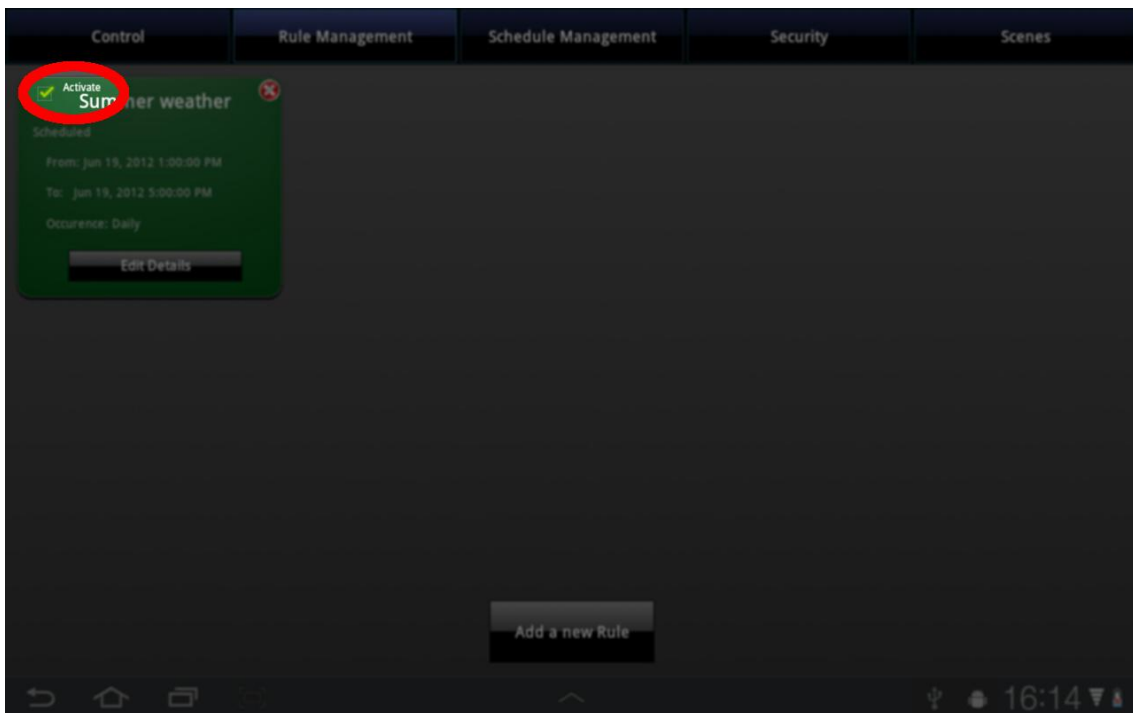


Figure 23: Activate a Rule (1)

