UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escuela Técnica Superior de Ingeniería Informática

Universitat Politècnica de València

Desarrollo del gestor de datos y casos para el soporte a un sistema multiagente para un entorno paisajístico

Proyecto Final de Carrera

Grado en ingeniería informática

Autor: Ferrer Mestre, Ramón

Director: Poza Luján, José Luis

Director: Carrascosa Casamayor, Carlos

2016-2017

Resumen

En la actualidad las tecnologías desarrolladas por el campo de la informática están incorporándose a diversos campos del conocimiento. La finalidad de esta cooperación es simplificar, mejorar y automatizar las tareas de estos campos.

En este proyecto se presenta el primer prototipo para la generación de un jardín automático. Este jardín estará compuesto por robots que monitoricen y controlen el estado de bienestar de la planta.

Este sistema debe ser capaz de tomar decisiones relativas a cada situación. Para ello se ha dotado al sistema de memoria sobre las situaciones previas y las acciones tomadas, junto a la capacidad de adoptar acciones de circunstancias similares en el caso de situaciones nuevas.

Adicionalmente debe ser capaz de dar acceso a la información monitorizada y su histórico. Para ello se ha guardado esta información en una base de datos y se ha habilitado el acceso mediante un *API REST*.

Palabras clave: Base de datos, Artefactos, Jardines, Comunicación, Plantas, Chombo, Prototipo.

Abstract

At present the technologies developed by the field of computer science are being incorporated into various fields of knowledge. The purpose of this cooperation is to simplify, improve and automate the tasks of these fields.

In this project is presented the first prototype for the generation of an automatic garden. This garden will be composed of robots that monitor and control the welfare state of the plant.

This system must be able to make decisions on each situation. To this end, the system has been equipped with memory about previous situations and actions taken, along with the ability to adopt actions of similar circumstances in the case of new situations.

Additionally, it must be able to give access to monitored information and its history. To do this, this information has been saved in a database and access has been enabled through a REST API.

Key words: Database, Artefact, Garden, Communication, Plants, Chombo, Prototype.

Resum

En la actualitat les tecnologies desenvolupades al camp de la informàtica estan incorporant se a altres camps del saber. La finalitat de esta cooperació es simplificar, millorar y automatitzar les tasques d'aquests camps.

En este projecte es presenta el primer prototip per la generació d'un jardí automatitzat . Aquest jardí estarà composat per robots que monitoritzen y controlen el estat de benestar de les plantes

Es menester que el sistema siga capaç de prendre decisions relatives a cada situació. Per aquest motiu s'ha dotat al sistema de la capacitat de recordar les situacions anteriors y les accions realitzades. A més a més s'ha incorporat la capacitat d'adoptar accions de circumstancies semblants en el cas d'aparèixer situacions noves.

A més, es necessari donar accés a la informació monitoritzada amb caràcter temporal. Amb aquest fi s'ha guardat aquesta informació en una base de dades y s'ha habilitat el accés amb un *API REST* .

Paraules clau: Base de dades, Artefacte, Jardí, Comunicació, Planta, Chombo, Prototip.

Tabla de contenidos

Contenido

Ilustraciones

Tablas

1. Introducción

1.1 Motivación

Según la RAE el paisajismo es el “estudio o diseño del entorno natural, especialmente de parques y jardines”. Siendo una actividad muy relacionada con el cultivo, aunque desde un punto de vista más estético. Las primeras presencias de esta actividad se remontan a la antigua Babilonia, con unos de las grandes maravillas del mundo antiguo como son los jardines colgantes.

En la actualidad es una actividad que está presente en todo nuestro entorno cotidiano desde los jardines privados, hasta parques públicos. Y aun siendo tan extendida, el progreso en este campo continúa estancado, manteniendo procesos arcaicos en el cuidado de las plantas.

Las plantas son seres vivos que dependen del entorno para su supervivencia, dado que generalmente no tienen habilidad motriz.

Sería posible mejorar la capacidad de supervivencia de las plantas si se implementara un sistema autónomo que reaccionara de forma inmediata a sus necesidades, mediante acciones como pueden ser: dotarlas de movimiento, riego y fertilizante automáticos, entre otras.

Desde el ámbito de la ingeniería informática consideramos que existen herramientas que podrían dotar a las plantas de los mecanismos anteriores. De forma que se reducirá el trabajo de mantenimiento y la mortalidad de las plantas. Simplificando así la complejidad de la actividad paisajística.

1.2 Objetivos

La finalidad principal de este trabajo es el desarrollo, diseño e integración de diversas tecnologías a nuestro alcance para permitir la supervivencia de las plantas en diversos entornos y circunstancias. Y para poder llevar a cabo un objetivo de esta envergadura, este proyecto se estructurará en cuatro proyectos diferenciados, los cuales deberán coordinarse para realizar cada uno sus propios sub-objetivos sin perder la visión del general.

Siendo los participantes de cada sub-proyecto:

Nombre	Ramón Ferrer
Rol	Desarrollador del sistema de persistencia y comunicaciones
Objetivo	Implementar la persistencia

Nombre	Sergio Rives
Rol	Desarrollador de la interfaz de usuario (IU) y el *render*
Objetivo	Implementación en android de la IU

Nombre	Alberto Martín
Rol	Desarrollador del sistema multi-agente y *manager* (simulador)
Objetivo	Implementar el sistema de toma de decisiones

Nombre	Salvador Pons
Rol	Desarrollador *hardware*
Objetivo	Implementar funciones del Arduino y montar el hardware

En este proyecto nos centraremos en el ámbito de la comunicación y persistencia de la información obtenida por los distintos componentes del sistema. Marcando como hitos la consecución de los siguientes puntos.

- El sistema debe apoyar el conocimiento basado en casos mediante la consulta, modificación e inserción de casos.

- La comunicación permite el acceso a los datos históricos de la red de sensores creada por los *Chombos*(Maceta-robot controlada por un agente sin o con conexión con el sistema de agentes) del jardín desde las diferentes interfaces de usuario que se deseen utilizar.

- La información característica de las especies debe ser accesible por el sistema multi-agente.

2. Estudio de Mercado

2.1 Competidores potenciales/tecnologías similares

2.1.1 Parrot Pot [50]

Maceta inteligente ya a la venta a un precio de 150 $, cuenta con 4 tipos de sensores: sensor de PH, de temperatura, de luz y de humedad. Si la planta tuviese alguna necesidad la maceta envía notificaciones vía bluetooth al móvil para poder atenderlas. Como función autónoma puede autorregularse si la planta necesita agua y está equipada con un tanque de agua a tal efecto. Funciona con 4 pilas AA que le dan una autonomía de 1 año. Utiliza la tecnología Flower Power. Características:

- **SENSORS**
  - Capacitive soil moisture sensor (measurement range: 0 to 50%)
  - Fertiliser level sensor
  - Light sensor: 0 to 1000 µmol m-2. s-1
  - Air temperature sensor: 0°C to +55°C
  - Rain and water resistant (IPX5 rating)
- **SIZE**
  - Diameter x height: 20.5 cm x 31.2 cm
- **DETAILS**
  - Operating temperature: 0°C to 55°C
  - Material: ABS, PP, rubber
  - LED status indicator (red, green)
  - Built-in water tank: 2.2 litres
  - Soil volume: 2.4 litres
  - Batteries: 4 AA
- **CONNECTIVITY & APPS**
  - Connection via Bluetooth® Smart/Bluetooth V4.0 BLE
  - Free Parrot Flower Power app
  - Application programming interface (API)

2.1.2 Flower Power[51]

Tecnología usada por las macetas Parrot Pot, consistente en un dispositivo bluetooth adaptable a cualquier maceta de tamaño medio/grande que contiene 4 tipos de sensores: sensor de PH, de temperatura, de luz y de humedad. Gracias a una conexión con una base de datos de más de 8000 plantas [49], es capaz de enviar alertas al móvil por bluetooth si las condiciones de la planta no son óptimas.

Mediante su aplicación móvil es posible monitorizar el estado de una o más plantas, consultar un histórico con las condiciones de cada planta o planificar tareas a realizar para que las plantas estén en las mejores condiciones, siendo esta planificación automática.

El dispositivo funciona con una pila AAA y su precio es de 50 $.

Ilustración 1 Flower Power
(extraída de https://www.amazon.com/Parrot-Flower-Power-Bluetooth-dedicated/dp/B00FOM2Y6W)

2.1.3 RoseRunner (2012)

Consiste en un robot capaz de moverse entre las macetas, fertilizarlas y moverlas de un punto a otro.

- Interfaz de control
    - Joystick
    - PC interfaz

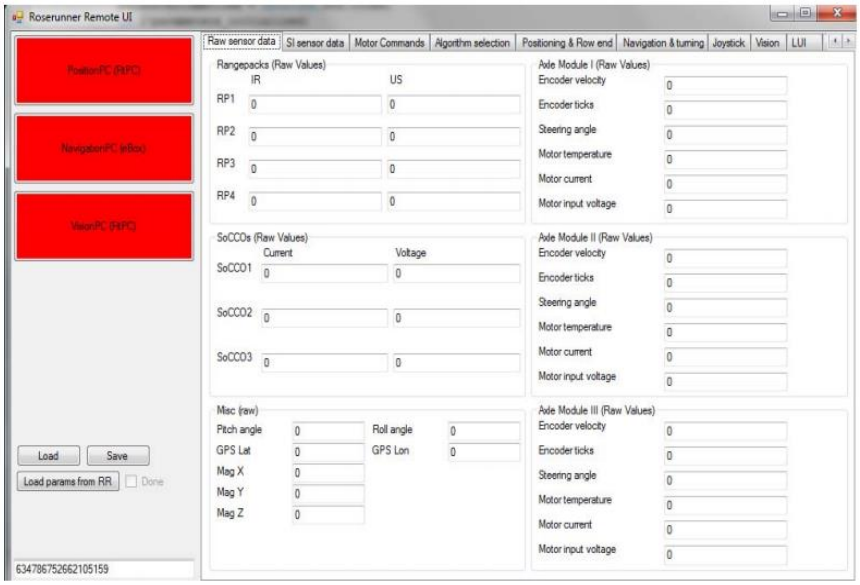


Ilustración 2 Interfaz de usuario

Ilustración 3 RoseRunner Hardware(extraída de http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.259.3801&rep=rep1&type=pdf)

2.1.4 Jurema Action Plant

Máquina bio-interactiva capaz de transportar una planta sensitiva (Mimosa Pudica), intenta aumentar las funciones que puede realizar una planta habilitando que usen tecnologías similares a las que usan los humanos. Usa unas señales eléctricas que circulan a través de las células de las plantas como si de nervios humanos se tratase, y permite que la planta se aleje de un obstáculo al tocarlo, usando la propia planta y sus señales "nerviosas" como sensor. Adicionalmente dispone de un tanque de agua y una placa con los componentes electrónicos, la cual interpreta las señales y necesidades de la planta para traducirlas en movimiento o dispensación de agua.

Ilustración 4 Arquitectura de Jurema (extraida de https://ivanhenriques.com/2011/06/02/jurema-action-plant/)

2.1.5 Plantas nómadas

La Planta Nómada es un organismo vivo, constituido por un sistema robótico, una especie vegetal orgánica, un conjunto de celdas de combustible microbianas y fotovoltaicas. Para sobrevivir, este organismo toma agua contaminada y la procesa en sus celdas de combustible mediante una colonia de bacterias autóctonas de estas aguas, que se alimentan transformando los nutrientes en electricidad, para ser almacenada por su sistema de cosecha de energía. En este proceso de biodegradación mejora la calidad del agua y provee a la especie vegetal que también produce electricidad con su metabolismo. La liberación de oxígeno es el remanente de este ciclo energético. Por tanto, no solo es una especie adaptada al entorno modificado, sino que también restituye la energía que dispone de la tierra.

Ilustración 5 Prototipo de Plantas Nomadas (extraida de http://www.plantasnomadas.com/)

2.1.6 Fliwer

Sistema de riego inteligente para huertos, jardines y macetas. Es una tecnología que valora las necesidades de las plantas en tiempo real para proporcionar un riego adecuado, así como proporcionar información al usuario mediante WiFi o 3G sobre el estado de las mismas.

Ilustración 6 Fliwer (extracción http://www.fliwer.com)

2.1.7 Click & Grow Smart flowerbed

Semillero inteligente que proporciona los cuidados suficientes para que las plantas broten en 1-2 semanas, cuenta con un pequeño depósito y se encarga de que las semillas tengan las cantidades adecuadas de agua, oxígeno y nutrientes.
Utiliza 4 pilas AA y el precio oscila entre 20-60€ según el modelo.

Ilustración 7 Click Grown (extraída de https://www.amazon.com/Click-Grow-flowerbed-Cockscomb-Indoor/dp/B008K95K80)

2.1.8 A Control Method for a Swarm of Plant Pot Robots that Uses Artificial Potential Fields for Effective Utilization of Sunlight (CMSPR)

Es un proyecto de la universidad de Tokio de agricultura y tecnología, desarrollado por Masato Yuasa y Ikuo Mizuuchi. Que tiene como objetivo maximizar el uso de la luz solar en el cultivo de plantas. El sistema que implementan se basa en una serie de robots capaces de moverse y una serie de sensores instalados en la maceta, con el fin de monitorizar de la planta. El funcionamiento es realizado por medio de la transmisión de la información de estado de los sensores al ordenador de control, este calcula la orden y la envía a los robots para su ejecución.

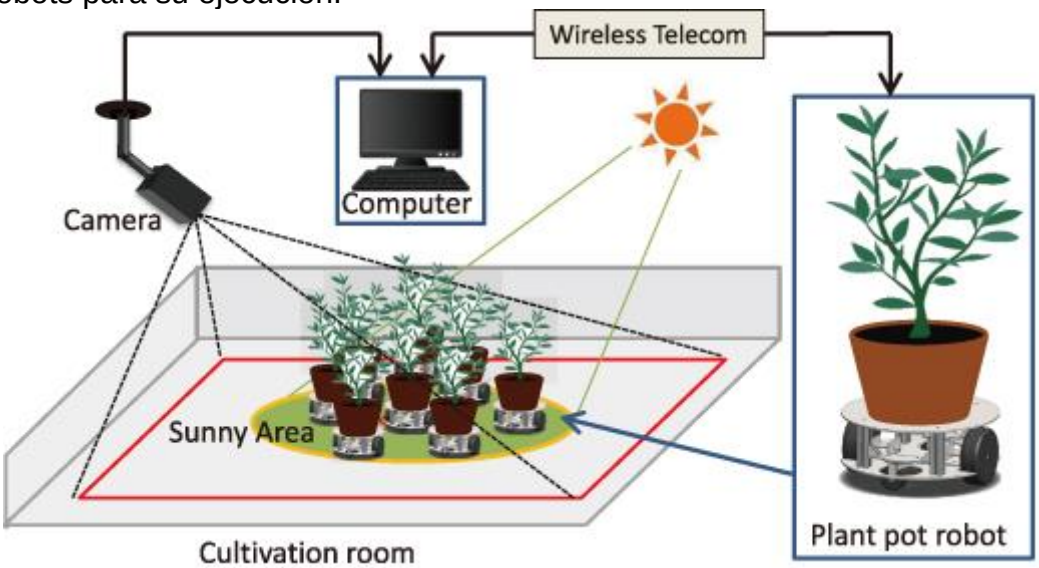


Ilustración 8 Arquitectura del CMSPR (extraída de https://www.fujipress.jp/jrm/rb/robot002600040505/)

2.1.9 PotPet [52]

Este sistema tiene el objetivo de permitir a las plantas el moverse a hacia lugares soleados (sensor de luz) o detectar personas (sensor de movimiento), además puede advertir a las personas de la necesidad de riego y del punto de parada del riego mediante el sensor de humedad.

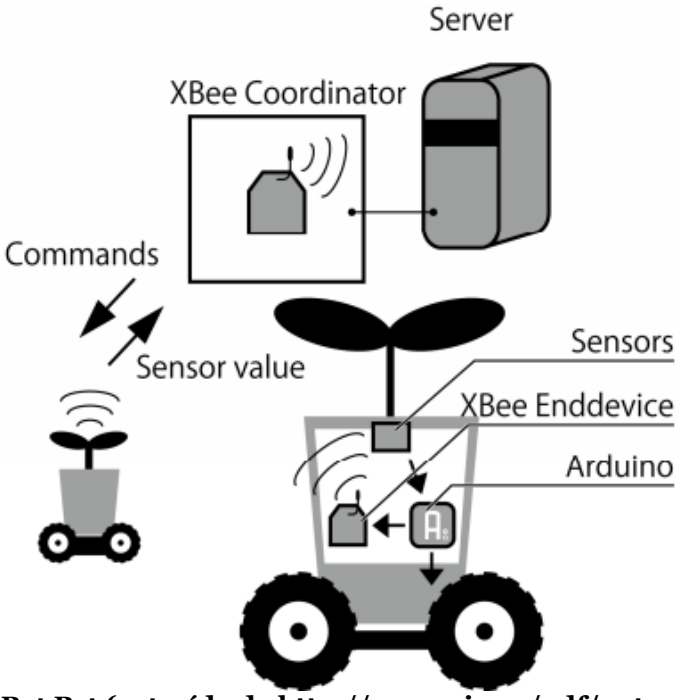


Ilustración 9 Pot Pet (extraída de http://sappari.org/pdf/potpet_tei2011.pdf)

2.2 Tabla comparativa

	Parrot Pot	Flower Power	Rose Runner	Jurema Action Plant	Planta nómada	Fliwer	Click & Grow	Maceta intel.
Sensor humedad	✓	✓	✗	✓	✓	✓	✓	✓
Sensor fertilizante	✓	✓	✓	?	?	✓	✓	✓
Sensor luz solar	✓	✓	✗	?	✓	✓	✗	✓
Sensor temperatura	✓	✓	✗	?	✗	✓	✗	✓
Riego automático	✓	✗	✓	✓	✓	✓	✓	✓
Depósito agua	✓	✗	✓	✓	✗	✗	✓	✓
Movimiento	✗	✗	✓	✓	✓	✗	✗	✓
Alertas	✓	✓	✗	✗	✗	✓	✗	✓
Interior	✓	✓	✗	✓	✗	✓	✓	✓
Exterior	✓	✓	✓	✓	✓	✓	✗	✓
Simulación	✗	✗	✓	✗	✗	✓	✗	✓
Precio	150 $	50 $	?	?	?	1200 €	20-60 €	?
Disponible	✓	✓	✗	✗	✗	✓	✓	✗

Tabla 1 Comparación de productos en el mercado (primera parte)

	Parrot Pot	Flower Power	Rose Runner	Jurema Action Plant	Planta nómada	Fliwer	PotPet	CMSPR	Click & Grow	Maceta intel.
Sensor humedad	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓
Sensor fertilizante	✓	✓	✓	?	?	✓	✗	?	✓	✓
Sensor luz solar	✓	✓	✗	?	✓	✓	✓	✓	✗	✓
Sensor temperatura	✓	✓	✗	?	✗	✓	✗	✓	✗	✓
Sensor de movimiento	✗	✗	✗	✗	✗	✗	✓	?	✗	✗
Riego automático	✓	✗	✓	✓	✓	✓	✗	?	✓	✓
Depósito agua	✓	✗	✓	✓	✗	✗	✗	?	✓	✓
Movimiento	✗	✗	✓	✓	✓	✗	✓	✓	✗	✓
Alertas	✓	✓	✗	✗	✗	✓	✓	?	✗	✓
Interior	✓	✓	✗	✓	✗	✓	?	✓	✓	✓
Exterior	✓	✓	✓	✓	✓	✓	✓	?	✗	✓
Simulación	✗	✗	✓	✗	✗	✓	✗	✓	✗	✓
Disponible	✓	✓	✗	✗	✗	✓	✗	?	✓	✗

Tabla 2 Comparativa de mercado (segunda parte)

3. Análisis previo al diseño

Durante este capítulo se presentará el análisis de las características recomendables para el sistema, y la selección de tecnologías para el desarrollo de la persistencia del histórico de valores de actuadores y sensores, la comunicación entre la persistencia y el resto de componentes, y el apoyo al conocimiento basado en reglas del sistema multi-agente.

