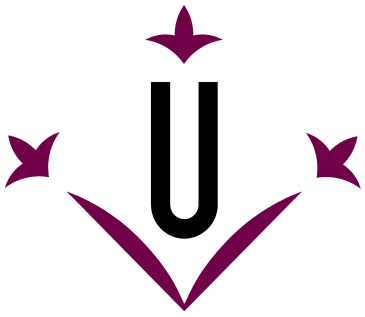
Escola Politècnica Superior

Master of Computer Engineering

ICT Project: Development and Implementation



Universitat de Lleida

Sprint 3:
Requirement definition and system analysis

Github: `https://github.com/GarduinoTeam/Garduino`

Students:
Adrià Casals Espax Joan Pau Castells Gasia
Gerard Donaire Alos Roger Truchero Visa
David Sarrat González

Date: December 18, 2019

Contents

List of Tables

List of Figures

Sprint 3

Introduction

This document highlights the main advances and aspects of work relevant to the third sprint of the development of the Garduino project.

This second iteration of the development of the project is mainly focused on the backend and frontend of the application, as well as the administration of users and the database.

As such, the main aspects of the progress and changes in the project, as well as the requirements of the resulting sprint, product backlog and sprint backlog will be presented.

As a result of this sprint, the general architecture of the application, a database model and a navigation scheme of the different screens and their internal relationship will be presented, both for the part of the web application and for the part of the Android application. Finally, it will offer a greater degree of deepening in financial factors and economic feasibility study.

1 Software Installation

- Android Studio 3.5.2
- JBOSS with WildFly 10.0
- Pgadmin 4
- PostgreSQL 12.0.1
- Arduino 1.8.9

Id	As a	I want to be able to	So that	Priority	Sprint
1	Administrator	Create, modify and delete devices	I can properly configure my irrigration system as a whole	HIGH	2
2	Administrator	Enable and disable devices	I can properly coordinate the different devices in my ittigation system at each time	MEDIUM	2
3	Administrator	Create, modify and delete rules	I can properly set up the behavior of each device	HIGH	2
4	Administrator	Enable or disable rules	I can perform punctual changes to the devices' behavior at a certain moment	MEDIUM	2
5	Administrator	Create, modify and delete conditions	I can customize and set up what each rule stands for in a high depth level	HIGH	2
6	Administrator	Enable or disable conditions	I can perform punctual changes to the rules' definition at a certain moment	MEDIUM	2
7	Administrator	Request the system to start and cancel manual irrigations for a device	I can order the system to start an immediate irrigation in a device regardless of the configuration and whether the expected conditions are being met	LOW	2
8	Administrator	see the current real-time status of my device	I can keep track of each part of my irrigation system and my garden as a whole	HIGH	2