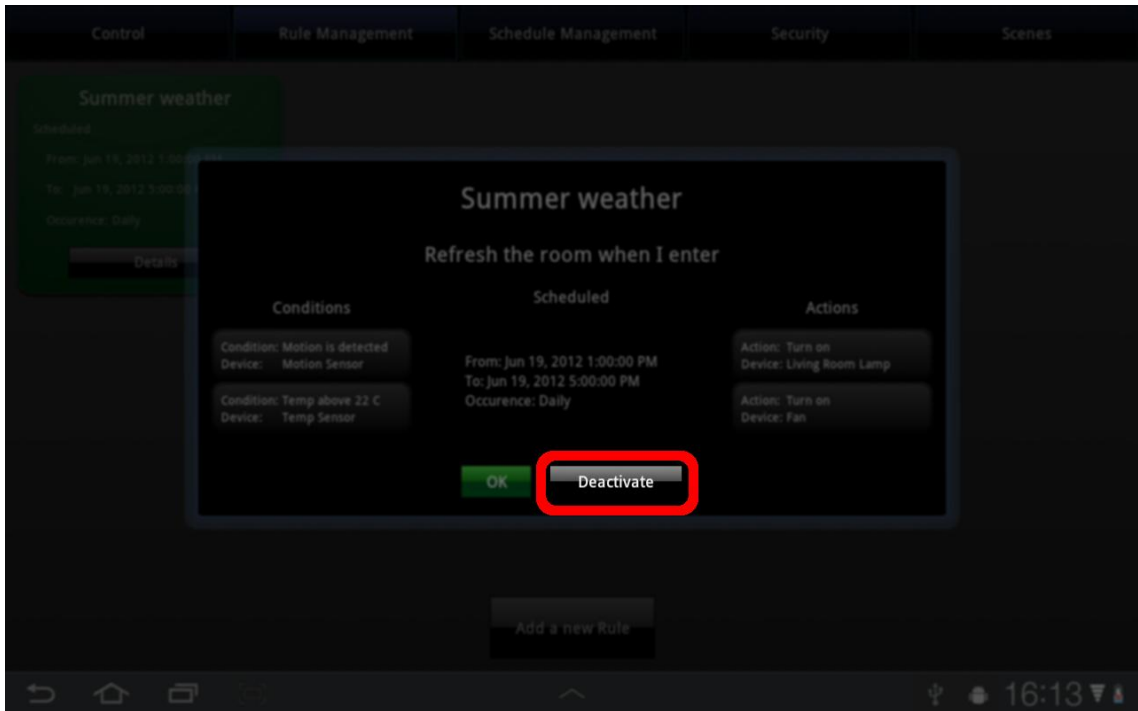


Figure 24: Activate a Rule (2)

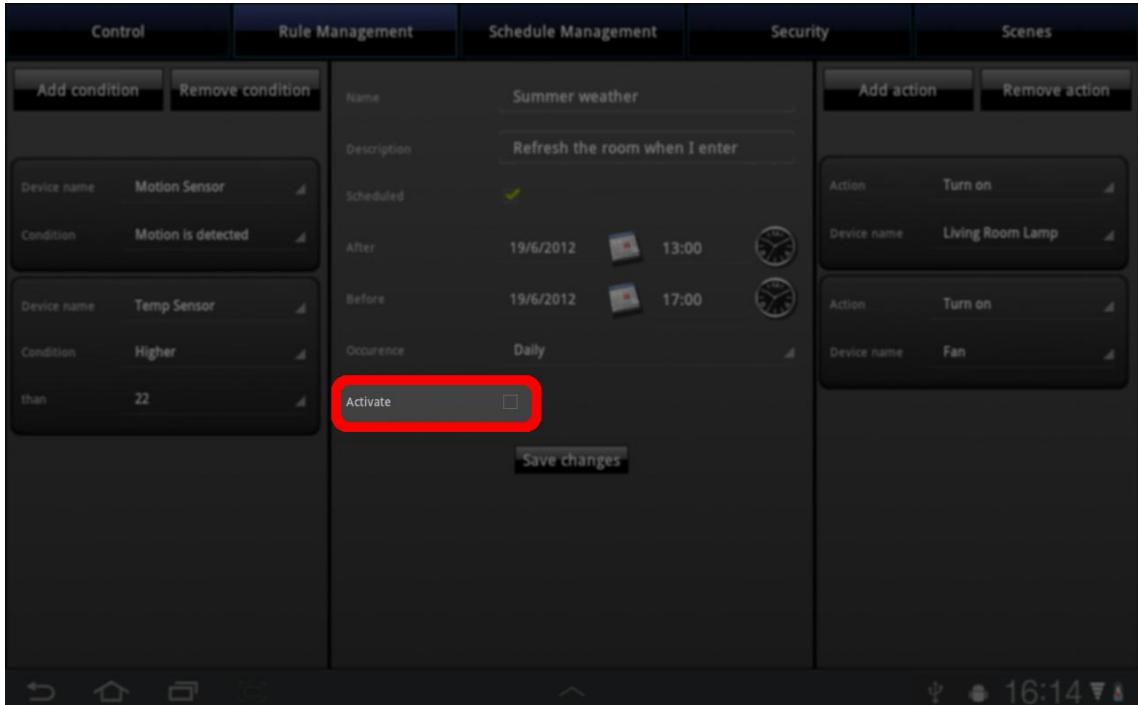


Figure 25: Activate a Rule (3)

4 SECURITY CAMERAS

4.1 Watch the cameras

In the Security window, you will see a menu on the left listing all the installed cameras. In the right side of the screen you will see four frames in which the installed cameras will show their images. Because of the heavy data consuming of these operations, there will be a loading time of a few seconds before the cameras show their images in the application (see Figure 26).