3.1 Características deseables

En este apartado se han especificado una serie de características basadas en el análisis de los sistemas anteriores. Estas características buscan proveer al diseño de los puntos clave para ser competitivo en este mercado. Por otro lado, se han derivado requisitos del objetivo final. En este proyecto se ha optado por codificar estas características de la forma siguiente.

CA1: Persistencia de la información relativa a los casos.
CA2: Integración de datos útiles para la evaluación de casos.
CA3: Disponibilidad 24x7 del histórico de casos.
CA4: Escalable con el incremento de macetas en el sistema.
CA5: Acceso de la información desde el Sistema de Agentes.
CA6: Guardar y modificar las valoraciones de los casos.
CA7: Preparación para incorporar aprendizaje.
CA8: Descarga de datos históricos en formatos abiertos.
CA9: Comunicación escalable y flexible entre los distintos componentes del sistema

3.2 Tecnología a emplear

En base al análisis del mercado y las características que se desean conseguir, es posible descartar y argumentar para este sistema concreto los beneficios y perjuicios de las diferentes tecnologías que se han encontrado en la bibliografía.

3.2.1 Persistencia para datos históricos de actuadores y sensores

El análisis ha tomado como punto de partida la capa de persistencia realizada en los proyectos anteriores [3]. Esta se implementó con MySQL, debido a que en este proyecto se guardan datos simples y en su análisis de mercado era uno de los sistemas más utilizados en los sistemas similares. En nuestro caso no se ha encontrado información del software usado por los sistemas similares que compiten en nuestro mercado. Por este hecho se ha considerado interesante realizar una comparativa con otras tecnologías [19] [17] [24].

Tecnología de base de datos	Orientación o Tipo de estructura	ACID	Distribución	Entorno recomendado
MySQL	Relacional	Si	Permite el particionado de tablas, pero las transacciones y los joins son ineficientes	Información que no puede quedar nunca inconsistente
Neo4j	Grafos	Si	Permite la distribución mediante un sistema de nodos cores(I/O) con quórum y nodos réplica(O)	Información altamente conexa
Orient DB	Grafos	Si	Permite la distribución mediante un sistema de nodos maestro(I/O) con quórum y nodos esclavos(O). Además, permite sharding	Información altamente conexa. Sistema distribuido con el que escalar
Couchbase	Documentos	Si (1)	Permite la distribución mediante el protocolo DCP [15]	Sistema distribuido en zonas alejadas geográficamente
Mongo DB	Documentos	Si (1)	Permite la distribución mediante un conjunto de nodos mongo y mongod	Datos con mucha variabilidad de esquema. Eficiente para gran variedad de consultas. Sistemas altamente distribuidos
HBase	*BigTable*	Si(2)	Permite la distribución mediante un sistema compuesto por *ZooKeeper*, *HMaster* y *RegionServer* [13]	Entorno con gran capacidad de cómputo y grandes cantidades de información

Tabla 3 Comparativa de tecnologías de base de datos

1. Mediante una comprobación en dos fases como se puede comprobar en [16] [22]
2. Nivel de aislamiento en lectura confirmada como se puede ver en [18]

Ilustración 10 Posición de mercado de las bases de datos relcionales

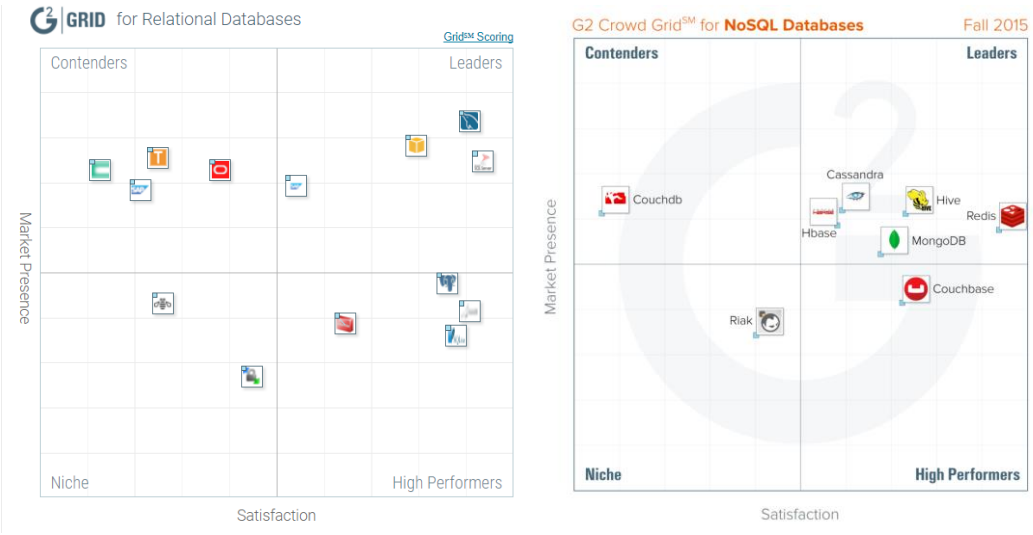


Ilustración 11 Posición de mercado Bases de Datos NoSQL

Gráficas sobre posición de mercado de BBDD obtenidas de https://www.g2crowd.com/

Como se ve en las gráficas anteriores MySQL se encuentra en una posición interesante de uso entre las bases de datos relacionales. En principio por las características vistas en la comparativa y la bibliografía es una buena opción para la realización de un primer prototipo que no necesite ser demasiado escalable. Por otra parte, es interesante ver [4] y [5], donde se observaron implementaciones de sistemas de persistencia distintos a las usuales bases de datos relacionales: Aplicando otros tipos de orientación para la mejora del rendimiento en entornos distribuidos.

En el caso de nuestro sistema es interesante la idea de poder generar un sistema que en conjunto pueda aumentar su capacidad de almacenamiento y de atender lecturas con el aumento del número de Chombos. Por este motivo se podría considerar interesante el seleccionar MongoDB como base para la persistencia del sistema, pero el sistema que se está tratando de realizar se encuentra en una fase demasiado joven para asociarle la complejidad de la "*clusterización"* de datos.

Por este motivo, la implementación se realizará sobre un servidor único y los Chombos no entregaran parte de su capacidad de procesamiento y almacenamiento para la distribución de datos. Por este mismo motivo se preparará para un futuro el sistema de persistencia con MongoDB.

Por otro lado, se descarta Apache Hadoop debido a la falta de datos que se prevé para el sistema, dado que se supone que las bases de datos son únicas para cada jardín, se considera que el número de información por minuto será elevado, pero dentro de lo que se considera un volumen moderado. Estas previsiones pueden cambiar en futuras iteraciones del sistema

3.2.2 Comunicación entre la persistencia y el resto de componentes

En este apartado se busca la selección de una tecnología que permita integrar la capa de persistencia del histórico con el resto de componentes del sistema global. Con este objetivo es posible observar proyectos en otros ámbitos, pero integrados dentro de la temática de las redes de sensores o IoT (internet de las cosas). Si se presta atención al número de tecnologías que se utilizan en este ámbito pudiéndose ver que hay una abundante cantidad debido a ser un ámbito en expansión, por este motivo se tomará como referencia la selección realizada por [1] y algunas propuestas realizadas por los tutores del proyecto.

Protocolos	Tecnologías	Esquema Pub/Sub	Esquema req/res	Tamaño de cabecera	Protocolo de transporte
HTTP	REST (Express, Jersey...)	✗	✓		TCP
ZMTP	0MQ	✓	✓		TCP
COAP	Californium,Leshan, Node-coap	✓	✓	4B	UDP
MQTT	[2]	✓	✗	2B	TCP
AMQP	RabbitMQ	✓	✗	8B	TCP
DDS	Vortex Café	✓	✗		TCP/UDP
XMPP	Clients(Babbler,JSJaC…), Server(Apache Vysper…)	✓	✓		TCP

Tabla 4 Comparativa de protocolos de comunicación

De las tecnologías anteriores, una de las más destacadas actualmente es MQTT: *"MQTT is a messaging protocol that was introduced by Andy Stanford Clark of IBM and Arlen Nipper of Arcom (now Eurotech) in 1999 and was standardized in 2013 at OASIS [70]. MQTT aims at connecting embedded devices and networks with applications and middleware. The connection operation uses a routing mechanism(one-to-one,one-to-many,many-to-many) and enables MQTT as an optimal connection protocol for the IoT and M2M."* [1].
Esta tecnología destaca por el uso reducido de ancho de banda, propiedad que es especialmente interesante en este proyecto donde el tráfico es bastante elevado entre los mensajes de los agentes y el de los sensores, además en este caso el guardado de la información histórica es menos prioritario en comparación al funcionamiento del agente, dado que el fin último de este proyecto es generar

un sistema autónomo que pueda proveer a las plantas de acciones que las permitan sobrevivir.

Entre las tecnologías que implementan este protocolo están las que aparecen en [2] para la parte del *broker*, de éstas es interesante destacar la comparativa realizada por [7]. Partiendo de estos datos, es interesante pensar en Mosquette que es una implementación en java del *broker* MQTT y además puede ser embebido, dándonos mayor control sobre la comunicación. En cambio, tampoco podemos descartar una posible infraestructura que sea más flexible, dado que Mosquette tiene como desventaja que es rígido y que no es multiprotocolo. Estas cuestiones pueden llevar a que se descarte una posible implementación de la comunicación con Mosquette en el futuro, por esto es interesante poder plantear otras opciones como RabbitMQ que es un *broker* multiprotocolo, aunque como desventaja tiene que el protocolo MQTT que implementa esta aun en desarrollo.

Otra tecnología interesante por el hecho de que fue implementada para la comunicación entre las anteriores macetas inteligentes en proyectos anteriores es ZeroMQ.:
"ZeroMQ [19] is a messaging library offering a socket API with more advanced messaging patterns than Berkeley sockets. Supported patterns include request/response and pub/sub. The wire-level protocol is the ZeroMQ Message Transport Protocol (ZMTP), which is made available under the GNU General Public License." [5].

Esta tecnología destaca por ofrecer la posibilidad de crear un sistema bastante complejo con nodos muy simples, como podría ser la combinación de nodos con patrones *req/res* y nodos con patrones pub/sub, por lo tanto, ofrece una gran flexibilidad para el desarrollo de la arquitectura final.

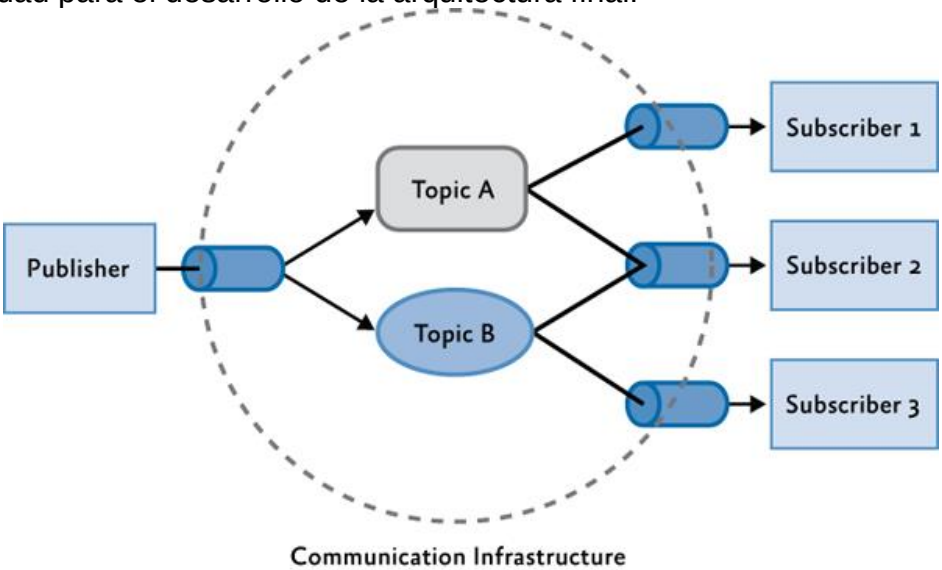


Ilustración 12 Patrón publicador-subscriptor (extraída de https://msdn.microsoft.com/en-us/library/hh404091.aspx)

Por último, cabe destacar la posibilidad de implementar la comunicación mediante REST, dado que es una arquitectura de comunicación que fácilmente puede ser utilizada por todos los componentes del sistema.

Spring MVC es un framework que provee una arquitectura de modelo-vista-controlador, que puede ser utilizada para el desarrollo de aplicaciones web flexibles y con bajo acoplamiento, se podría utilizar para implementar un servidor REST como en el ejemplo [6]. Por otra parte, se podría utilizar Jersey, que es una implementación para el JSR-311.

Una vez vista la gama de opciones para la creación del API web de consulta de datos históricos, es interesante plantear diferentes opciones para necesidades futuras, pero en la actualidad de este proyecto, esta es una parte no prioritaria. Por lo tanto, es interesante pensar en una opción que consuma pocos recursos, además el prototipo solo contará con un servidor. Por esto mismo otra opción viable a generar un servidor *restful* con Spring MVC o Jersey, es utilizar Node.js junto con la librería Express para la generación del servidor *restful*.

Node.js es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Node.js usa un modelo de operaciones E/S sin bloqueo y orientado a eventos, que lo hace liviano y eficiente. Express es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles, con miles de métodos de programa de utilidad HTTP y middleware a su disposición. Combinación que permite la creación de una API REST para la recepción de información de los nodos del sistema y envío de ésta en respuesta.

3.2.3 Apoyo al conocimiento basado en reglas del sistema multiagente

El sistema de agentes estará implementado utilizando Jason, por lo tanto, después de buscar tecnologías compatibles para la intercomunicación entre Jason y el driver de los gestores de base de datos, se encontró CArtAgO[25] que es una infraestructura que hace posible la programación y ejecución de un entorno virtual para sistemas multiagente.

CArtAgO permite desarrollar y ejecutar entornos basados en artefactos, estructurados en *workspaces* abiertos donde pueden unirse agentes de diferentes plataformas para trabajar juntos. CArtAgo facilita la conexión de los agentes con servicios/componentes externos al sistema mediante la creación de artefactos que mediaran entre los agentes y estos servicios/componentes. De esta forma la mediación entre los gestores de base de datos y el sistema de agentes se realizarán con un artefacto que actuará como un API para la entrega de información.

Por otro lado, para la selección de la tecnología para guardar la información de casos y permitir su acceso y modificación, se puede hacer uso de la comparativa del apartado 3.2.1. En este caso el sistema multiagente es un sistema distribuido. Por lo tanto, las decisiones tomadas en ese apartado son válidas para este entorno. Aunque la opción de MongoDB en este entorno tiene otro punto a favor;

como es la flexibilidad que ofrece el *schemaless* para este sistema que no tiene la información definida, sino que depende de los sensores que se añadan.

4. Diseño

Con base en las características que se deben cumplir, seleccionadas en el apartado anterior, y las tecnologías seleccionadas para la realización de cada parte, se presenta el diseño de los diferentes componentes en este capítulo.

4.1 Arquitectura general

El prototipo estará compuesto por diversos agentes, artefactos, bases de datos, hardware e interfaces de usuarios relacionados como se puede observar en la ilustración siguiente.