Table 1: Table of user stories Sprint 3

2 Product backlog + Sprint backlog

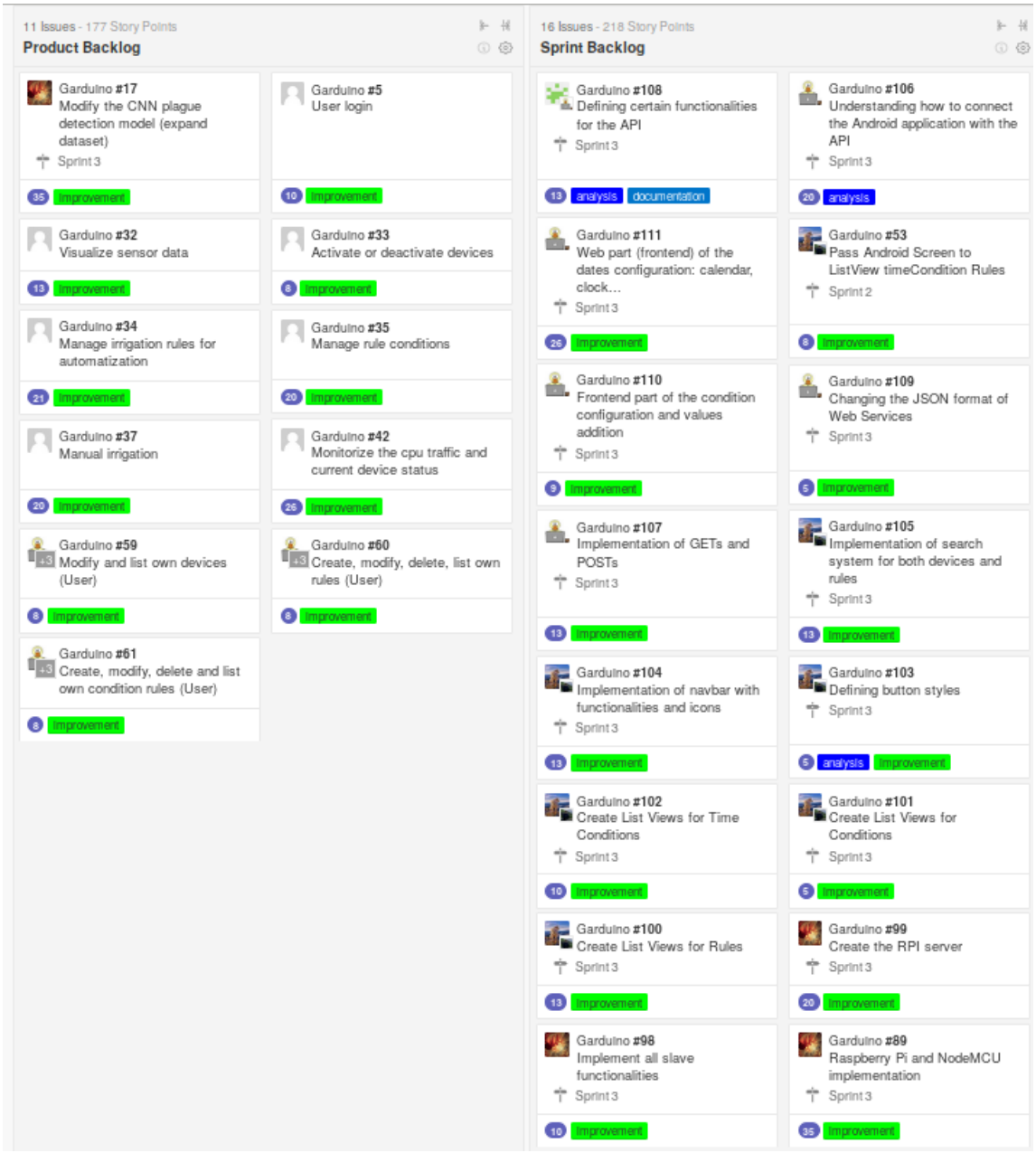


Figure 1: Product backlog and sprint backlog issues

Event	Trigger	Priority	Weight
Modify the CNN plague detection model (expand dataset)	Users	LOW	35
User Login	Users	LOW	10
Visualize sensor data	Users/Admin	MEDIUM	13
Enable or disable devices	Users	MEDIUM	8
Manage irrigation rules for automatization	Users	HIGH	21
Manage rule conditions	Users	HIGH	20
Visualize users info	Admin	LOW	10
Manual irrigation	Users	MEDIUM	20
Show plaged detection analysis results	Users	MEDIUM	10
Monitorize the cpu traffic and current device status	Admin	MEDIUM	26
Modify and list own devices	User	MEDIUM	8
Create, modify, delete, list own rules	User	MEDIUM	8
Create, modify, delete, list own condition rules	User	MEDIUM	8