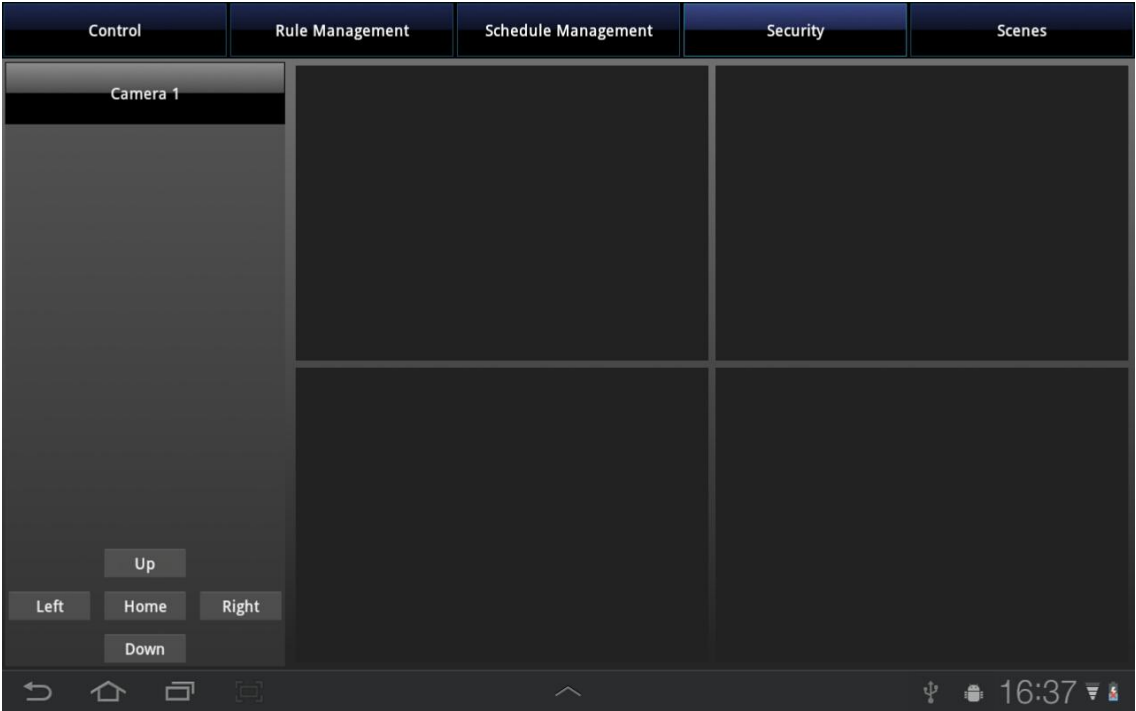


Figure 26: Security Cameras

By default, the first four cameras will be shown in the right panel. You can pick a camera from the left menu and drop it in any of the right frames in order to visualize the camera in that frame.

4.2 Control a camera

In the Security window, you can control the pan and tilt of a camera by selecting it from the left menu and operating the button panel in the bottom left side of the window (see Figure 27).