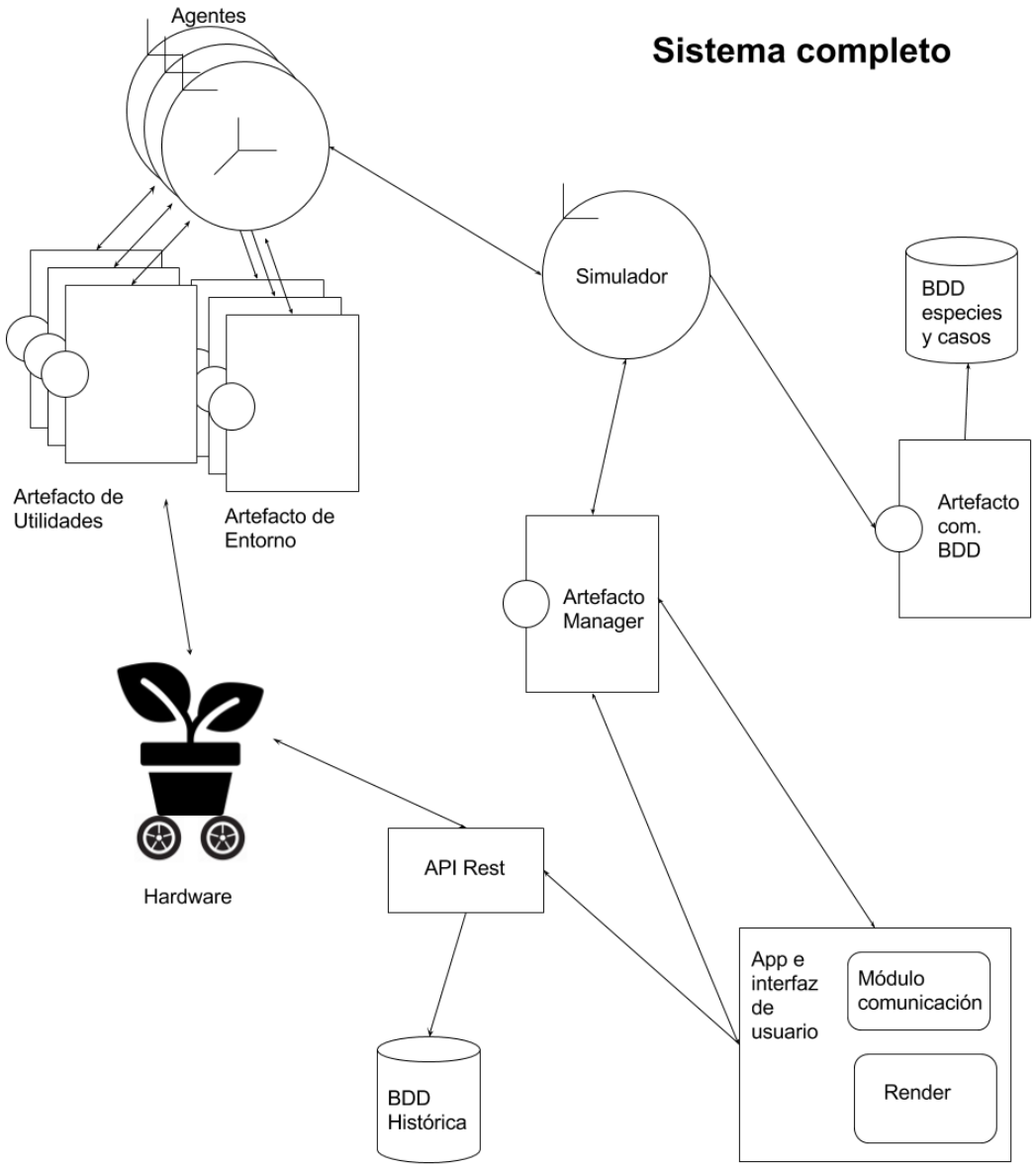
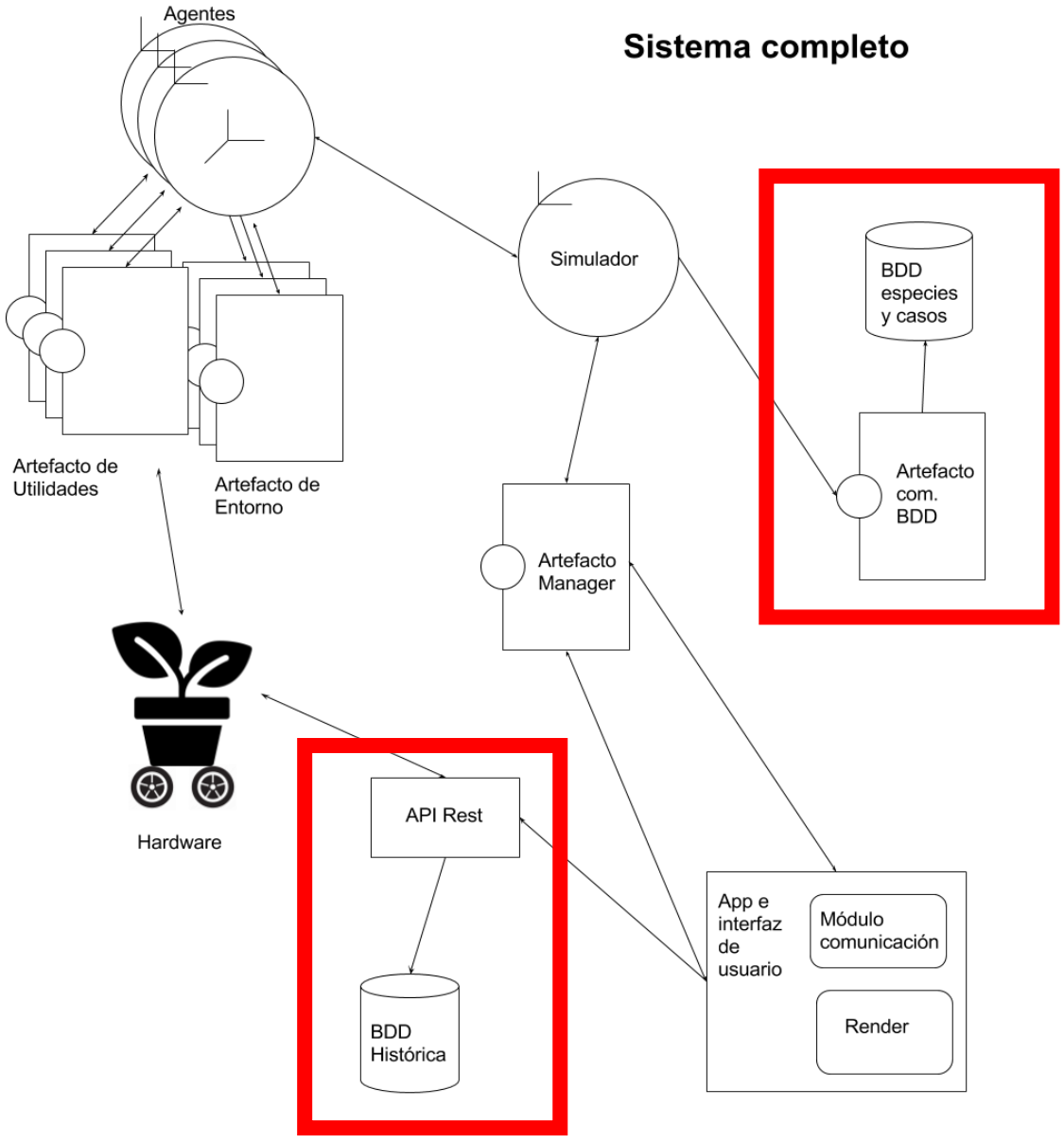


Ilustración 13 Modelo por componentes del sistema global

4.2 Subsistema



Las partes que se presentarán en las siguientes secciones son las remarcadas en rojo sobre la ilustración. Se ha decidido separar la capa de persistencia en dos bases de datos independientes entre ellas. Esta decisión tiene como principio la separación modular por funcionalidad a la que responden, de forma que se pueda separar en nodos distintos, incluso sea posible cambiar la tecnología o el diseño de una sin afectar a la otra. Para este proyecto se han diseñado una base de datos orientada a almacenar la información histórica de los sensores y otra orientada al apoyo de la toma de decisiones.

La comunicación con la persistencia se ha diseñado separada para poder minimizar el acoplamiento del sistema y maximizar la cohesión de los módulos. De forma que la comunicación con la base de especies y casos se realizará

mediante un artefacto, interfaz que proporciona la tecnología CArtAgO para la comunicación con el sistema multiagente. Por otro lado, se ha diseñado el API que comunica la base de datos histórica con la red de sensores y actuadores, y la interfaz de usuario. Ambos módulos de comunicación abstraerán la tecnología en que se implementen las dos bases de datos y su dirección de servició.

4.3 BDD especies y casos

Tomando como punto de partida para iniciar el modelado un esquema de información marcado por los sensores actuales del sistema (Fuentes internas de información). Aunque hay que tener en cuenta que esta información puede ser insuficiente para la resolución de la funcionalidad final del sistema. Por esta razón el diseño de la capa de persistencia debería tener en cuenta la futura existencia de nuevas características para las entidades. Por otra parte, el dominio de estos datos está fuera del campo de estudio de este TFG por lo tanto se han requerido expertos y estándares aceptados en el dominio del problema para no acercar el sistema a terrenos cuestionables por expertos en este dominio. Este punto es crítico en el diseño para la consecución de un apoyo real al conocimiento basado en casos.

Es interesante ver que tenemos dos formas distintas, más bien dos granos distintos, en los que poder guardar la información, como es por Chombo y por planta. Esta división es necesaria para poder entender cómo se desea interpretar la amplia realidad que se quiere poder consultar en las diferentes funciones del sistema. Por esta razón se ha tomado la decisión de guardar por planta. Debido a la gran influencia que representa la especie en las limitaciones de una planta, en estos casos se guardará por especie, utilizando por lo tanto la información para que todos los casos de las distintas plantas de una misma especie en el jardín serán compartidos. Esta aproximación nos da ventajas en cuanto a la generalización, aunque es argumentable que las plantas son individuos únicos que varían de forma que las capacidades de adaptación y resistencia entre otras no sean exactamente exportables de un individuo a otro de la misma especie.

Por lo tanto, Se consideran las especies de plantas como entidades para nuestra base de datos. Estas especies tendrán un listado de casos aprendidos por el sistema o insertados por expertos (en este caso también se podrían aceptar Fuentes externas verificadas por expertos). En este momento es necesario definir qué es un caso en términos de la información que encierra o contiene. Para este proyecto consta de la acción tomada, la especie sobre la que se toma, la bondad de la acción y la situación. De esta especificación de atributos se puede extraer que las situaciones pueden ser iguales o cercanas para muchas especies dado que dependen del entorno no de la especie en sí, mientras que el caso depende de la especie, las acciones y la situación, por lo tanto, las entidades o tablas serían los casos, las especies, las acciones y las situaciones, si se desea tener la base de datos en tercera forma normal.

Como se ha explicado al principio de este apartado, es necesario estructurar la información con carácter de futuro que van a contener las entidades. Dado que el entorno relacional precisa de esquema, el esquema inicial tiene la necesidad

de prever futuros atributos para evitar cambios y migraciones de la base de datos a otro esquema.

La primera aproximación se puede observar en el siguiente diagrama de clases:

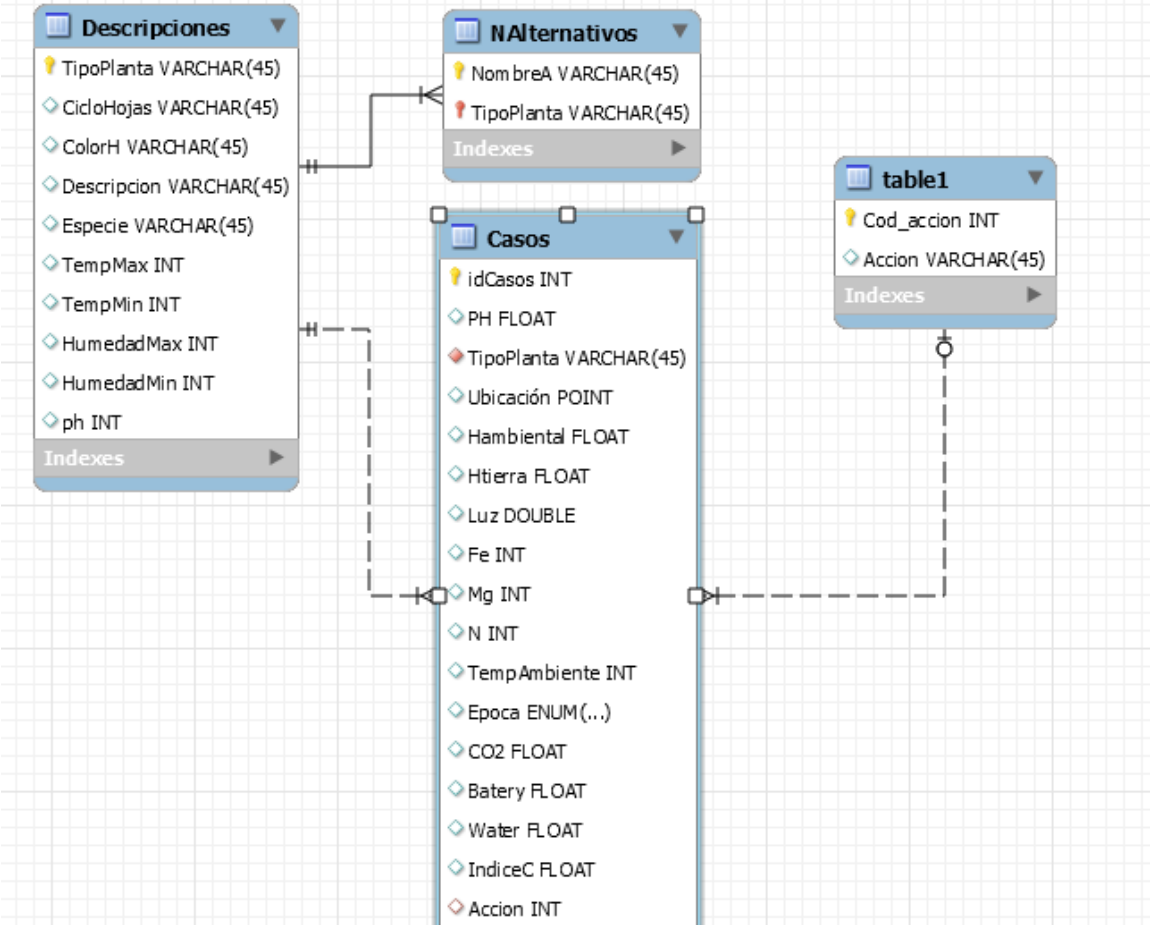


Ilustración 14 Diagrama de clases para la información de especies y casos orientada a la optimización de consultas sobre casos

En este diagrama se puede observar que las situaciones se encuentran integradas dentro de la tabla de casos dado que las búsquedas van a ser de las dos tablas conjuntamente. Esta decisión en otras tecnologías relacionales como Oracle Database se podría realizar esta unión a nivel físico con *clúster tables* [53], pero en MySQL no se ha encontrada esta funcionalidad así que se debe realizar la unión desde el nivel lógico teniendo en cuenta que en principio mejora la velocidad de búsqueda de los datos en las consultas de las dos tablas unidas, además la tabla de situaciones no recibirá ninguna modificación dado que funciona como histórico, también hay que tener en cuenta que este modelo genera un gasto mayor de almacenamiento al guardar datos redundantes.
Por otra parte, se ha encontrado interesante facilitar la búsqueda de las especies de plantas, dando mayor contexto a las especies guardando otros nombres que pueden referenciar a una misma especie con la tabla de nombres alternativos.

Por otra parte, es posible valorar que el número de casos será finito dado que nace de tres conjuntos finitos de datos y no crece con el tiempo. Por lo tanto, la última versión del sistema tendrá pocas inserciones y se concentrará en la búsqueda y actualización de los datos. Por esto mismo en la última versión del

sistema será más eficiente tener las tablas separadas dado que se podrá sesgar las consultas por la tabla de situación usando índices que no se modificaran. Siguiendo este esquema el diseño se ha modelado como apara rece en la ilustración 15:

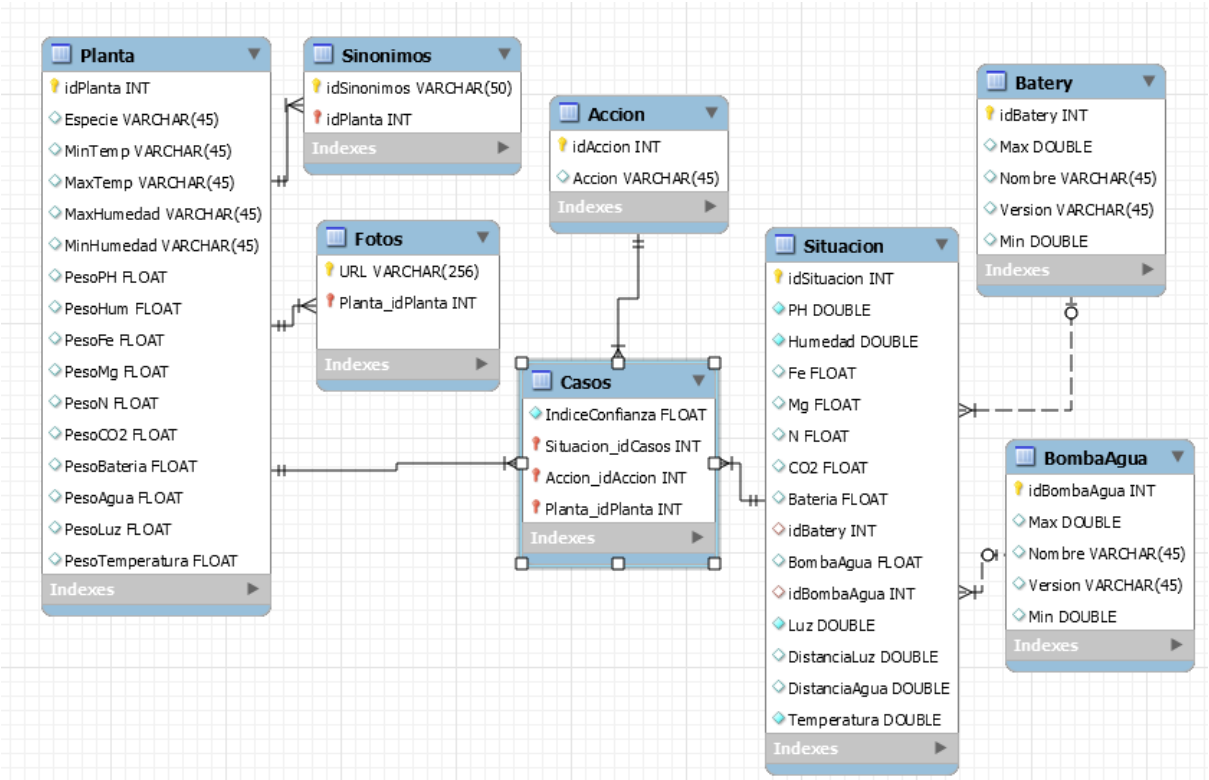


Ilustración 15 Diagrama de clases para la información de especies y casos orientada a la normalización

Donde se ha añadido dos entidades más, como son la bomba de agua y la batería, de forma que se pueda dar acceso a la precisión de las medidas, información que podría ser útil para un futuro análisis de las situaciones.