Table 2: Product Backlog explicit table

Event	Trigger	Priority	Weight
Define certain functionalities for the API	User/Admin	HIGH	13
Understand how to connect the app with the API	User	HIGH	20
Web part of time conditions settings	Admin	MEDIUM	25
Pass Android screen to ListView timeCondition rules	User	LOW	8
Frontend part of the condition settings	Admin	LOW	9
Changing the JSON format of WS	Admin	MEDIUM	9
Implement GETs and POSTs	Admin	HIGH	13
Implementation of search systems	User	LOW	13
Navbar with functionalities and icons	User	LOW	13
Button style definition	User	LOW	5
List views for time conditions	Admin	MEDIUM	10
List views for regular conditions	Admin	MEDIUM	5
List views for rules	Admin	MEDIUM	13
RPI server	Admin	HIGH	20
Slave functionalities	Team	HIGH	10
Raspberry Pi and NodeMCU implementation	Team	HIGH	35

Table 3: Sprint 3 Backlog explicit table

3 Requirements

3.1 Non-functional requirements

1. **Product**
   (a) **Accessibility and Usability**
      i. Regardless of the style of interaction, the interface has to be simple and intuitive, providing a high level of interactivity and usability. The tasks will be as visual as possible, since the application will be oriented to all types of age ranges.
   (b) **Concurrency**
      i. The app has to be multi-user, allowing the usage of several users simultaneously of the application.
   (c) **Portability**
      i. **Adaptability (multi-device):** It is required that the design is "responsive" in order to ensure proper display on multiple devices, such as tablets, smartphones...
   (d) **Support**
      i. The mobile app will be developed by the Android platform.
2. **External**
   (a) **Privacy**
      i. All data management has to conform to the requirements of the organic law of data protection in order to preserve privacy in the processing of personal data.

3.2 Functional requirements

3.3 List of functionalities and features

List of functionalities and features that all services (Android app, web service and Irrigation System) will accomplish

3.3.1 Android app

- **The Android app will have to** allow communication with the web service.
- **The Android app will have to** allow each user to log in.
- **The Android app will have to** allow each user to configure/create an irrigation unit independently.
- **The Android app will have to** show all irrigation units configured by the user.
- **The Android app will have to** allow the manual watering option for each watering unit.
- **The Android app will have to** allow planned irrigation of each irrigation unit.
- **The Android app will have to** allow the option of monitoring the crop with data obtained from sensors and the webcam.
- **The Android app will have to** allow the change of language.

3.3.2 Web service

- **The web service will have to** allow communication with the Android application and the Arduino devices.
- **The web service will have to** allow the execution of a cron to make requests to the Arduino every X time so that a user can automatically receive notifications of his crop.
- **The web service will have to** allow the execution of a machine learning script in charge of pest detection and return the data back to the Android application.
- **The web service will have to** be able to determine when watering is optimal (if the automatic watering function has been used without programming) and send the request to the Arduino.
- **The web service will have to** allow CRUD operations directly from the DB.

3.3.3 Detection plague script

- **The pest detection algorithm will have to**, using an image of a plant as an input, determine if is contaminated or not.
- RPi(**R**aspberry **Pi**) 3B (master):
  - **The RPi will have to** allow communication with the web service.
  - **The RPi will have to** send/receive data from/to the slaves to perform operations.
- NodeMCU ESP8266 (slave):
  - **The NodeMCU will have to** be able to send the data to the RPi.
  - **The NodeMCU will have to** recolect the data from the humidity, temperature and soil sensors.
  - **The NodeMCU will have to** capture and image of the device plant.
  - **The NodeMCU will have to** control the relay to make irrigation when receives the request from the RPi.

3.4 Observations

- The leaf trimming algorithm requirement has been removed because the system will have a model that will already have the capabilities to process the whole image of the plant without the need of trimming it
- The arduino/rpi architecture has been replaced by a master/slave architecture using RPi like a master and NodeMCU slaves.