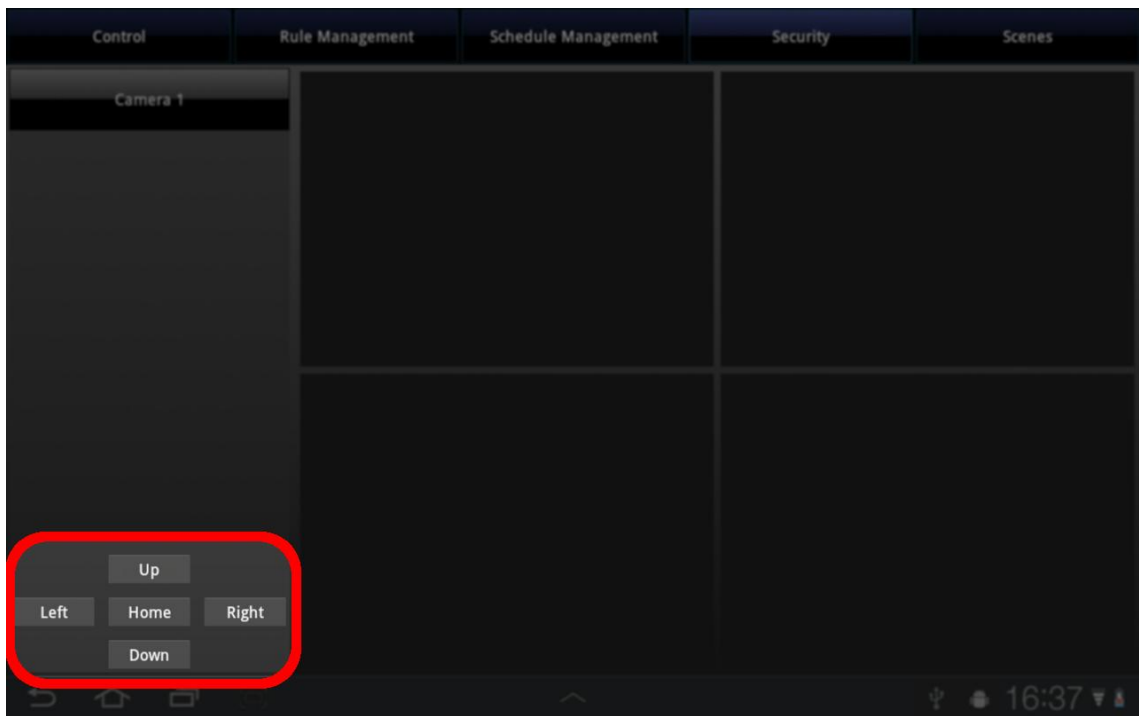


Figure 27: Control a Camera

5 SCENES

5.1 What is a Scene?

A scene consists of a saved status of a set of devices so it can be easily loaded instead of setting these statuses manually. In order to do this, you have to provide the list of devices so the system saves their current status and create a scene.

5.2 View the available scenes

In the Scenes window, you will see a list of all the actually available scenes. For each scene in the system you will see a container with the scene name and the description, as well as a “Details” button. If you click on this button, a popup window will show you all the details of the scene, including the name, the description, and the list of saved devices and statuses. It will also show a “Load” button to instantly load that scene.

If the system detect a scene is currently loaded, its container’s background will be green (see Figure 28).

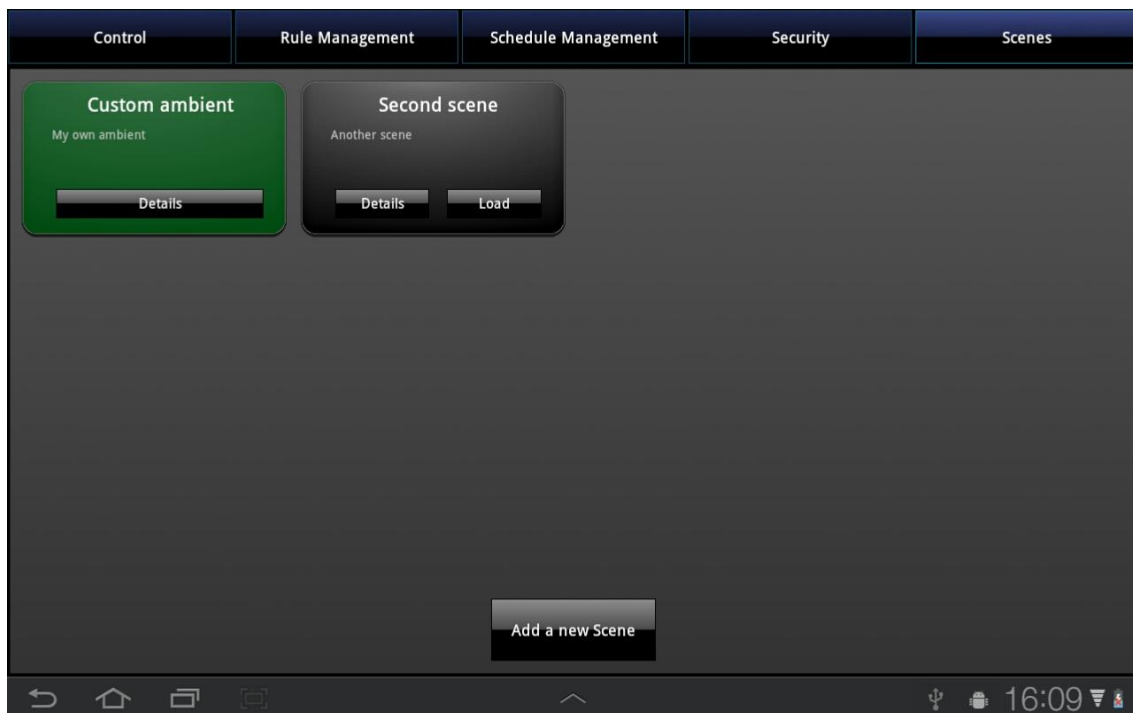


Figure 28: Scene list

5.3 Add a Scene

In the Scenes window, you will see an “Add a new Scene” button in the bottom side of the window. If you click this button, a floating window will appear. In this new window you can select the scene name, the description, and the list of devices you want to save. The device list can be grouped by their category or their location, by changing the drop down menu “Sort devices by”. Note that the system will save the current status of those devices, so before you add a new scene, you have to manually set that scene in the Control window. Once you finish selecting the details, you can add the scene to the system by pressing the “Add a new Scene” button again, or cancel the action by pressing the “Cancel” button. Either ways, the floating window will disappear (see Figure 29).