4.4 BDD histórica

Complementariamente al diseño de la base de datos que entrega la persistencia para realizar el razonamiento basado en casos, se ha diseñado la base de datos histórica para los sensores y actuadores de los chombos, de forma que se busca minimizar el consumo de almacenamiento y la consulta de los datos quedará en un segundo lugar dado que se limitará la consulta a un subconjunto de los datos o se entregarán todos los datos en para consultas de aplicaciones enfocadas en el análisis de datos. Entendiendo estas prioridades y accesos, hay que tener en cuenta que el dominio de estos datos no es finito dado que crece en el tiempo y puede llevar a la creación de una base de datos con una cantidad masiva de información por lo tanto es necesario reducir el número de datos y realizar el control de las inserciones para almacenar únicamente la información relevante. Partiendo de esta base para el diseño se realiza la siguiente aproximación a la realidad:

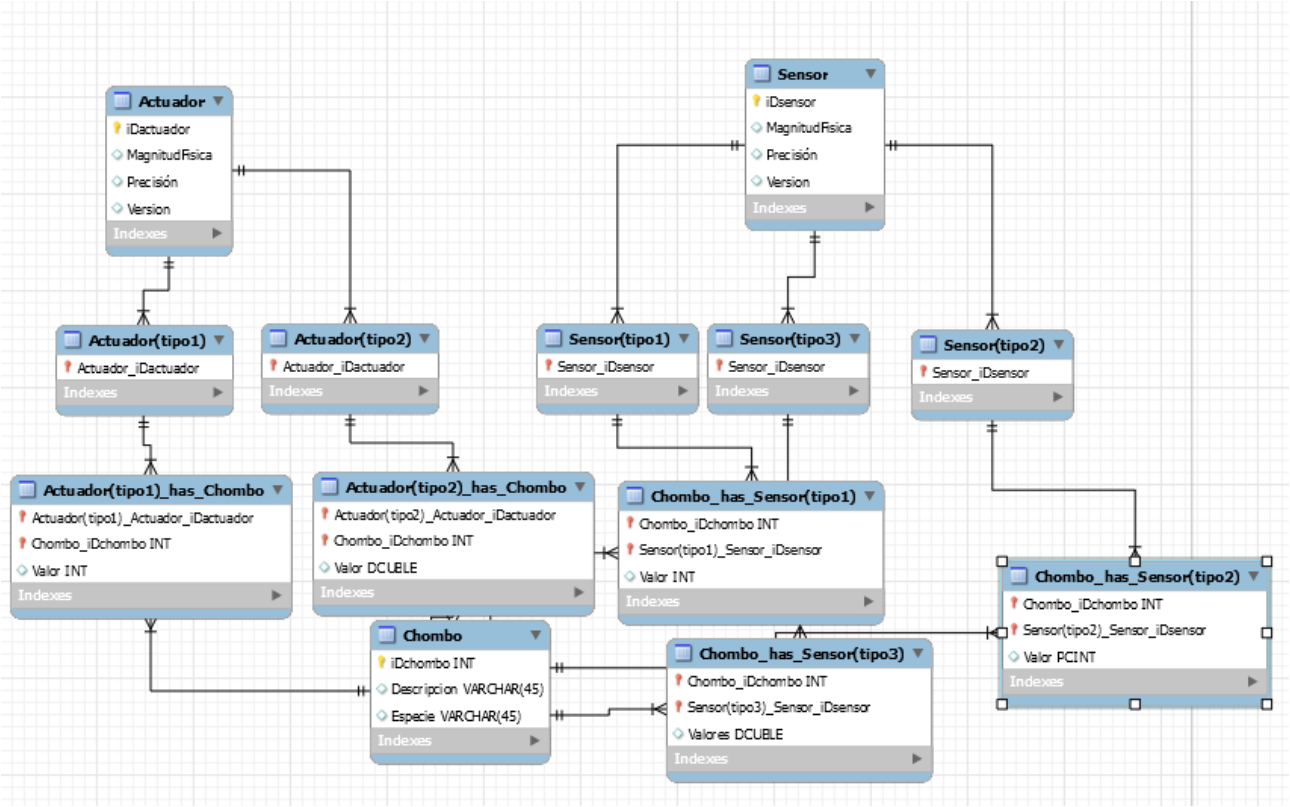


Ilustración 16 Primer diagrama de clases para la información histórica

En este diagrama se ha utilizado herencia para modelar los tipos de sensores y actuadores por tipo de valor que guardan. El padre contiene los atributos comunes de todos los sensores, es decir todos los atributos de la clase sensor. En el caso de los actuadores se ha modelado siguiendo el mismo patrón. La necesidad de utilizar herencia nace de la restricción de esquema que tiene MySQL sobre el tipo de valor que pueden tener las instancias de una tabla. Por esta razón es necesaria la generación de diferentes tablas, una por cada tipo de valor. Por otra parte, con esta solución se consigue restringir que un sensor de un tipo no pueda tener instancias en tablas de otro valor.

El problema de este diseño es que no es nada flexible ante la existencia de nuevos sensores o actuadores con diferentes tipos de valores. Además, la creación de tablas que únicamente tienen una clave ajena como atributo, es una decisión cuestionable desde el punto de vista de optimizar el almacenamiento. También es interesante plantear que las tablas que heredan de sensor o de actuador tendrán una cardinalidad muy limitada. Esto se debe a que son un subconjunto de los datos de sensores y actuadores que ya se prevén con cardinalidad baja al representar los modelos de sensores y actuadores que hay en el mercado.

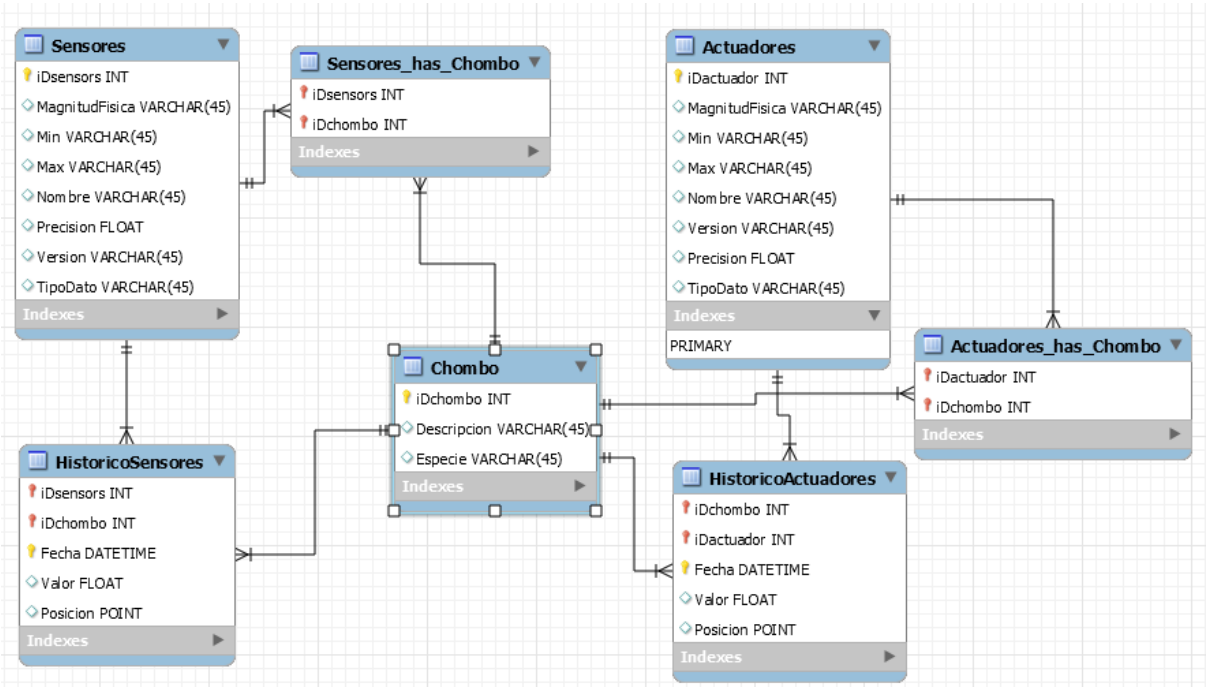


Ilustración 17 Segundo diagrama de clases para la información histórica

En esta aproximación se ha decidido introducir los mínimos y los máximos como caracteres para aumentar la flexibilidad de la inserción, y obligando al control de la veracidad de los datos desde el módulo que inserte la información, por otra parte se ha considerado reducir el tamaño de la variación de la información mediante la inserción de valores derivados del máximo y el mínimo, es decir que el valor insertado se encuentra en el dominio del 0 al 100 siendo 100 el valor de Max del sensor y Min si es 0, cualquier valor intermedio será extraído como un porcentaje, de esta forma se reduce el tamaño de los datos aunque se aumente el tiempo de retorno de los datos reales, y se incrementa la flexibilidad porque se simplifican los tipos de datos para los valores.

Por otra parte, para el guardado de la información de los sensores de posición se utilizará el tipo de datos POINT, que es una clase del OpenGIS y se define como *"A Point is a 0-dimensional geometric object and represents a single location in coordinate space"* [8], y se implementa según el manual de MySQL como aparece en el punto 11.5.2.3 Point Class del manual de referencia de MySQL versión 5.7 al que se puede acceder para mayor información en la cita [9].

4.5 Artefacto comunicación(com.) BDD

El artefacto actúa como un intermediario entre el código java que conecta con el gestor de la base de datos y el agente, de forma que se envuelven los métodos de lectura y escritura de la base de casos mediante el artefacto utilizando CArtAgO para generar una comunicación reactiva entre ambas tecnologías y dotando de mayor abstracción al código del agente, que no depende de la topología lógica de la base de datos ni de la tecnología en la que esta implementada.

Este *wrapper* tendrá los siguientes casos de uso con respecto al agente que vaya a utilizarlo.

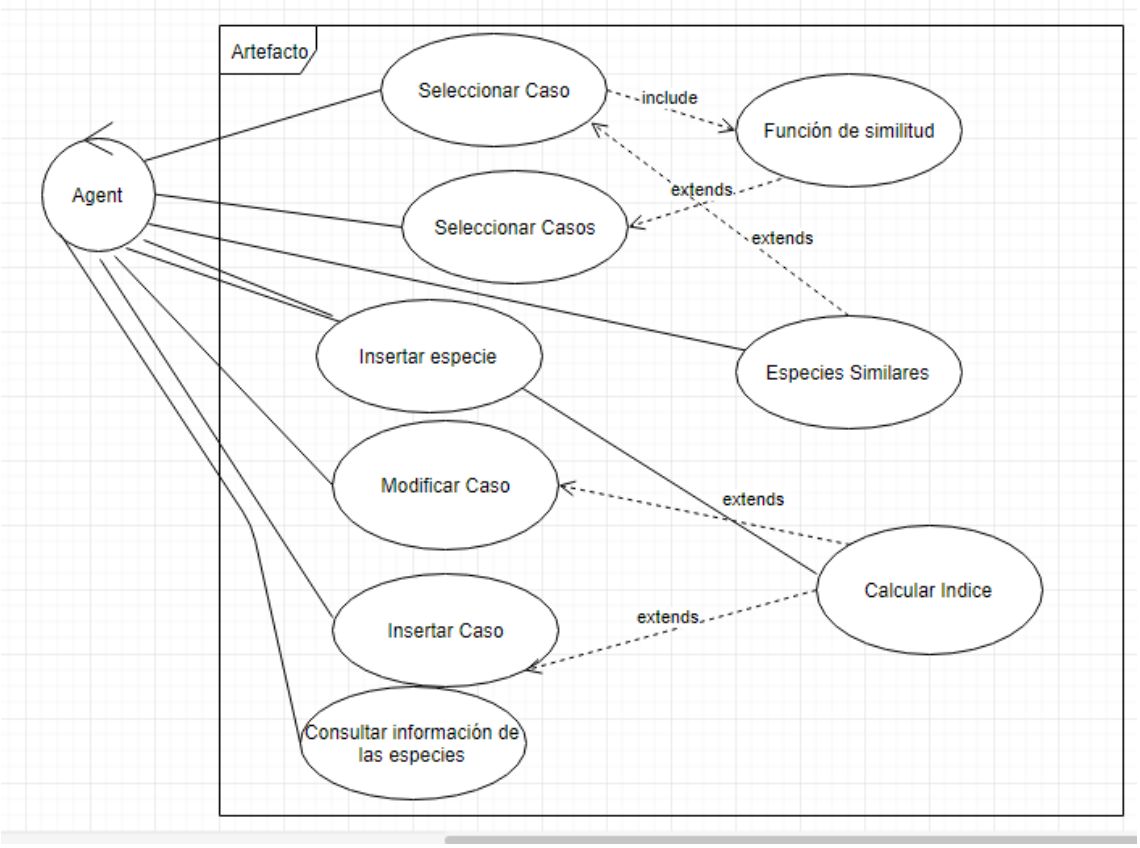


Ilustración 18 Diagrama de Casos de Uso UML para el Artefacto

Seleccionar Caso enmarca la funcionalidad de buscar en la base de datos un caso similar o igual que sea positivo y retornarlo. Si se dan las circunstancias en las que no existe ningún caso que sea similar debería de comunicarse, por otra parte, si se encuentran uno o más y son negativos, es interesante que el agente tenga conocimiento sobre todos.
Seleccionar Casos representa la funcionalidad de retornar todos los casos dentro de límites genéricos de similitud al agente, de forma que pueda discriminar los casos por los criterios que se crean convenientes. Este caso pretende dotar de mayor flexibilidad al sistema para futuros uso o necesidades.

Función de similitud proporciona la base para discriminar los casos comparando por características.
Modificar Caso permite la modificación del índice de confort de un caso
Insertar Caso permite la inserción de un caso no repetido
Calcular índice constituye la funcionalidad para evaluar la bondad entre dos casos.
Consultar la información de las especies permite el acceso a los datos referentes a las especies de plantas, de forma que estos datos solo dependen del tipo de planta.
Especies similares realiza el desempeño de la comparación sobre necesidades de las especies para poder evaluar cuán parecidas en prioridades son las especies.
Insertar especie realiza la inserción de una especie en la base de datos de especies y casos. Esta inserción debe ser una nueva planta con sus características.

Por otro lado, es necesario hablar de la estructura de los datos, que debe coincidir con el esquema lógico de la base de datos dado que es la información que prácticamente se va a trabajar. Aunque si se observa otra vez el problema, los datos son increíblemente dinámicos y puede que sea mucho más práctico recibir como entrada de cada método toda la información, dejando de lado la información de la situación, la planta y la respuesta con su acción, valor e índice de confianza. De esta forma directamente se entregan los métodos que implementen las funcionalidades presentadas con los casos de uso y se guardarían como atributos únicamente valores que rara vez se modificarán, como puede ser el margen de similitud.

Adicionalmente es interesante discutir sobre el punto de toma de decisiones entre el artefacto y el agente, por un lado, para la selección de casos es necesario sesgar por criterios objetivos basados en el fin último del cuidado de la planta y por lo tanto en el conocimiento del dominio del cuidado de las plantas. Siendo posible generar una función de similitud en el artefacto o directamente llegar a una convención entre el número de casos que es posible o rentable elevar hasta el agente para que este seleccione el caso más aproximado. Por otro lado, el cálculo del índice de confianza también puede ser generado en el agente o en el artefacto, aunque al ser un punto crítico y dada la filosofía de este proyecto con la búsqueda de flexibilidad para futuras iteraciones del proyecto, esta discusión se dejará a decisión del creador del agente, dejando abiertas ambas sendas.

De los casos de uso anteriores nacen los siguientes flujos de datos, que responden a la comunicación coordinada con el trabajo cuya finalidad es implementar el sistema de toma de decisiones. Representados en los siguientes diagramas de secuencia UML.

Ilustración 19 Diagrama de secuencia referente al caso de consultar la información de especies

En este diagrama de secuencia se observa como el manager crea la instancia del Artefacto por petición del agente, a partir de esta interacción base, que se repetirá en los siguientes diagramas de secuencia, el flujo es específica de la funcionalidad del caso de uso **Consultar la información de las especies**, dentro de esta función se dejará como optativo para la implementación el paso de todos los casos referentes a esta especie, dado que no es una funcionalidad que para este proyecto tenga interés práctico, dado que no se plantea en esta etapa el análisis de los casos para realizar otros tipos de aprendizaje.

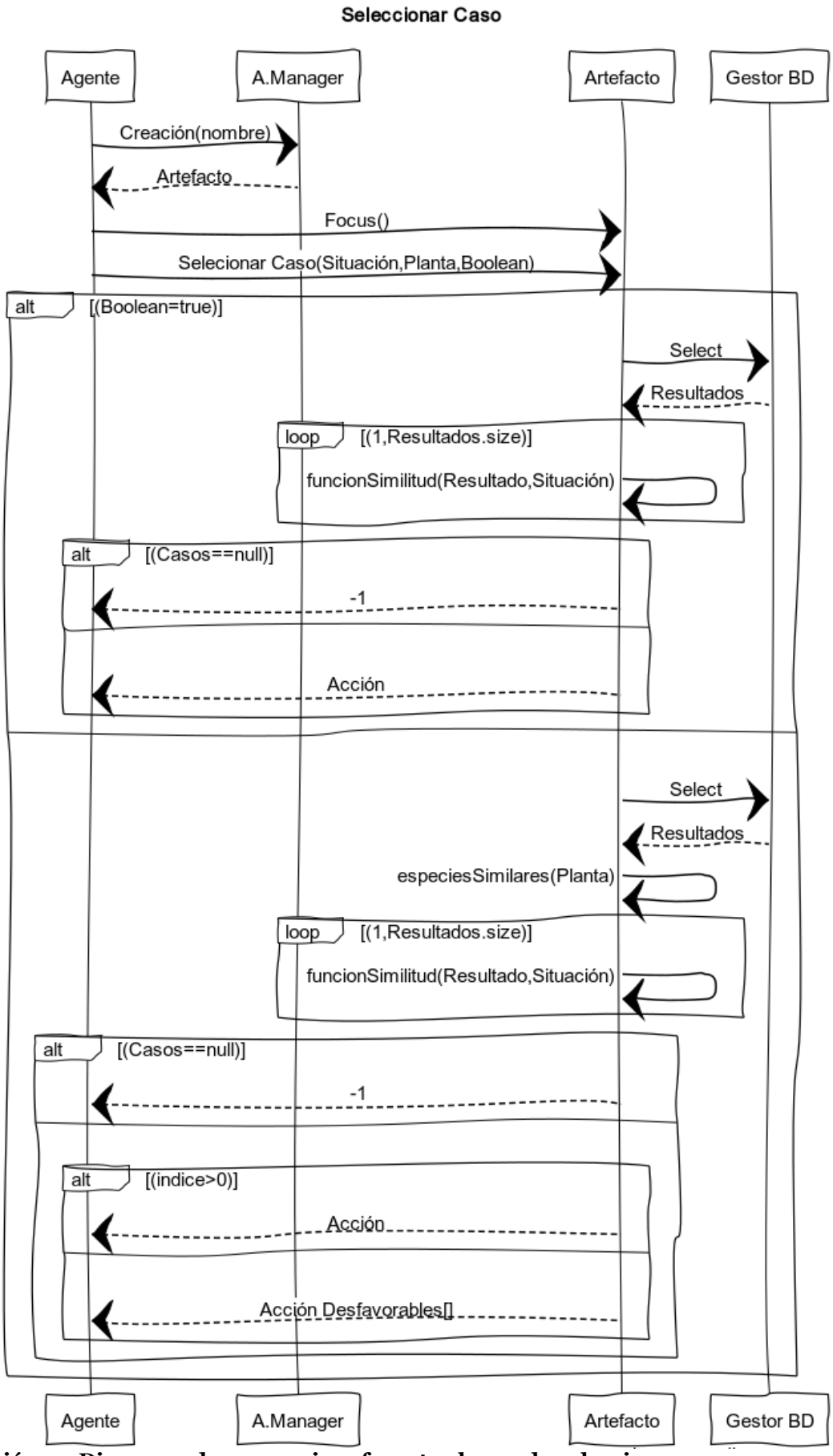


Ilustración 20 Diagrama de secuencia referente al caso de seleccionar caso

Este diagrama de secuencia responde al caso de uso de su mismo nombre. Como se puede observar, el agente transmite una Situación ; la especie de planta (asignada en su creación); y un valor (que puede ser verdadero o falso). Verdadero que indica si desea que el caso sea exactamente el mismo que el encontrado por la base de datos, o falso si se permite evaluar casos con una similitud acotada por un valor base en la creación del artefacto. Si no se encuentran casos que cumplan con los requerimientos de similitud se devuelve un *flag* de no encontrado. Si se encuentran casos, pero son desfavorables se devuelve la lista de estos casos. Si existen casos favorables que superen las restricciones de similitud se envía el mejor.