4 Main use cases

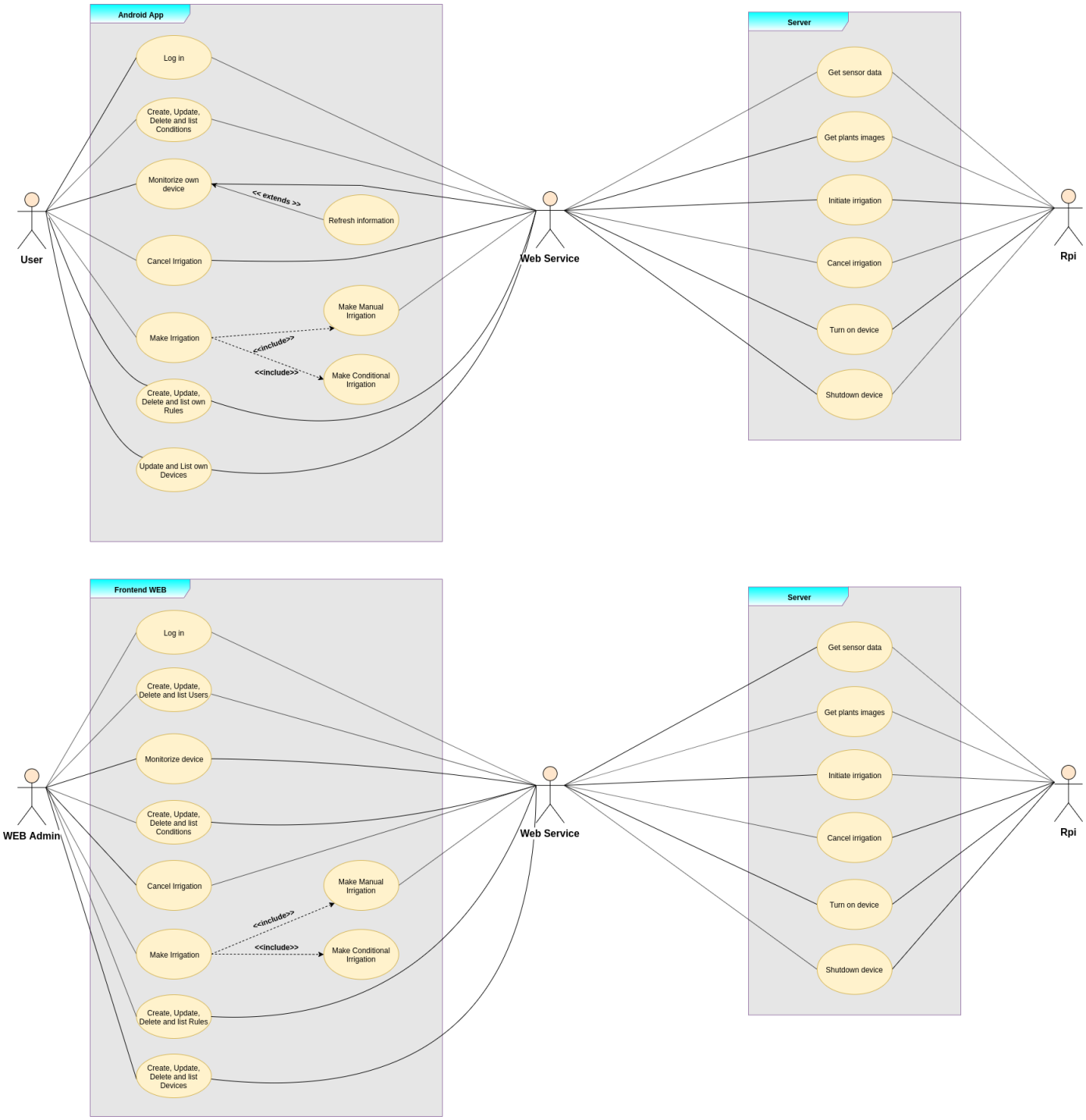


Figure 2: Application use cases diagram

Figure 2 shows the following actors: the user, the web service, the Rpi and the Admin.