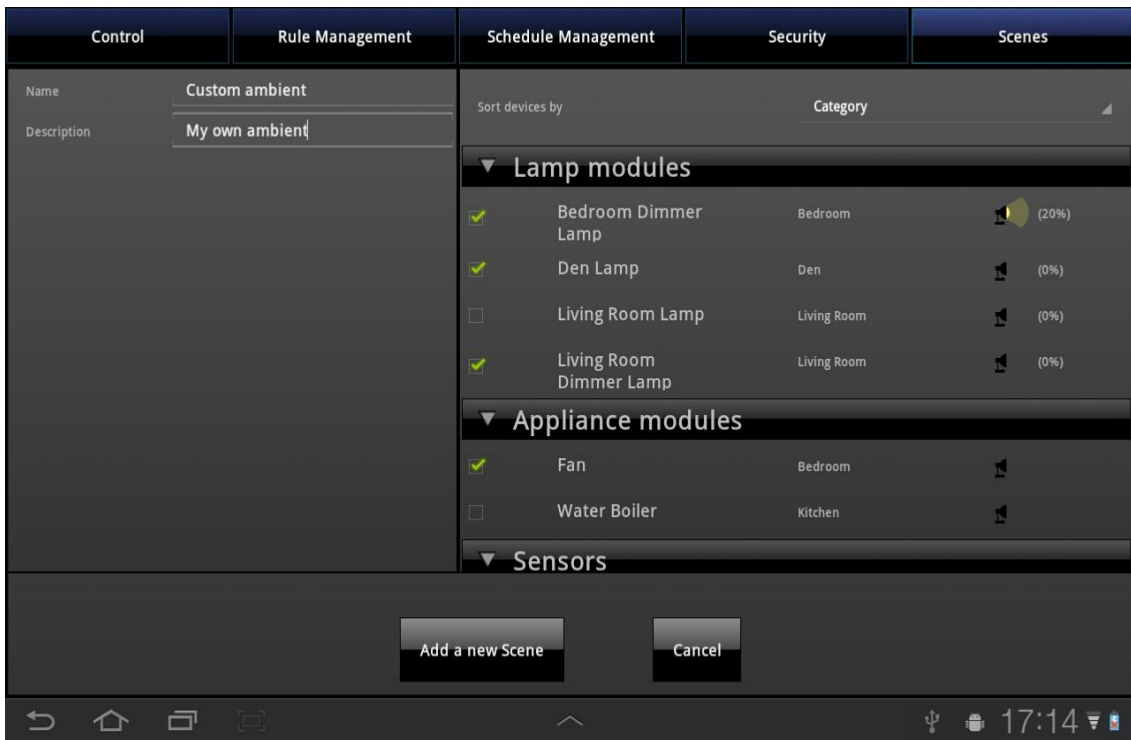


Figure 29: Add a Scene

5.4 Remove a Scene

In the Scenes window, if you long-click the container of a scene, a red cross button in the top right corner of the container will appear. If you click this button, the scene will be removed from the system (see Figure 30).