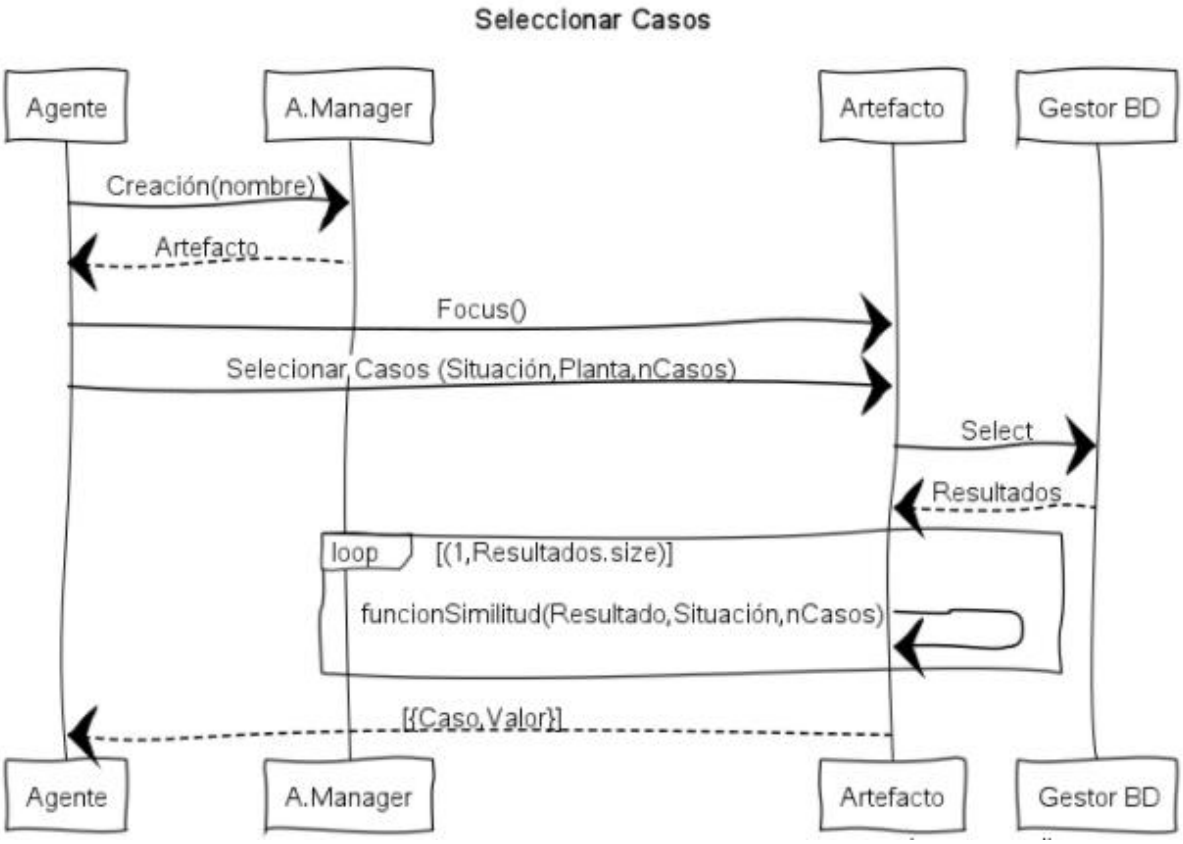


Ilustración 21 Diagrama de secuencia referente al caso de seleccionar casos

Al igual que el anterior diagrama de secuencia, éste responde a la entrega de acciones como apoyo del razonamiento basado en casos. La principal diferencia entre la funcionalidad es la misma explicada entre los casos **Seleccionar Caso** y **Seleccionar Casos**. Por otra parte, la devolución de todos los casos puede tener un gasto excesivo sobre todo en la comunicación, por lo tanto, se limitará el número de casos devueltos tomando como base a los nCasos más similares, donde nCasos será decisión del agente y en el futuro podría ser calculado según los diversos factores como por ejemplo la congestión de la red.

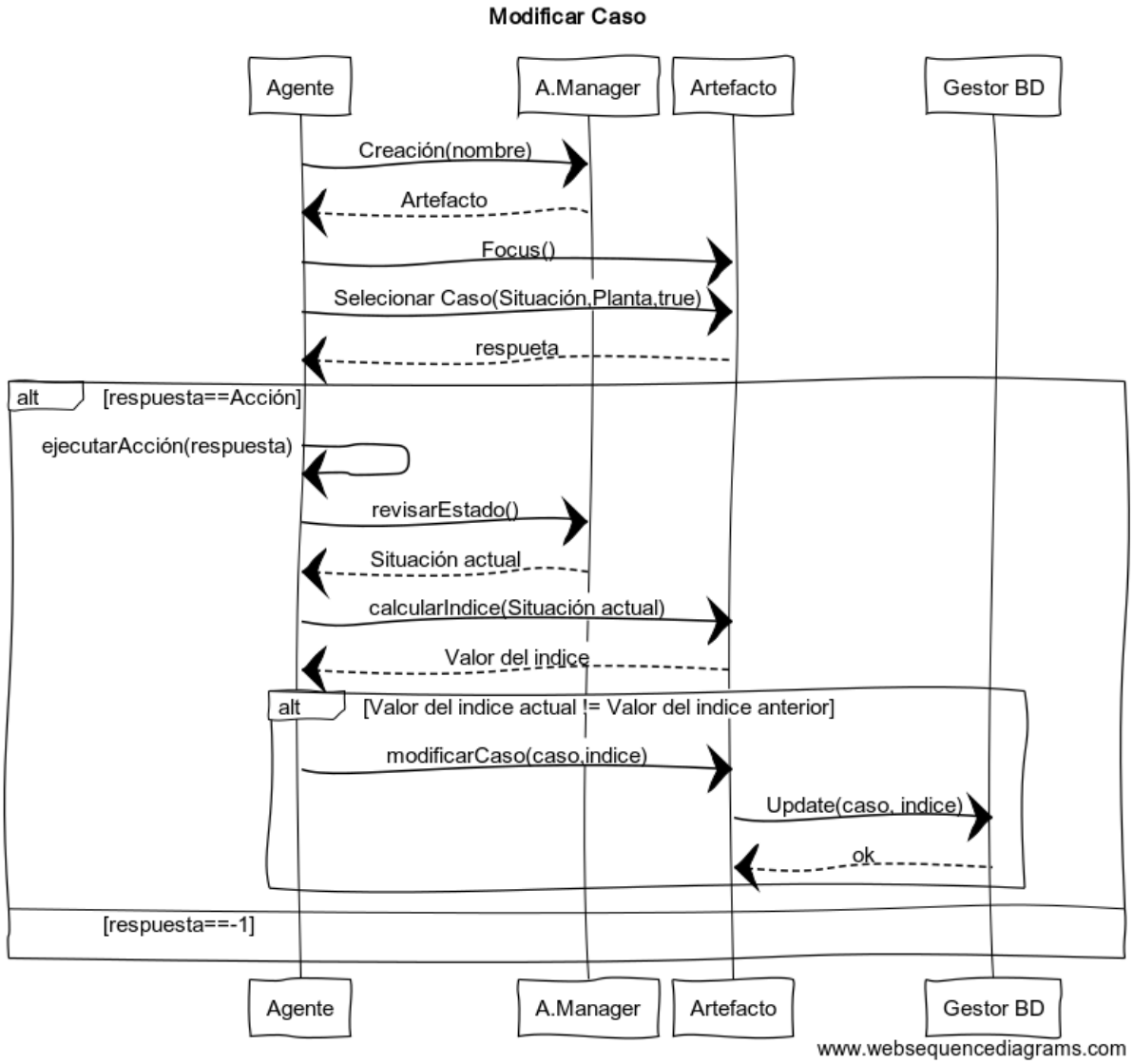


Ilustración 22 Diagrama de secuencia referente al caso de modificar caso

Cuando al usar los métodos de selección se reciba que existen casos, el agente ejecutará la acción correspondiente a la información de casos que se le devolvió. En este diagrama se ha utilizado la situación de utilizar el método de seleccionar un único caso y si acierta se llevará a cabo la acción y se calculará el índice de confort o mejora de la acción sobre el caso, si y sólo si el índice es distinto el caso recibirá la modificación del índice, siendo esta la única modificación permitida sobre la tabla de casos por el *wrapper*.

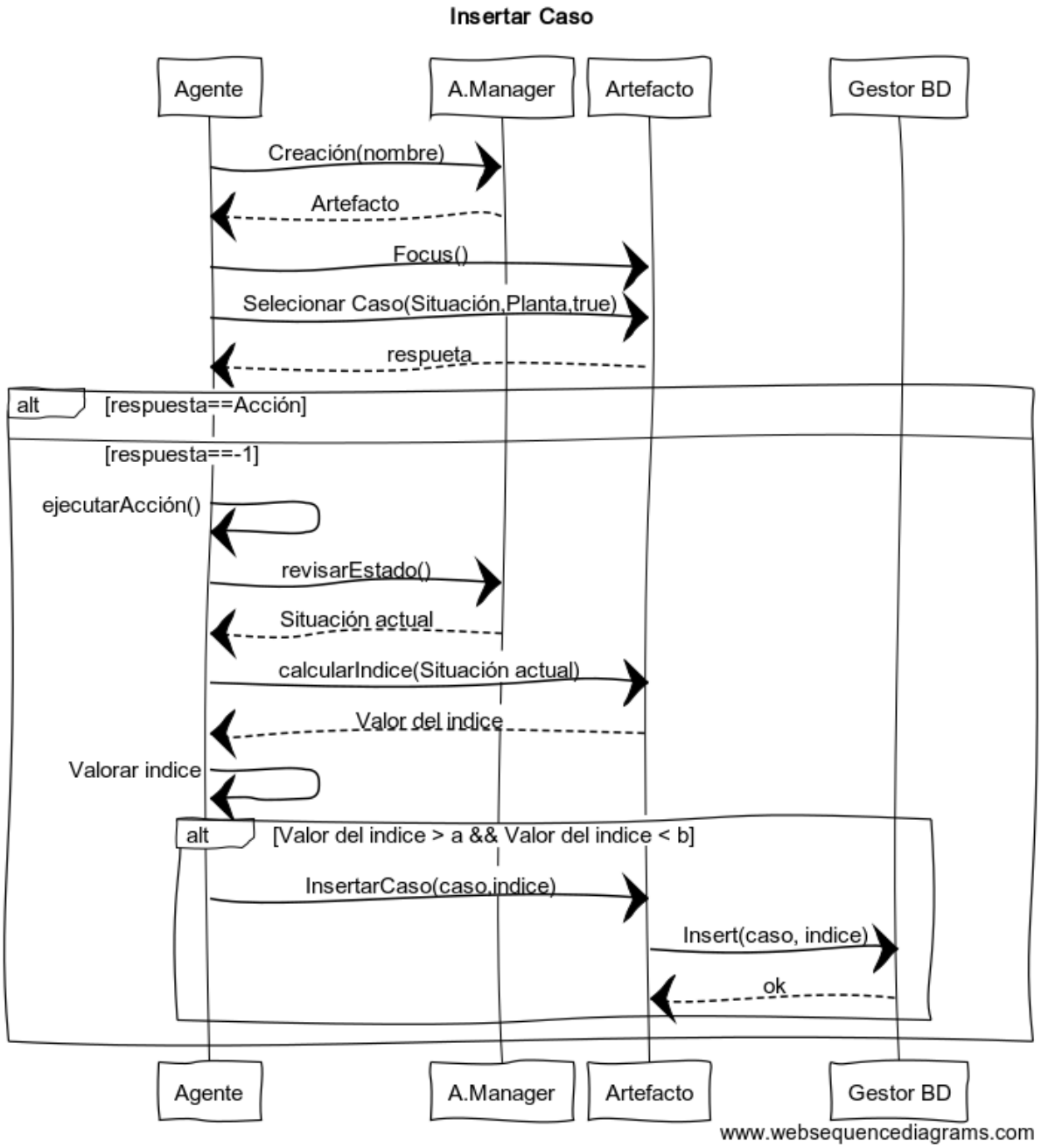
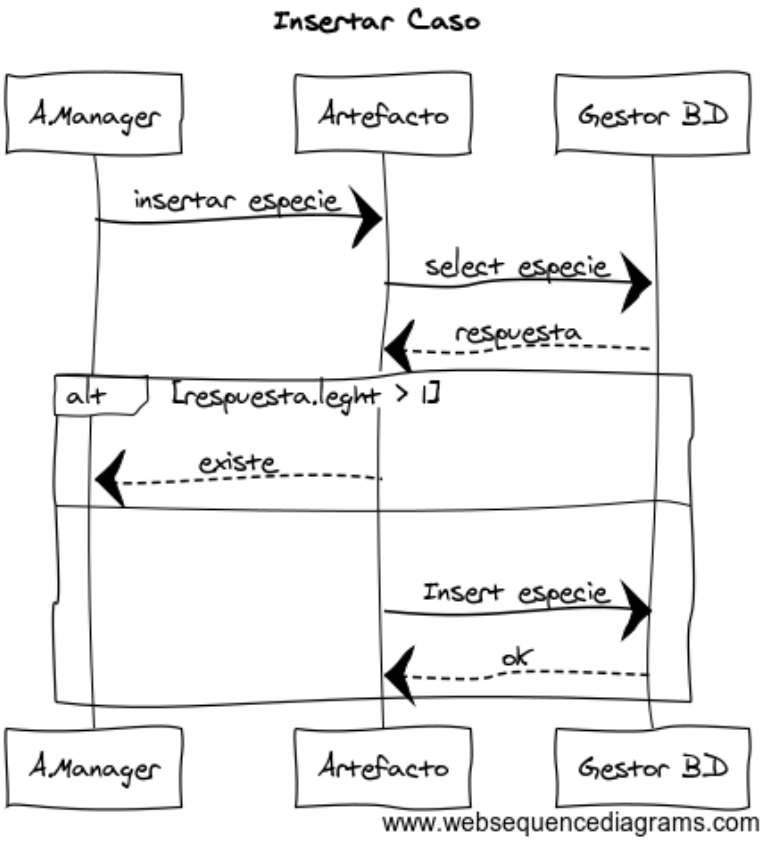


Ilustración 23 Diagrama de secuencia referente al caso de insertar caso

Prácticamente sigue el mismo esquema que el diagrama anterior, pero en este punto se ha modelado la secuencia en el caso de recibir como respuesta -1, qué es el *flag* de no hay casos. En esta circunstancia el agente realizará la acción que considere apropiada sin tener en cuenta el conocimiento previo, la acción se evaluará calculando el índice de confort igual que en la secuencia anterior, pero en este caso se valorará si es interesante guardar el caso o no, en caso de si es interesante tomando como base a los límites a y b, el caso se enviará y se insertará en la base de datos.



Por último, tenemos la secuencia del caso de uso de insertar especie. En esta funcionalidad es el manager el que utiliza el artefacto, porque la información característica de la especie y la petición de insertar la especie vienen de la interfaz de usuario en este prototipo. En este prototipo se decidió que no era necesario contemplar la modificación de la información de las especies. Aunque la base de datos si tiene en consideración la existencia de estas futuras necesidades. Dado que no hay modificación, si se envía una especie existente no se inserta, en caso contrario si.

4.6 API REST

Con la mirada puesta en el análisis de las tecnologías, el diseño de la comunicación entre la base de datos histórica y el resto del sistema será mediante REST. Se podría realizar un API que envuelva las operaciones CRUD sobre la base de datos, de forma que se abstrae el resto del sistema de la tecnología de base de datos que se utiliza, dado que si se cambiará la tecnología simplemente se implementarían las mismas funcionalidades en las mismas URIS, de forma que los cambios son transparentes para los clientes del API.

El API en este caso representa la puerta de entrada a la base de datos histórica. La realización de un punto de acceso intermedio que provee de abstracción sobre la lógica de la base de datos y las operaciones desde los clientes. También dado la opción de limitar el tipo de acceso y operaciones que se pueden realizar sobre la base de datos.

Por tanto, el API representa añadir un nuevo módulo en el sistema que aumenta el consumo de recursos y la complejidad de este. Por otra parte, reporta los beneficios de la separación por capas importante para las futuras modificaciones o cambios en la base de datos. Además de la opción de proveer únicamente de las operaciones que sean necesarias para el funcionamiento del sistema. De forma que se limita enormemente el tipo de modificaciones sobre la base de datos. Según los casos de uso del API, este debe permitir: la creación de nuevos chombos en la base de datos histórica; la inserción y consulta de los datos históricos de los sensores y actuadores de los chombos; la creación de sensores y actuadores; además de la consulta de la información de los sensores y actuadores que tiene un chombo para dotar de conocimiento de los rangos y magnitudes con los que trabajan.