The user will be able to log in, ask for their installations to be shown and edit them. He can also monitor his own installations. They can also create, modify, delete and list both their rules and irrigation conditions. In order to irrigate, the user will be able to irrigate their facilities manually and with irrigation conditions. All these functionalities are received by the Web Service which will wait for the requests that are made from the Android application and this one will ask for the necessary information to the Rpi to be able to give this information to the Android part and consequently to show it.

The admin will be able to logge, request that all the installations are shown as well as to create them, to modify them and to eliminate them. He can also monitor them. You can then create, modify, delete and list all rules, irrigation conditions and users. In order to water, the admin will be able to water all installations manually and with watering conditions. All these functionalities are received by the Web Service which will wait for the requests that are made from the frontend of the web part and it will request the necessary information to the Rpi to be able to give this information to the part of the frontend and consequently to show it.

5 General architecture

5.1 System architecture

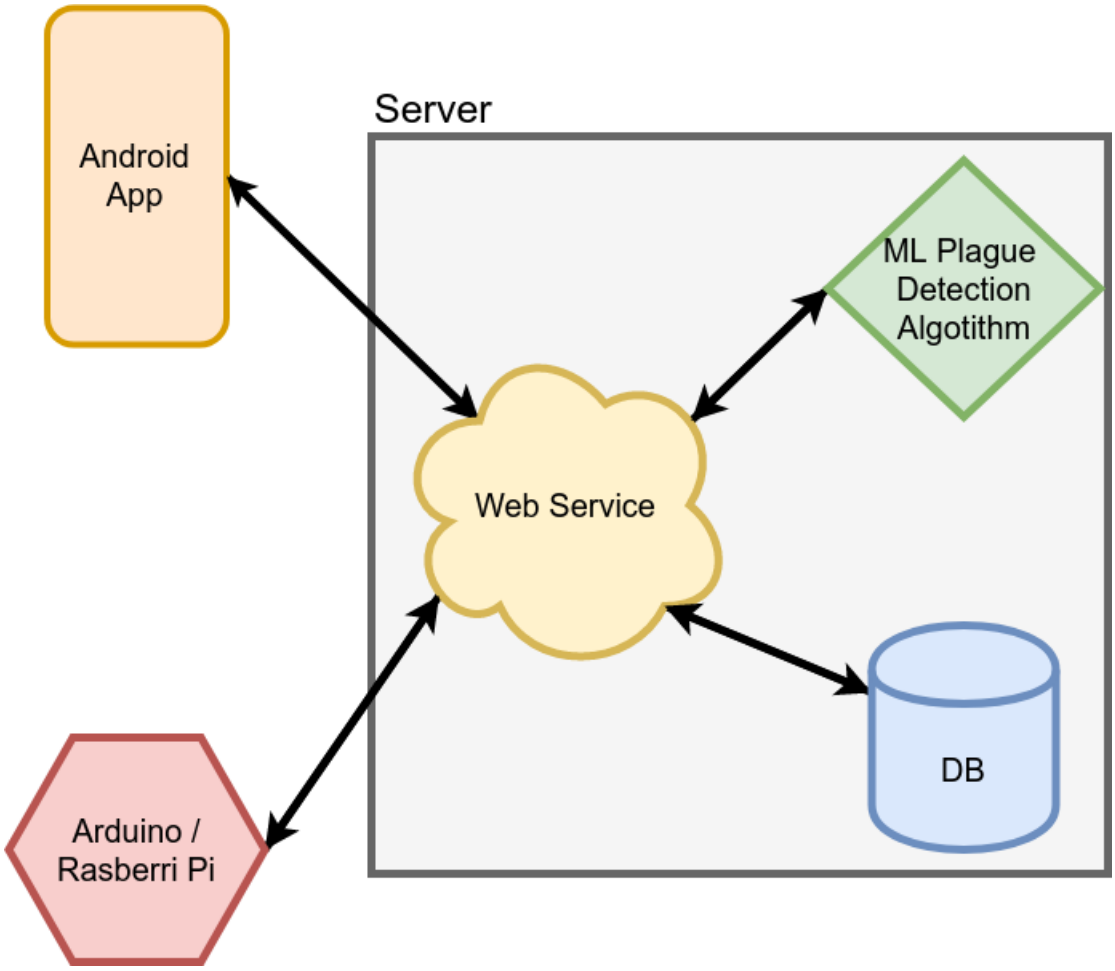


Figure 3: System app architecture

The architecture of our application is composed by:

- **Android App**: Graphical interface, in charge of collecting the data entered by the user. We are doing it using Android Studio.
- **Web Service**: Makes middleware of the application. It manages requests between the Android app and the RPi/NodeMCU, the database, the frontend ...
  - **Frontend:** Graphical interface used by the web application administrator to perform operations against the backend of the webservice and monitor the status of each user's devices, thus allowing absolute control of the data to be given to the administrator.
  - **Backend:** It is in charge of communicating with the android application, frontend and Rpi, receiving your requests, processing the data and preparing a response. It will act as the middleware of our system and therefore, will have more computational load.
- **RPi/NodeMCU**: Device that will be in charge of the irrigation of each environment (gardening, flowerpot, ...).
- **Database**: Manage all the data of the application, user credentials, register all requests, device info, ...

5.2 RPi/NodeMCU architecture

Figure 4: Arduino and rapsberry pi architecture

So, we have the following components

- **Raspberry pi 3B**: We will use this microprocessor to ask the slaves for information about the sensors and the webcam, and then return the call to the web service with all the data.
- **NodeMCU ESP8266**: It will act as a device slave, in charge of collecting sensor data, activating irrigation and capturing our plant. This microcontroller will have a wifi module to maintain a wireless communication with the RPi(**R**aspberry **Pi**).
- **Webcam ESP32**: This module will be in charge of capturing the plant and sending the data to your NodeMCU.
- **Relay**: Useful to control a motor, a led strip, or any other module using arduino. The use is sumple, just connect a digital output of our esp8266 to our relay module, and then we can control a power-demanding appliance with the digital signal provided by the esp8266.
- **Water pump**: He will be in charge of pumping the water in order to carry out the irrigation correctly.
- **Power supply**: Our devices will need a source of energy, in this case we have chosen a power supply of 12V to batteries. In the future it could be exchanged for a solar panel or even make a hybrid between the two depending on the climate.
- **Soil moisture sensor**: Measures the volumetric water content in soil. Since the direct gravimetric measurement of free soil moisture requires removing, drying, and weighing of a sample, soil moisture sensors measure the volumetric water content indirectly by using some other property of the soil, such as electrical resistance, dielectric constant, or interaction with neutrons, as a proxy for the moisture content.
- **DHT sensor**: This sensor will allow us to simultaneously measure temperature and humidity. We have two types of DHT sensors, DHT11 and DHT22.

  The DHT11 is a very limited sensor that we can use for training, testing, or in projects that don't really require accurate measurement.

  The DHT22 has acceptable features for use in real monitoring or logging projects that require medium accuracy.

  In this project we will start implementing the DHT11 version due to its low cost (70 cents).