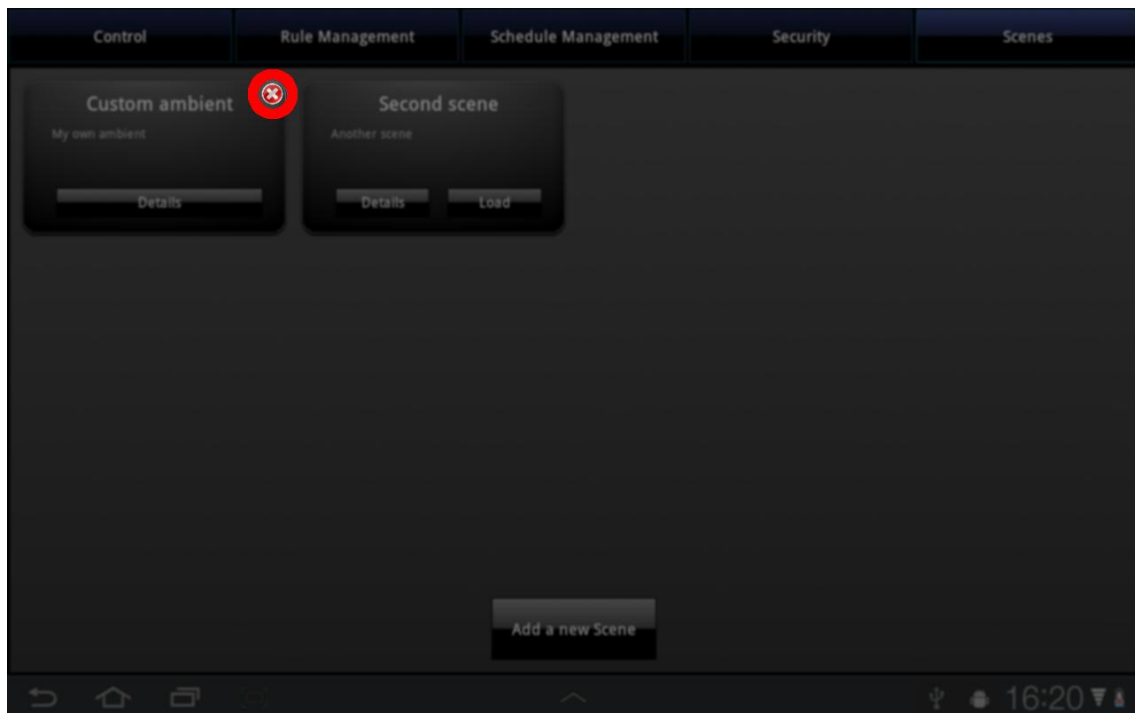


Figure 30: Remove a Scene

5.5 Load a Scene

In order to load a scene, you have many options:

- In the Scenes window, you can single-click a scene to load it. A confirmation box will ask you to confirm the action (see Figure 31).
- In the Scenes window, you can click the “Load” button of the container to instantly load that scene (see Figure 32).
- In the Scenes window, you can click on the “Details” button of a scene. A “Load” button will let you load that scene (see Figure 33).

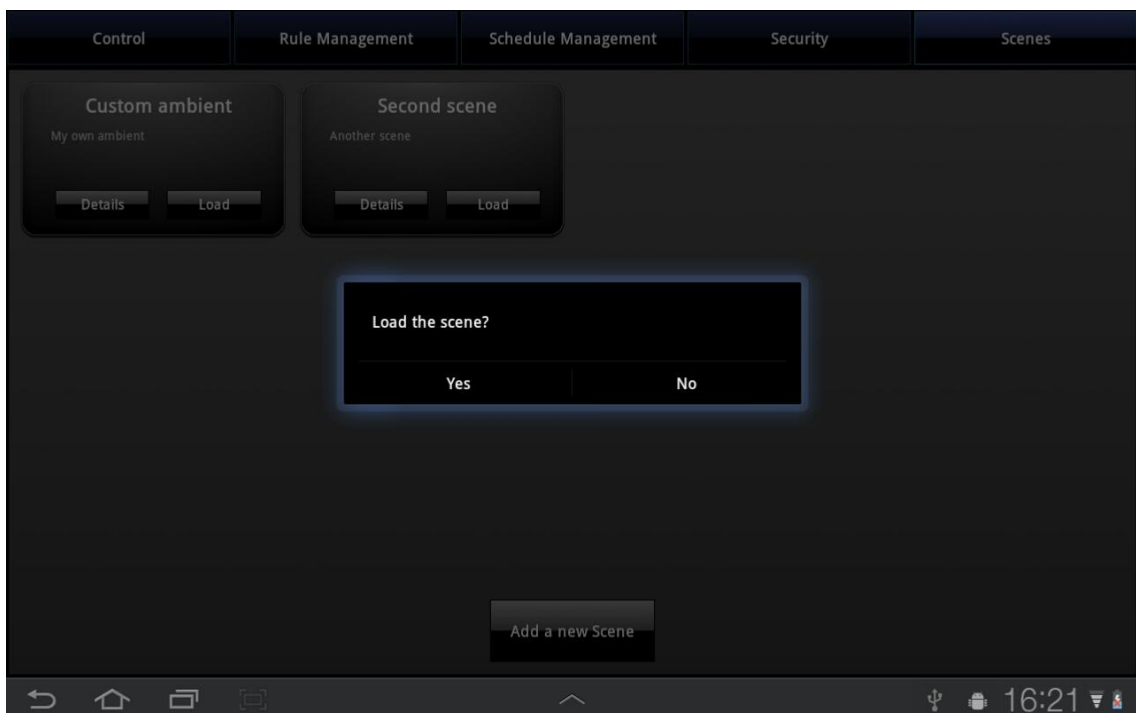


Figure 31: Load a Scene (1)