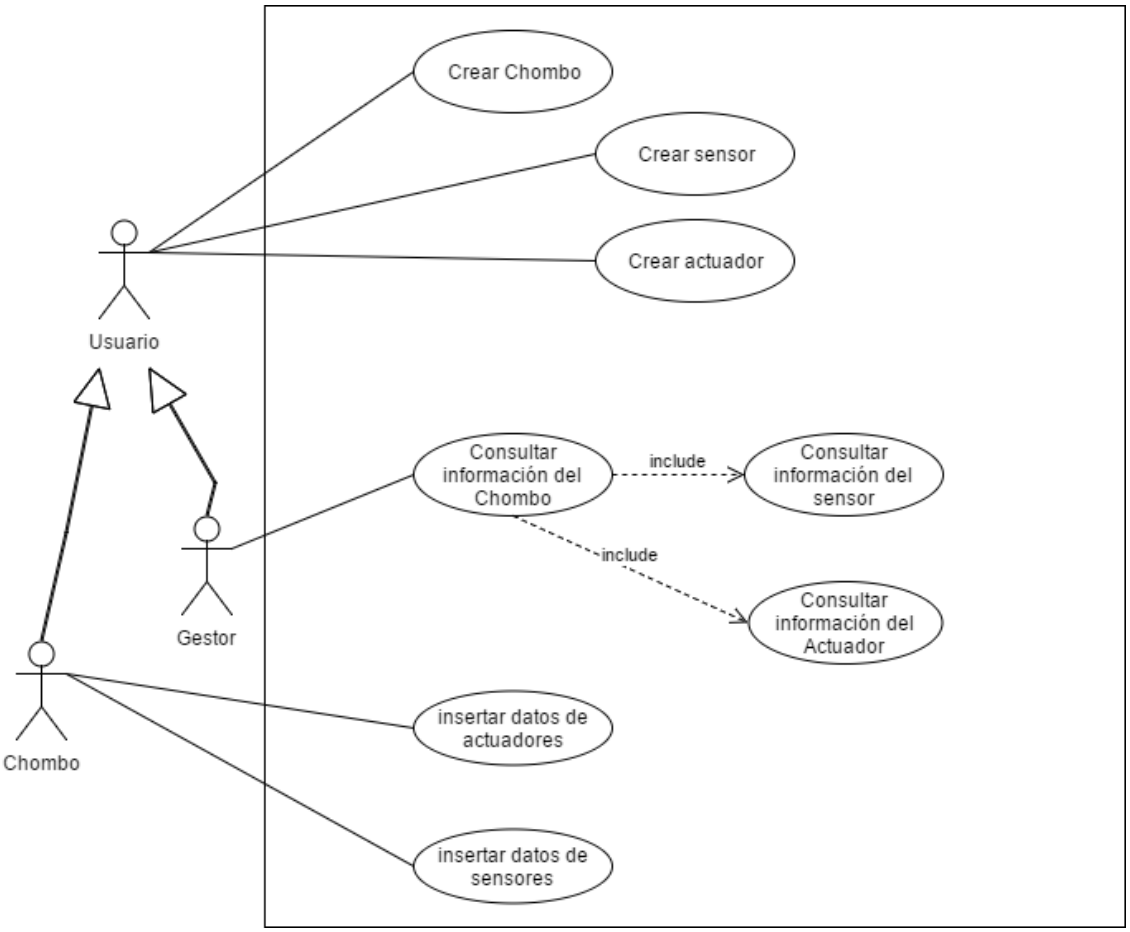


Ilustración 24 Diagrama de casos de uso del API REST

Si se toma en consideración que nos encontramos ante un sistema ubicuo. Es preciso tener en el punto de mira que se necesita pensar en un diseño donde las entradas y salidas del sistema serán mediante peticiones por algún protocolo web. Si se presta atención a los servicios o funcionalidades que debe ofrecer, podemos pensar en separar cada funcionalidad como un tipo distinto de acceso al API, de forma que cada caso de uso que necesita ser iniciado por un usuario se convierte en un punto de entrada de nuestro sistema, de forma que cada caso de uso actual o futuro se convertirá en una *URI* para la API.

4.7 Análisis del desarrollo actual

Los pasos presentados hasta el momento conforman el diseño del prototipo que se ha realizado durante este trabajo.

El diseño de la comunicación con el agente presentado durante este capítulo modela las distintas funcionalidades para realizar el apoyo al conocimiento basado en casos del sistema multiagente mediante el tipo de comunicación por operación ofrecida por CArtAgO.

En el caso de comunicación con la base de datos histórica un cambio en la tecnología no nos aportaría ningún beneficio de diseño. Aunque si puede presentarlos a nivel de eficiencia y arquitectura en el caso llegar a generar un

sistema con múltiples nodos. Nuestro sistema tiene un único nodo dado que es un primer prototipo que responde a una prueba del concepto expuesta en la motivación.

Como finalización de este análisis se han realizado los siguientes diseños de la capa de persistencia, respetando el enfoque expuesto en el punto 4.2 y persiguiendo el cumplimiento de las características del punto 3.1. De esta forma se podrán comparar las soluciones que se han desarrollado con MongoDB y las realizadas en MySQL. Aunque esto no invalida la decisión de utilizar MySQL, porque la principal ventaja de MongoDB está en la eficiencia en la distribución y el control de la consistencia de la información es menos eficiente que en MySQL.

4.7.1 BDD especies y casos con MongoDB

En primer lugar, se ha tomado el análisis de la información que se realizó en el apartado 4.3. En ese punto se generaron cuatro entidades con relaciones de muchos a uno con los casos. Dado que Mongo no tiene restricciones de clave ajena como las bases de datos relacionales, la forma de modelar este tipo de relaciones suele ser embebiendo la otra entidad como un subdocumento. En este caso las entidades de Acción y Situación nacen de proponer un diseño cercano a la tercera forma normal, pero MongoDB está orientado a almacenar datos de forma agregada en los documentos para mejorar la eficiencia de las consultas de gran cantidad de datos. Dado que MongoDB permite el guardado de estructuras como listas, también dejan de tener sentido las entidades que nacen para guardar listados como foto, sinónimos y zonas. Como resultado se ha llegado al siguiente modelo presentado en la ilustración 20.

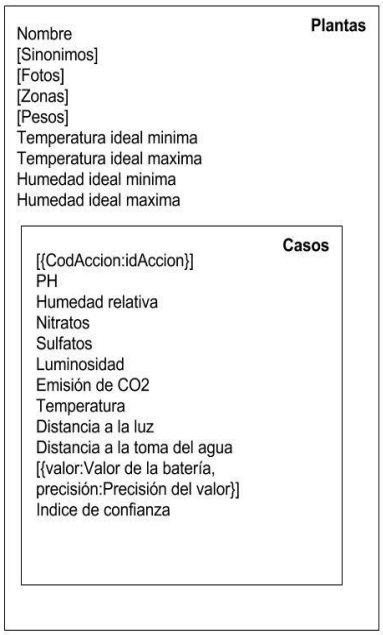


Ilustración 25 Diseño de la base de documentos para especies y casos

4.7 BDD Histórica con MongoDB

Partiendo de la metodología de diseño en MongoDB que se ha empleado en el apartado anterior se ha realizado un modelo para la parte de persistencia de los datos históricos. En el apartado 4.3 se llegó a un modelo que permite añadir nuevos esquemas de sensores y actuadores de forma flexible, pero tiene la problemática causada por la restricción de obligar a un esquema demasiado rígido. Generando la necesidad de tener diferentes entidades de actuadores y sensores para representar que difieren por el tipo de datos que producen. En este caso se solucionó elevando la responsabilidad de comprobar la restricción del tipo de datos a la capa que realice la inserción. Pero con MongoDB que no tiene restricciones de esquema porque es *schemaless* es posible abandonar las preocupaciones sobre esta problemática, porque las propiedades no están tipificadas por lo tanto podrán recibir valores del tipo que el sensor o actuador precisa.

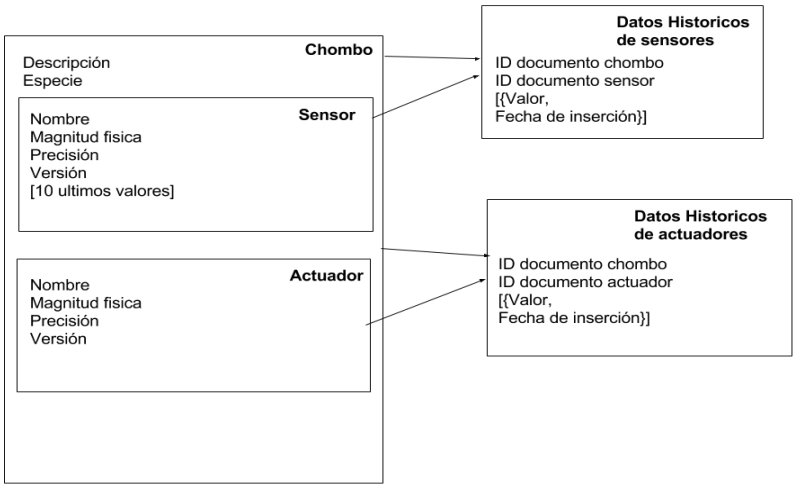


Ilustración 26 Diseño de la base de documentos para la información historica

Debido a la gran cantidad de datos que se esperan que almacene cada sensor y actuador se ha considerado que embeber estos datos, que dependen de estos elementos y el Chombo, es demasiado costoso tanto para las consultas de otros aspectos del documento como desde el punto de vista de redundancia de datos.

Por otra parte, se ha considerado utilizar un *array* que replique los últimos diez valores insertados por ese sensor, con el fin de optimizar las consultas sobre la información histórica cercana en profundidad diez.

4.8 Conclusión

En este capítulo se ha presentado todo el diseño del proyecto y las distintas decisiones tomadas durante la etapa de diseño del proyecto. En primer lugar, se han expuesto los esquemas lógicos de la capa de persistencia. Más tarde se ha mostrado el diseño del módulo de acceso a la persistencia desde el sistema de agentes. Finalmente, el diseño del API para guardar la información de los sensores y proveerla a la interfaz de usuario.
En el próximo apartado se detalla la fase de implementación del proyecto completo. Explicando cada uno de los elementos desarrollados para completar el prototipo.

5. Implementación

En este capítulo se presentarán las decisiones tomadas con respecto a cada parte diseñada en el capítulo anterior durante el proceso de su desarrollo.

5.1 Base de datos histórica

Cada entidad representada en el diagrama de clase se ha implementado como una tabla, agregando índices B+ sobre las claves primarias para optimizar las consultas sobre ellas y sobre las claves ajenas para optimizar las operaciones de *Join* en las consultas. Por otra parte, se ha considerado interesante crear un índice B+ sobre el atribute Especies de la tabla Chombo porque es un atributo sensible a ser utilizado en consultas como valor agregativo.

Por otra parte, sea decidido añadir las siguientes restricciones aparte de las claves ajenas y principales sobre los atributos:
Sensores.Max Not Hull.
Sensores.Min Not Null.
Sensores.TipoDatos Not Null.
Sensores.Precision [0,100].
Actuadores.Max Not Null.
Actuadores.Min Not Null.
Actuadores.TipoDatos Not Null.
Actuadores.Precision [0,100].
HistoricoSensores.Valor [0,100].
HistoricoActuadores.Valor [0,100].

5.2 Base de datos de Especies y Casos

En esta sección se ha seguido la misma forma de actuar que la aplicada en la base de datos histórica, creando para cada entidad representada en el diagrama de clase una tabla y agregando índices B+ sobre las claves primarias para optimizar las consultas sobre ellas y sobre las claves ajenas para optimizar las operaciones de *Join* en las consultas.

Por otra parte, se ha considerado dejar la mayor parte de los atributos sin condición de existencia, dada la gran falta de información sobre el dominio obtenida y verificada. Debido a este motivo las restricciones añadidas sobre los datos son las siguientes a excepción de las restricciones de clave primaria y clave ajena:
Planta.Especie Not Null.
Planta.HumedadMin Not Null.
Planta.TempMax Not Null.
Casos.IndiceConfianza Not Null.
Casos.Temperatura Not Null.
Casos.Luz Not Null.
Casos.PH Not Null.
Casos.Humedad Not Null.
Batery.Max Not Null.
Batery.Min Not Null.
BombaAgua.Max Not Null.

BombaAgua.Min Not Null.
Casos.BombaAgua [0,100]
Casos.Bateria [0,100]

Las dos últimas restricciones representan que los valores almacenados son porcentajes sobre los valores Max y Min de la Batería y la Bomba de agua.

5.3 Artefacto comunicación(com.) BDD

Como se comentó en el apartado de diseño del artefacto, este se encarga de calcular el índice de confianza de un caso, calcular la similitud entre casos y envolver las operaciones de la base de datos sobre las tablas de casos, especies y situaciones.

Para la realización de estas funcionalidades se extiende de la clase *Artifact* y se utiliza la etiqueta de OPERATION de CArtAgO con la finalidad de entregar acceso a los agentes a los siguientes métodos.

- Request (Situación, Código, Respuesta): En este caso Situación está envolviendo todos los parámetros (temperatura, luz…) que pasará el agente para buscar el caso similar. Código es un valor booleano que indica el nivel de similitud del caso que se busca, el valor true representa una búsqueda exacta y el valor falso una búsqueda con el nivel de similitud que tenga el artefacto. Respuesta es del tipo OpFeedbackParam<int []> y nos permite devolver la acción mediante su método set ().
    - Este método en caso de buscar por similitud (similitud de especies y valores de la situación por pesos de las especies) en la base de datos y devuelve la acción positiva mayor o (en caso de no existir) el conjunto de acciones negativas que superen el umbral de similitud
- Insert (Situación, Acción, Índice, Respuesta): Al igual que antes Situación envuelve los parámetros y Respuesta es del tipo OpFeedbackParam, devolverá un *flag* de acierto o error de la operación. Las novedades son la Acción que es el código de acción e Índice que representa el índice de confort del caso. La funcionalidad es la inserción del caso pasado en la tabla de casos y situación.
- llistaCasosSim (Situación, Código, Respuesta): Sigue el mismo esquema que Request en parámetros, salvo Respuesta que devolverá Object [] debido a que los casos tienen diferentes tipos entre sus atributos.
    - La funcionalidad difiere con Request, en que se devolverán todos los casos más cercanos por similitud de valor de la situación sin considerar casos de otras especies.
- Update (Caso, Índice, Respuesta): Donde Caso representa la conjunción de la información de Situación, Planta y Acción. Índice representa el nuevo índice de confort del caso. Por último, Respuesta es del tipo OpFeedbackParam, devolverá un *flag* de acierto o error de la operación. La funcionalidad que implementa es la actualización de la valoración de un caso si este al ser reevaluado ha obtenido un índice diferente.

- DatosEspecie (Planta): Devuelve los datos característicos de una especie (temperatura ideal, humedad ideal y pesos).
- InsertEspecie(Planta): Inserta una planta en la base de datos si no existe, si existe devuelve un false.
- CalcularIndice (Planta, Situación anterior, Situación actual): devuelve el índice de confort de la acción, obtenido como resultado de la siguiente formula:

$$\sum_{y} \alpha_y * ((V_{y\ ideal} - S_{y\ actual}) - (V_{y\ ideal} - S_{y\ anterior}))$$

Donde y es el parámetro de la Situación, V es el valor ideal del parámetro para la especie y alfa es el peso del parámetro para esa especie.

5.4 API REST

Esta es la parte principal de las comunicaciones del sistema dado que enlaza la capa de persistencia con los clientes que consultan la información y los clientes que la generan, siendo ambos lados clientes del API dado que los primeros consultan y los segundos insertan dicha información

Dado que finalmente se ha decidido utilizar el servidor de tipo REST, se utilizarán las *URLS* como puertas de entrada a la invocación de las diferentes funcionalidades explicadas en el diseño. Pero se ha decidido realizar la comunicación mediante métodos GET de forma que el paso de valores sea mediante parámetros, se podría haber utilizado el método POST para algunas funcionalidades, pero al ser un volumen corto de información el que se desea pasar se ha simplificado el acceso.

Debido al uso de JavaScript como lenguaje para el servidor se ha decidido realizar el retorno de datos en formato JSON, debido a que es el sistema nativo de notación de este lenguaje, aunque sólo se utilizará si la respuesta es una colección de datos como puede ser el resultado de la consulta de la información histórica, en caso contrario se usará el texto plano, por ejemplo, si la respuesta es un dato simple como puede ser el índice de un sensor que acaba de ser creado por una inserción.

La base del servidor REST será Express sobre Nodejs, de forma que será necesaria la instalación de los módulos dependientes de Express. Además, hay que tener en cuenta que se realizará desde el API el acceso al gestor de la base de datos que en este caso será MySQL, por lo tanto, es necesaria la instalación del driver. Para la simplificación de la replicación del sistema en otro entorno se ha encapsulado la instalación de las dependencias en el siguiente Packaje.json

Como se dijo en la fase de diseño cada caso de uso se ha convertido en una entrada al API, de forma que en este punto se va a repasar los casos de uso centrándonos en la implementación de las funcionalidades

Partiendo del caso de uso de creación de chombo Crear Chombo, referenciado en el diseño del API REST, en el apartado de Diseño, en el diagrama de casos que genera el requisito de inserción de las entidades Chombo en la base de datos histórica desde la interfaz del usuario final del sistema global. Con base a este caso de uso se ha

implementado la siguiente secuencia de acceso y se ha especificado el tipo de respuestas ofrecidas, la operación que envuelve y los parámetros que requiere.