6 Data model

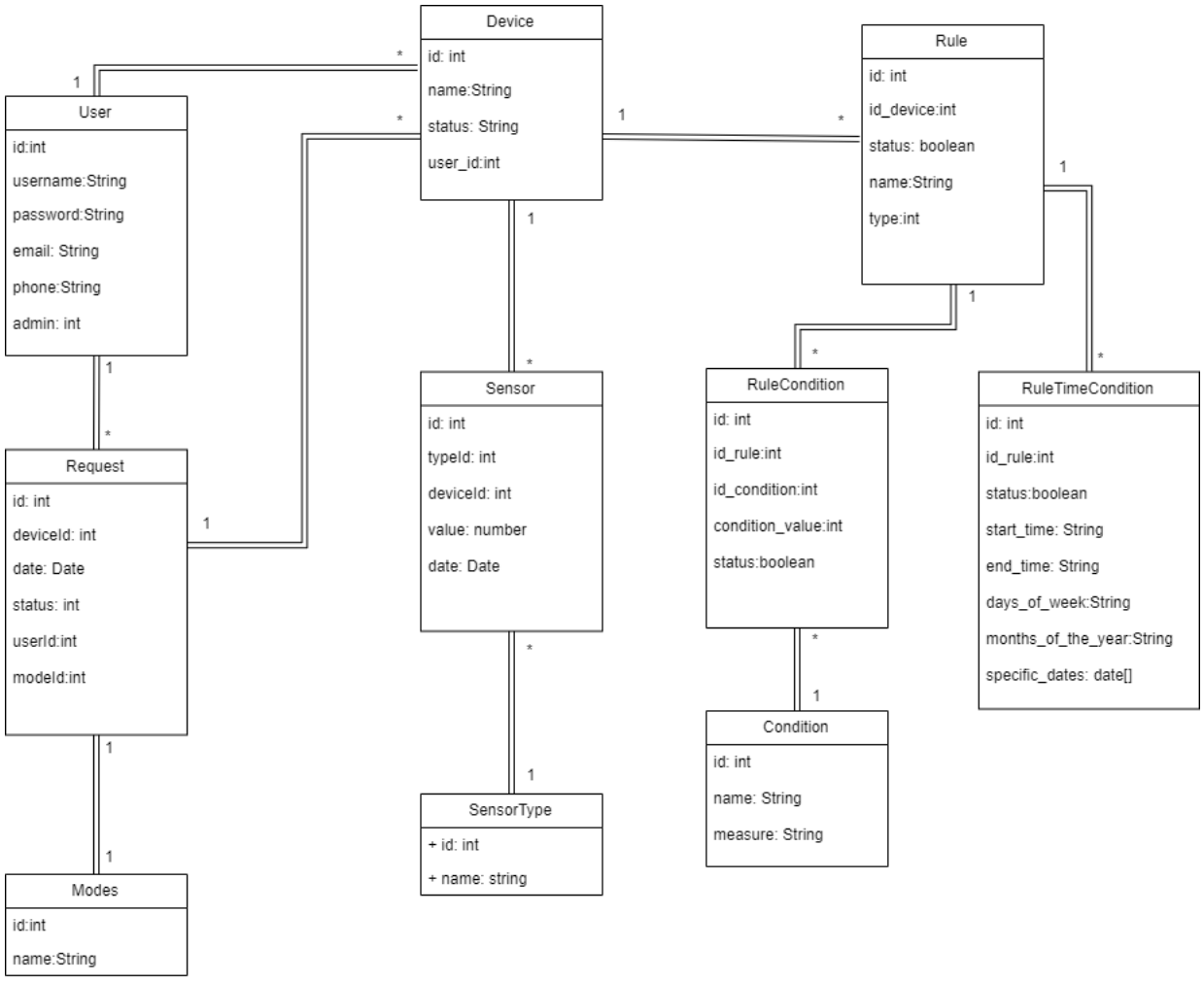


Figure 5: Data model UML

In this data model diagram we can see how our data is structured and related with the different datatables. We explain the target of the different tables:

- **User**: It stores all the data related to our users.
- **Request**: It is used to store the requests of the users to a specific device and to check if these requests have been completed.
- **Mode**: It stores the different kind of requests that a user can do like manual irrigations, conditional irrigations, the obtainment of the sensor data.
- **Device**: It stores the information of a device which corresponds to a user. It can have a default rule, so the user has not need to create its own.
- **Sensor**: It stores the value of a sensor of a specific type which correspond to a device.
- **SensorType**: It stores the different type of sensors which are: Humidity, Temperature, Soil.
- **Rule**: It stores all the configurations which correspond to a device. It can have RuleConditions and TimeConditions.
- **RuleCondition**: It stores the conditions with its respective value. It corresponds to a rule and a condition. If the sensors data of a device accomplish these conditions then an irrigation is produced.