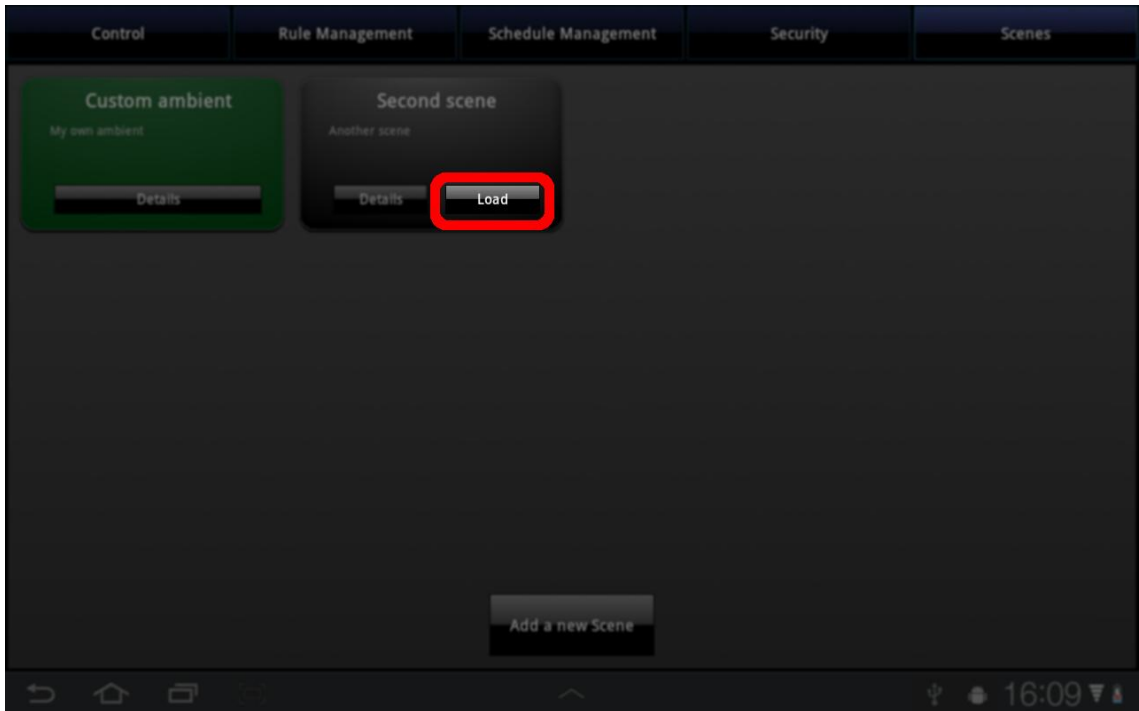


Figure 32: Load a Scene (2)