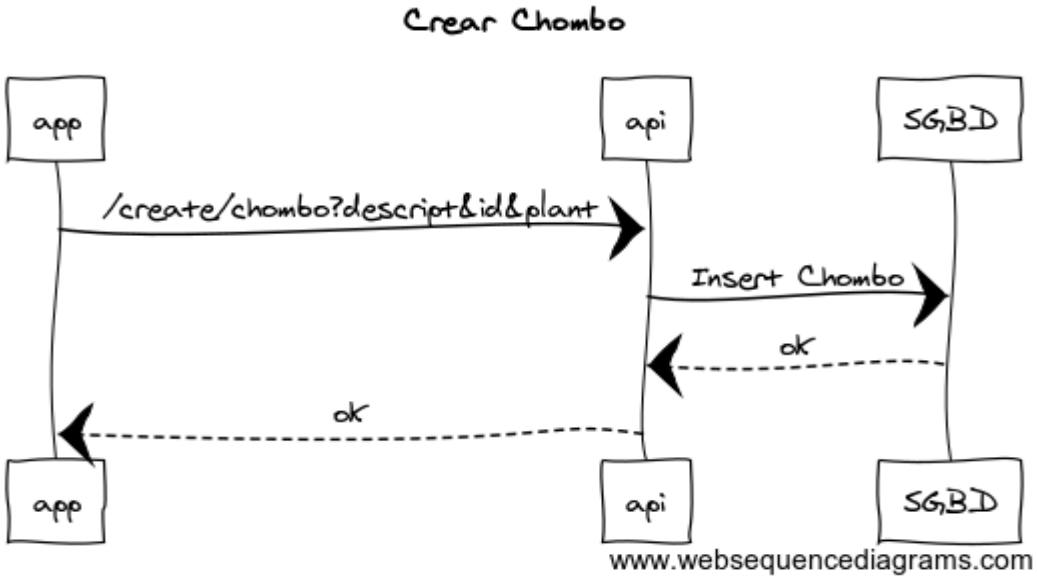


Ilustración 27 Diagrama de secuencia Crear Chombo

Caso de uso	Crear Chombo	
URL	/create/chombo	
Parámetros	descript:	cadena de caracteres que especifique información de la gama del chombo o variación
	id:	Identificador de serie de la maceta-robot, se convertirá en el código del Chombo para el sistema
	plant:	Codificación de la especie de la planta extraída de la base de datos de especies y casos.
Respuesta en caso de acierto	200	
Respuesta en caso de error	500, y se escribirá el error por pantalla, si se utiliza nohup el archivo .out actuará como log del sistema	
Operación SQL	`'Insert into Chombo (Descripcion,idChombo,Especie) VALUES ('+"'"+req.query.descript+"','"+"'"+req.query.id+"','"+"'"+req.query.plant+"'"+')'`	

Tabla 5 URL Crear Chombo

Los siguientes casos de uso implementados son la creación de un sensor y actuador en la base de datos histórica, funcionalidad requerida tanto por la interfaz de usuario como por el Chombo.

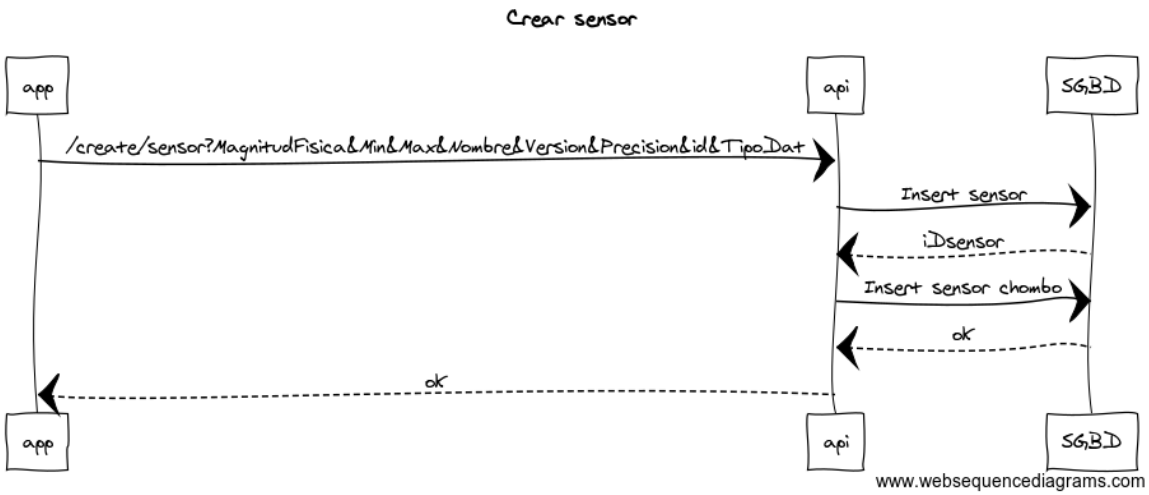


Ilustración 28 Diagrama de secuencia Crear sensor

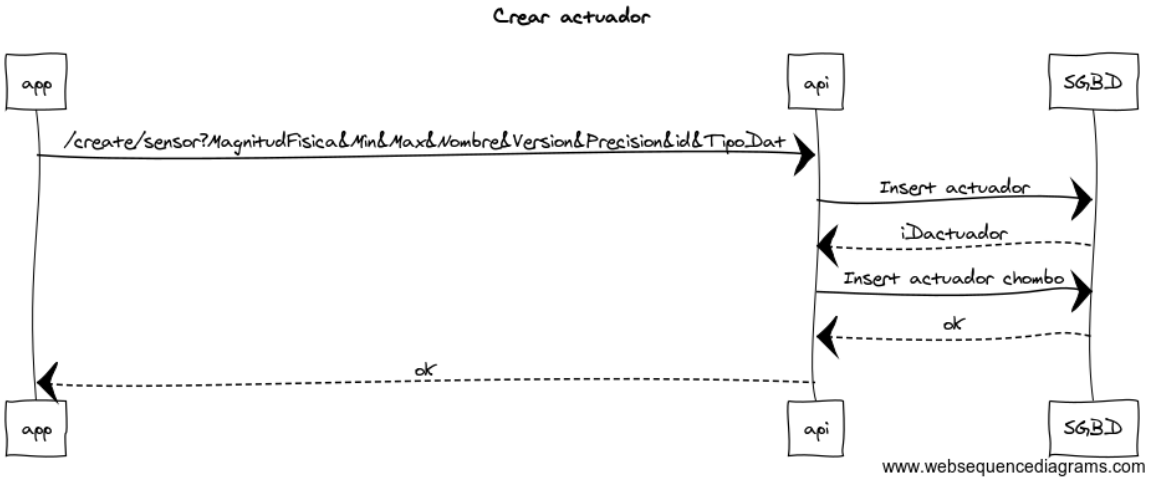


Ilustración 29 Diagrama de secuencia Crear Actuador

Caso de uso	Crear Sensor Crear Actuador	
URL	/create/sensor /create/actuador	
Parámetros	MagnitudFisica:	Magnitud física en la que representa la información el sensor/actuador
	Min:	Valor mínimo que tiene en su rango de representación el sensor/actuador
	Nombre:	Nombre del sensor/actuador
	Version:	Versión del sensor/actuador
	Precision:	Precisión en la digitalización de la información real que está midiendo.

	TipoDato:	Tipificación del valor en el que deberían convertirse el Min y Max al consultarse
	Id:	Representa el id de un Chombo
	Max:	Valor máximo que tiene en su rango de representación el sensor/actuador
Respuesta en caso de acierto	Integer, representa el id del sensor o actuador que se deseaba crear	
Respuesta en caso de error	500	

Operación SQL	`'Insert into Sensores SET ?'  'Insert into Actuadores SET ?'`
	`Donde ? será sustituido por el siguiente JSON rellenado con los parametros del metodo GET`
	`{`
	`MagnitudFisica:request.query.MagnitudFisica,`
	`Min:request.query.Min,`
	`Max:request.query.Max,`
	`Nombre:request.query.Nombre,`
	`Version:request.query.Version,`
	`Precision:request.query.Precision,`
	`TipoDato:request.query.TipoDato`
	`}`
	
	Solo se insertará en el caso de que no exista ya este sensor	

Tabla 6 URLs Crear Sensor/Actuador

Los siguientes casos de uso implementados representan el acceso a la consulta de los sensores y actuadores, funcionalidad requerida por la interfaz de usuario para facilitar la selección de estos.

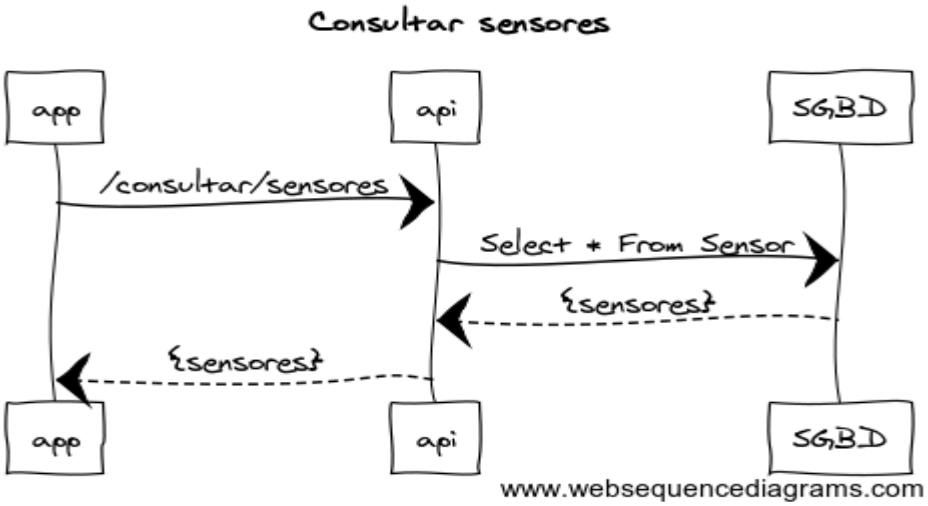


Ilustración 30 Diagrama de secuencia Consultar sensores

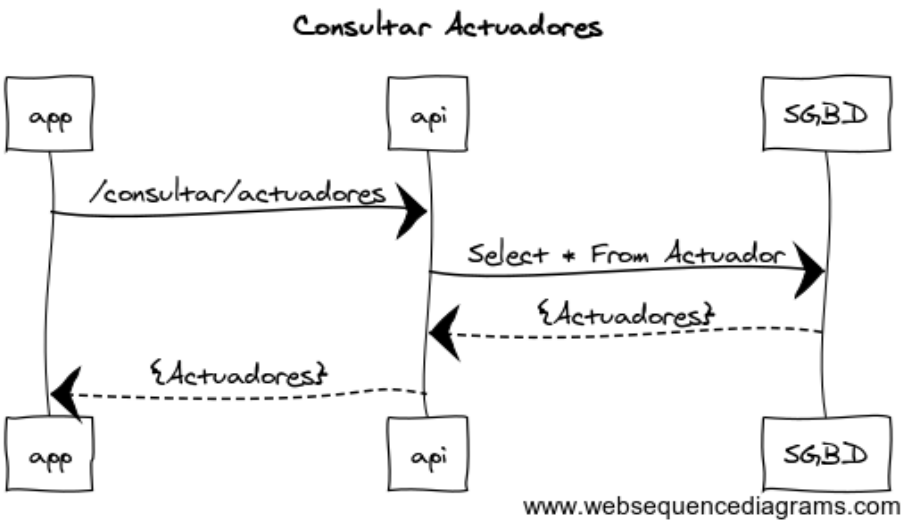


Ilustración 31 Diagrama de secuencia Consultar actuadores

Caso de uso	Consultar Sensores Consultar Actuadores
URL	/consultar/actuadores /consultar/sensores
Parámetros	No hay parámetros
Respuesta en caso de acierto	JSON, representa la colección de Actuadores o Sensores almacenada en la base de datos histórica, donde cada objeto JSON es una fila o instancia de la entidad y los parámetros de este representan las columnas o atributos de la entidad.
Respuesta en caso de error	500
Operación SQL	`Select * From Actuadores` `Select * From Sensores`

Tabla 7 URLS Consultar sensor/actuador

Los próximos casos de uso implementados son la consulta de las informaciones obtenidas por los sensores y actuadores con base al tiempo. Esta funcionalidad es requerida por la interfaz de usuario para la representación gráfica de la evolución de los chombos, además de poder ser utilizada para futuros análisis por otras herramientas.

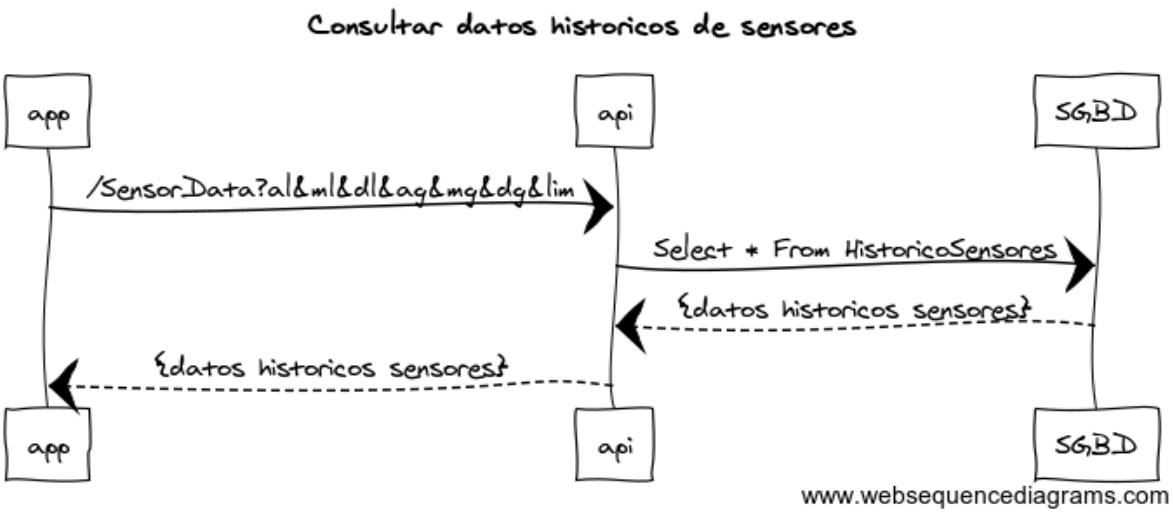


Ilustración 32 Diagrama de secuencia de consultar datos históricos de sensores

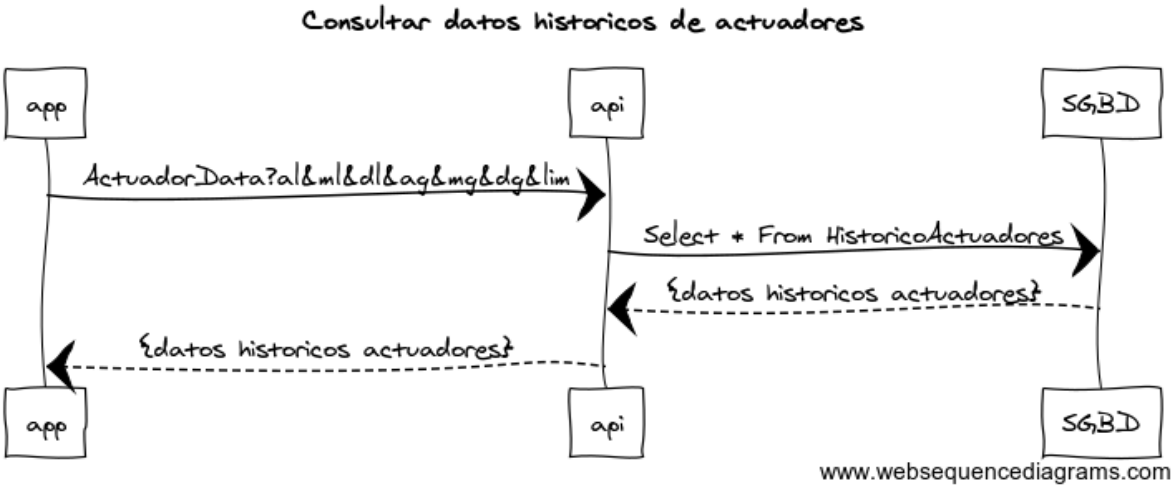


Ilustración 33 Diagrama de secuencia de consultar datos históricos de actuadores

Caso de uso	consulta de las informaciones obtenidas por los sensores y actuadores	
URL	/ActuadorData /SensorData	
Parámetros	al-ml-dl:	Representa la fecha mínima de los valores que se devolverán. Donde al es el año, ml es el mes y dl es el día
	ag-mg-dg:	Representa la fecha máxima de los valores que se devolverán Donde ag es el año, mg es el mes y dg es el día
	lim	Representa el número máximo de objetos JSON devueltos por la consulta

Respuesta en caso de acierto	JSON, representa la colección de Actuadores o Sensores almacenada en la base de datos histórica, donde cada objeto JSON es una fila o instancia de la entidad y los parámetros de este representan las columnas o atributos de la entidad.
Respuesta en caso de error	500

Operación SQL	`'Select * from HistoricoSensores where Fecha BETWEEN' +"'"+req.query.al+'-'+req.query.ml+'-'+req.query.dl+' 00:00:00'+"'"+' AND'+"'"+req.query.ag+'-'+req.query.mg+'-'+req.query.dg+' 00:00:00'+"'"+' order by Fecha DESC LIMIT '+req.query.lim`
	
	`'Select * from HistoricoActuadores where Fecha BETWEEN' +"'"+req.query.al+'-'+req.query.ml+'-'+req.query.dl+' 00:00:00'+"'"+' AND'+"'"+req.query.ag+'-'+req.query.mg+'-'+req.query.dg+' 00:00:00'+"'"+' order by Fecha DESC LIMIT '+req.query.lim`

Tabla 8 URLS consulta de información histórica de sensores y actuadores

Otro de los casos de uso implementado ha sido la consulta de la información de un Chombo específico con la información relativa a sus actuadores y sensores.

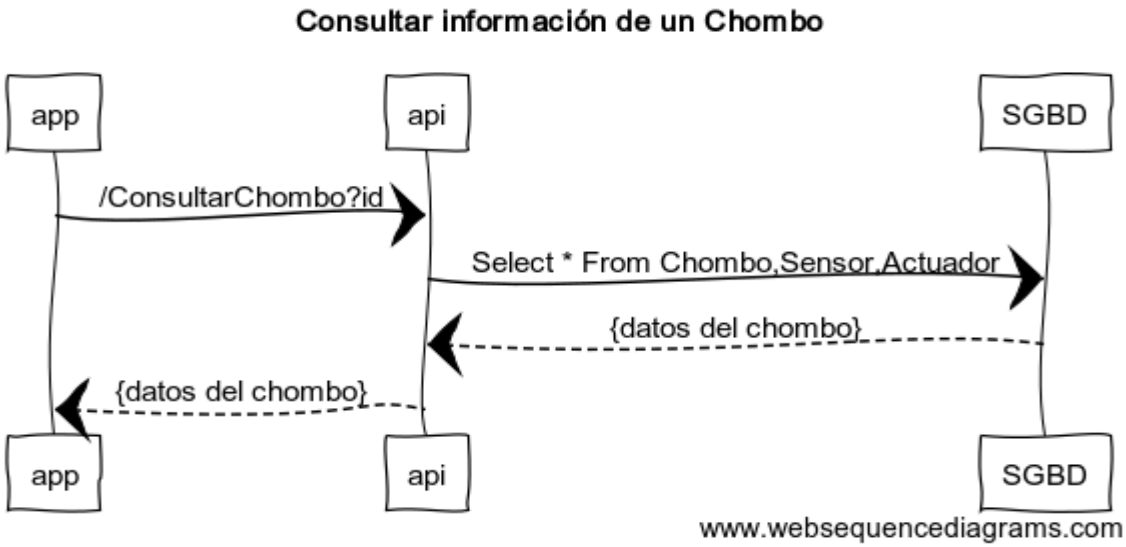


Ilustración 34 Diagrama de secuencia de consulta de la información de un Chombo

Caso de uso	Consultar información de un Chombo	
URL	/ConsultarChombo	
Parámetros	id:	Identificador del Chombo
Respuesta en caso de acierto	JSON, representa la colección de almacenada en la base de datos histórica, donde cada objeto JSON es una fila o instancia de la entidad y los parámetros de este representan las columnas o atributos de la entidad.	
Respuesta en caso de error	500	
Operación SQL	`'Select * from Chombo c, Sensores_has_Chombo sc, Sensores s, Actuadores_has_Chombo ac, Actuadores a  where a.iDactuador=ac.iDactuador and ac.iDchombo=c.iDchombo and c.iDchombo=sc.iDchombo and sc.iDsensors=s.iDsensors and iDChombo ='+req.query.id`	

Tabla 9 URL Consultar información de un Chombo

Por último, se han implementado los casos de uso de inserción de los datos pertenecientes a los sensores y actuadores desde los Chombos. En este punto hay que observar que el sistema está almacenando información que depende de la dimensión temporal. Por lo tanto, la cardinalidad de esta tabla en la base de datos crecerá con el tiempo dependiendo únicamente de los intervalos de consulta de los sensores. Por este motivo se limitará la inserción de la información a la variación de la nueva información con respecto a los dos últimos valores insertados. De esta forma se pretende poder guardar tendencias y datos anómalos.