- **RuleTimeCondition**: It corresponds to the conditions of a rule which a user decides when an irrigation has to be produced. With the status we determine if this condition has to be considered.
- **Condition**: It stores the different conditions which are: Temperature is lower than,Temperature is greater than, Humidity is lower than, Humidity is greater than, Soil is lower than, Soil is greater than.

We have made some changes regarding the data model of the previous sprint in:

- Added field *admin* (int) in User.
- Added fields *image* (image), *default_rule* (int) and *status rule* (bool) in Device.
- Removed field *type* (int) in Condition and added *measure* (String).
- Fields *start_time* and *end_time* in RuleTimeCondition switched their type from date to String. *specific_dates* switched from String to date array.

7 Main screens design and navigation

- Link Android Design:

  Click here: Android Screens Navigation

- Brief explanation of the different screens in the Android application:

  The initial screen shows the list of available devices, being possible to consult their previous information, as well as to activate and deactivate them.

  When you click on a device from the list, the application takes you to its management panel. In it, we can view detailed information, access its configuration and activate manual irrigation. If we activate manual irrigation, we will be given a pop-up with the option of specifying for how long we want the irrigation to take place. If there is a current irrigation, the panel will show the remaining irrigation time and the option to cancel it.

  By accessing the device configuration, the user is allowed to change the name of the device and the frequency of updating the sensor information, as well as save the changes and access the configuration of irrigation rules.

  If you access the watering rule panel, a list of all current rules will be displayed, and you will be able to activate, deactivate, delete and modify them. In addition, you can add new watering rules to the list and save your changes.

  If you enter the internal configuration of a rule, you will be shown the conditions that make up the rule. All of them can be activated, deactivated, deleted and modified. The value of the condition will depend on its type. You can change the changes and add new conditions.

  The conditions can be sensor related or time related. Weather conditions offer a number of options to specify at which time intervals they will be met. Those options related to dates will offer a calendar to the user, where he can specify the desired days, and those related to hours and minutes will offer a clock for the same.

  The improvements over the previous sprint are reflected in the style and layout, having added the use of buttons and screen designs more sophisticated, aesthetic and intuitive.

- Link Web Design:

  Click here: Web Screens Navigation

- Brief explanation of the different screens in the Web application:

  Screen 1 shows information about the statistics of our application like RPI petitions, available server ram, cpu, ...

  Screen 2 shows the control panel in order to list, create, edit or eliminate users.

  Screen 3 shows the control panel related to the operation of creating a user. Admin have to enter: user name, email, password and phone.

  Screen 4 shows the control panel related to the operation of editing a user account. Admin can modify whatever he wants (username, email, password and phone).

  Screen 5 shows the control panel related to the devices for each user in order to list, create, edit or eliminate devices.

  Screen 6 shows the control panel related to the operation of creating a device.

  Screen 7 shows the control panel related to the operation of editing a device.

Screen 8 shows the control panel related to the conditions of a rule in order to list, create, edit or eliminate conditions. **New from this sprint:** it will now also display the rule time conditions.

Screen 9 shows the control panel related to the operation of creating a condition.

Screen 10 shows the control panel in order to list, create, edit or eliminate rules.

Screen 11 shows the control panel related to the operation of editing a rule.

Screen 12 shows the control panel related to the operation of creating a rule.

New from this sprint:

Screen 13 shows the control panel related to the operation of creating a rule condition.

Screen 14 shows the control panel related to the operation of editing a rule condition.

Screen 15 shows the control panel related to the operation of creating a rule time condition.

Screen 16 shows the control panel related to the operation of editing a rule time condition.

8 Financial Factors

The financial factors analysis is present in the Market and economic study document.