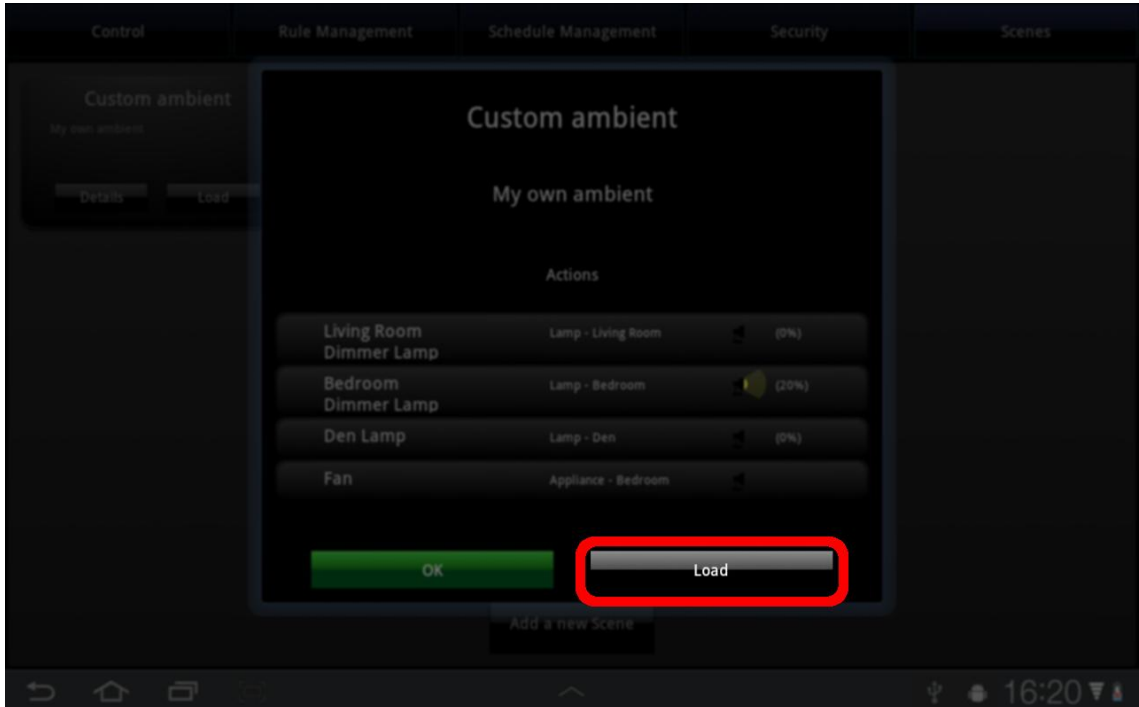


Figure 33: Load a Scene (3)

ANNEX III

HOME AUTOMATION PROTOCOLS

During the project we have included support for the following home automation protocols.

1 X10

X10 is an international industry that produces electronic devices for domestic uses, such as lighting control and sensor data reading. It was the first company to introduce these devices in the open market, making it affordable for home use. Though there have been many more companies that have developed these kinds of devices after them, X10 remains as the cheapest and simplest solution for domestic purposes, and are the most used technology nowadays.

Each device has a unique address, usually composed of one letter indicating the room of the device, and a number indicating the unit (see Figure 1). To specify an address, the devices have two switches that can be changed with a screwdriver (one for the room letter, one for the unit number). Some specific devices, such as motion sensors, get their address from an X10 console, once the device is linked to the console. Normally, this console is sold separately and cannot be controlled from a PC.



Figure 1: X10 appliance module

X10 uses the powerline of the house to send and receive commands from/to the modules. Some modules can also send and receive radio-frequency commands (for example the wireless motion sensor). To control these modules the user can either get a centralized console sold by X10 to link the devices and configure their behavior, or get a Main Component with a PC connection and use this PC to send the desired commands (see Figure 2).



Figure 2: X10 USB controller

To control the devices from a PC, X10 has released a simple SDK that offers two basic functions to communicate with the devices: `send(plc/rf)` to send commands via powerline (plc) or radio-frequency (rf), and `recvplc` to ask via the powerline (at the time of this document, there is no way to receive radio-frequency responses). Also, the SDK offers an event listener that receives events whenever a device receives a command.

There is one major limitation with this SDK, and it is that it doesn't support sensor information events, such as motion detection or door sensor events. In order to detect these kinds of events, the user has to buy a special console from X10, or develop a radio-frequency receiver to detect the events.

The most common devices sold by X10 include appliance modules, lamp modules, main components, centralized consoles, door/window sensors, and motion sensors.

2 INSTEON

Insteon is another company that designs domotic devices. It is a more actual technology and offers more advanced modules than X10. It is also more expensive. The main functionality is the same as X10, both of them can send and receive powerline and radio-frequency commands via either a centralized console or a PC interface controller (see Figure 3).



Figure 3: Insteon PLM

There are two main ways to communicate the PC with the Insteon devices: an Insteon PowerLinc Modem (PLM), or an Insteon PowerLinc Controller (PLC).

The PLC is built to store and execute local programs in a specific language (SALad) and the PLM always needs a PC to control the devices. To control the PLC, Insteon offers a management software called SmartLabs Device Manager (SDM) that lets you download and store SALad applications in the PLC.

The way to communicate the PLM with the Insteon devices is via RS232 packets. Once the developer's kit from Insteon is purchased, a guide with all the commands that can be sent to the devices is accessible.

3 ZWAVE

This is another commercial technology that offers home automation devices. The main difference from the other two technologies is that there is a big, complex open-source library to control the devices from a PC, called Open-ZWave.

As this is an open-source library, there is the possibility to adapt its source code to the specific needs of our project, such as controlling the polling system to ask the devices status.

Another difference from the other technologies is that the devices are automatically detected and stored whenever they are plugged in the system, so there is no need to manually add the device in our system. Of course, once it is added, the user might want to change some of the information of the device, such as their name or location, as the default ones might not be completely informative.

The main way to communicate the PC to the ZWave devices is by a USB adapter (see Figure 4).



Figure 4: ZWave USB adapter

Adding a new ZWave to the system is as easy as pressing a button in the USB adapter, and pressing the button in the device afterwards.