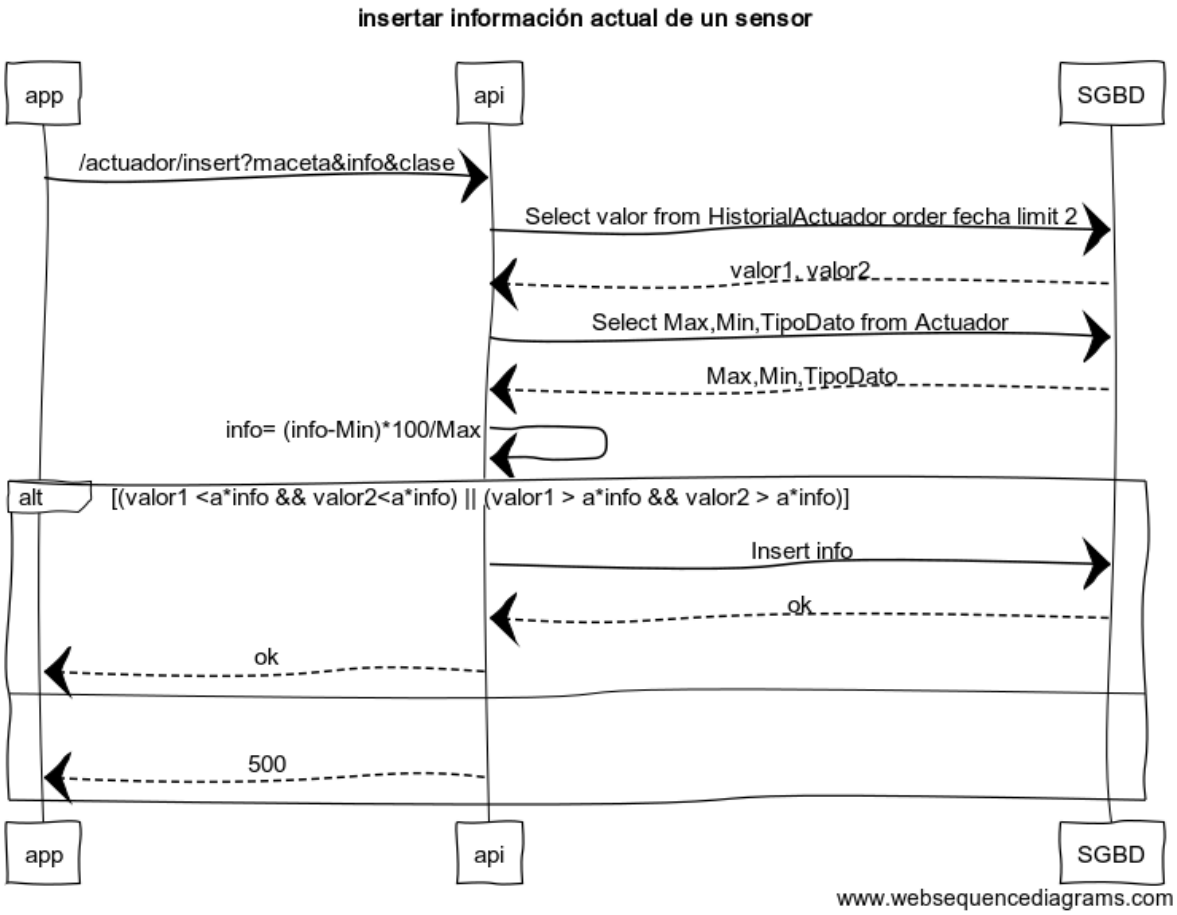


Ilustración 35 Diagrama de secuencia de insertar información actual de un sensor

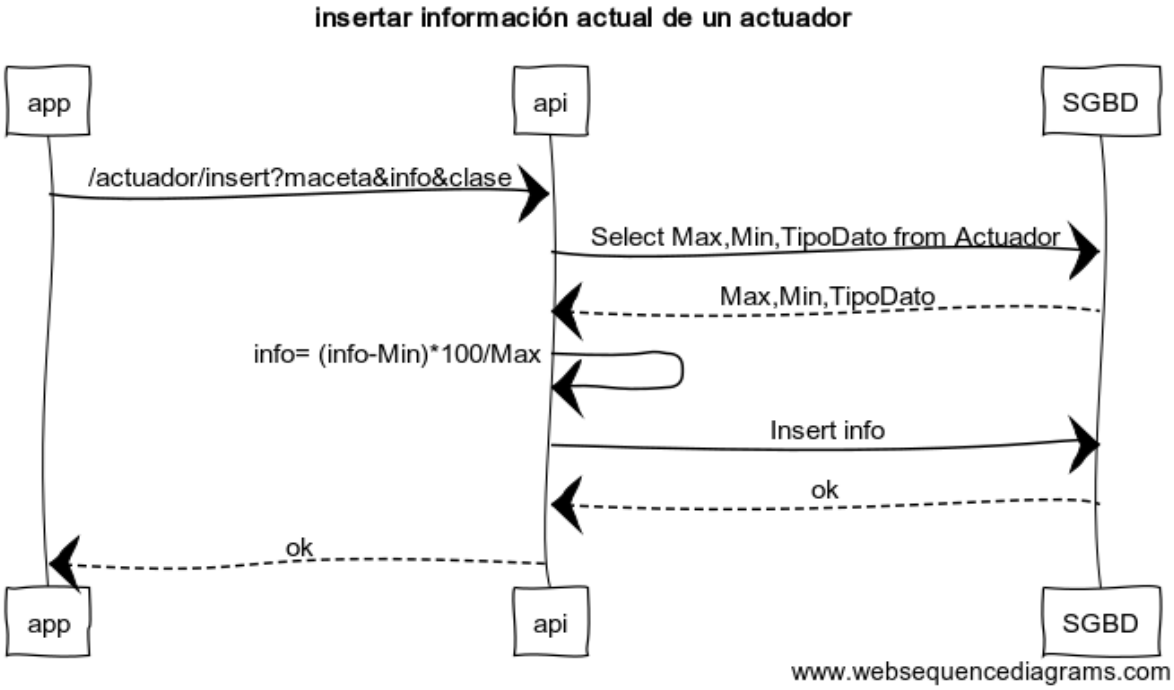


Ilustración 36 Diagrama de secuencia de insertar información actual de un actuador

Caso de uso	Insertar la información de sensores y actuadores	
URL	/sensor/insert /actuador/insert	
Parametros	info:	Valor del sensor
	clase:	Identificador del Sensor
	maceta:	Identificador del Chombo
Respuesta en caso de acierto	200, si no es información redundante o en cualquier caso para los actuadores. 200 Información redundante si es información redundante para un sensor, en este caso se considera redundante si la diferencia a los dos datos anteriores es inferior a 0.5.	
Respuesta en caso de error	500	

Operación SQL	`Select * from HistoricoSensores where iDchombo='+req.query.maceta+ '+ iDsensors =' +req.query.clase+' order by Fecha DESC LIMIT'+2`
	
	`Insert into HistoricoSensores (Fecha,Valor,iDchombo,iDsensors) VALUES (NOW(),'+"'"+req.query.info+"'"+',+"'"+req.query.maceta+"'"+', '+"'"+req.query.clase+"'"+')`
	
	`Insert into HistoricoActuadores (Fecha,Valor,idChombo,iDactuador) VALUES ( NOW(),'+"'"+req.query.info+"'"+',' +"'"+req.query.maceta+"'"+',+"'"+req.query.clase+"'"+')`	

Tabla 10 URLs Insertar la información de sensores/actuadores

5.5 Implantación

Para facilitar la instalación de una réplica del entorno de trabajo donde se desarrolló el proyecto, se ha decidido utilizar Docker como base para la automatización de la generación de este mediante un Dockerfile.

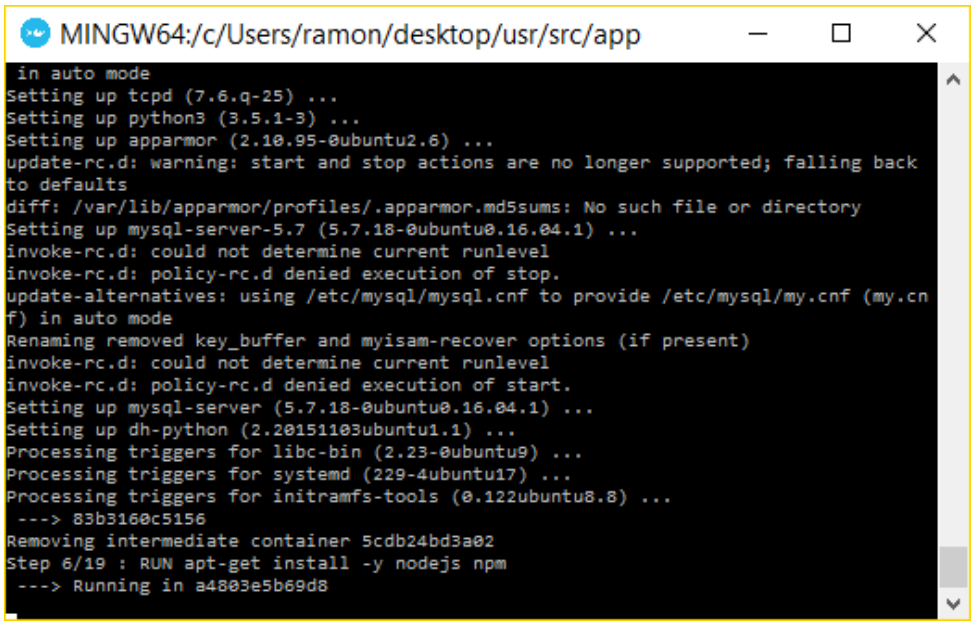


Ilustración 37 Ejecución del Dockerfile

Mediante el uso de este Dockerfile se realiza la instalación del servidor MySQL, Java, Node.js y el instalador de paquetes NPM.
Se crea un directorio de trabajo donde se realiza la instalación del entorno y la copia de los siguientes ficheros:

- El packaje.json, fichero que recoge las dependencias de paquetes del API

  REST

- El código del API REST

- Ficheros SQL que recogen la creación de los esquemas de las bases de datos e inserción de los datos base de especies

El Dockerfile se ha implementado para que busque en el directorio donde se ejecute Docker estos archivos y realice automáticamente la instalación de las dependencias de paquetes del código REST, iniciar el servidor de gestión de base de datos de MySQL, la creación de las bases de datos y llenado con la información base, ejecución del API REST, y abrir el acceso al servicio de la base de datos, para permitir la comunicación con el artefacto, y el puerto 80 para el acceso a la interfaz REST del API.

El uso de Docker ha sido una decisión basada por una parte en la portabilidad para la replicación del entorno de desarrollo o trabajo de este proyecto. Es

verdad que existen otras tecnologías que permiten la virtualización de máquinas y nos ofrecen portabilidad, como puede ser Virtualbox, pero Docker tiene como ventaja sobre estos competidores el ahorro de memoria al generar el entorno, porque utiliza *containers* [11] y no máquinas virtuales. Esta ventaja es el otro motivo por el que se ha tomado la decisión de usar Docker.

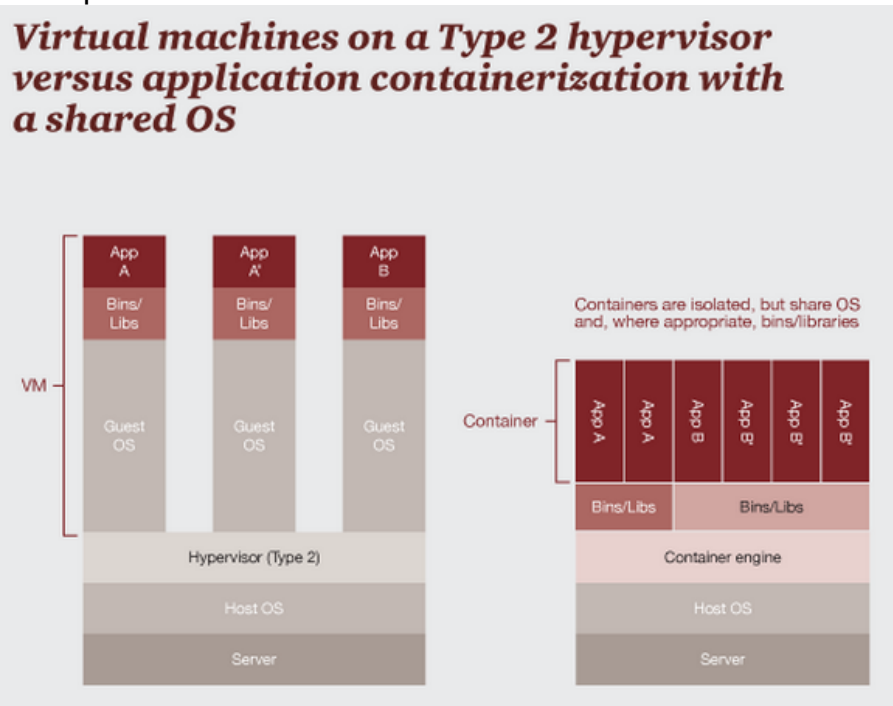


Ilustración 38 Comparativa entre VM y Container (extraída de http://www.bogotobogo.com/DevOps/Docker/Docker_Container_vs_Virtual_Machine.php)

5.6 Pruebas

En este apartado se presentarán los resultados de las pruebas realizadas en este proyecto. Además, se mostrará el siguiente extracto de las pruebas más significativas.

1. Prueba de inserción de datos del dominio del problema en la base de datos de especies y casos.
2. Pruebas de comunicación con el agente.
    a. Prueba de inicialización del artefacto.
    b. Prueba de selección de un caso y retorno de este.
3. Pruebas de acceso al API.
    a. Prueba de crear Chombo.
    b. Prueba de crear Actuador.
    c. Prueba de insertar datos de sensores.
    d. Prueba de consultar información histórica de sensores.
    e. Prueba de inserción de redundancia.

5.6.1 Prueba de inserción de datos del dominio del problema en la base de datos de especies y casos

En primer lugar, se realizó un *web scrapper* para la inserción de información base de las especies. Este *web scrapper* se hizo a partir de la tecnología de Selenium[26] para atacar un repositorio web con datos de plantas. De este repositorio se recogió el nombre, las imágenes, la zona según el sistema de Hardiness zone[27] y si tiene o no flores. A partir de la zona se obtiene la temperatura mínima y desde la información de si tiene flores o no se obtiene la humedad relativa ideal. De esta forma se completa la característica CA2 con esta información de una fuente externa.

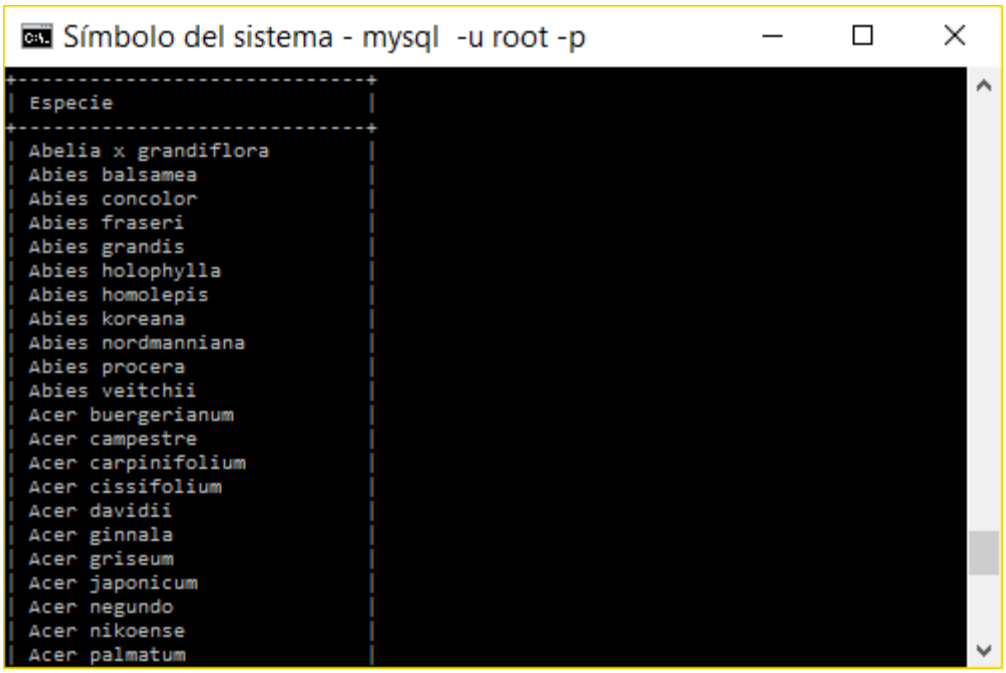


Ilustración 39 Tabla de plantas

5.6.2 Pruebas de comunicación con el agente.

5.6.2.a Prueba de inicialización del artefacto

Por otro lado, la comunicación con los agentes se ha testeado juntamente con el sistema multiagente. En la siguiente ilustración se puede ver como el sistema multiagente inicializa el artefacto de comunicación con la base de datos presentado en este trabajo.

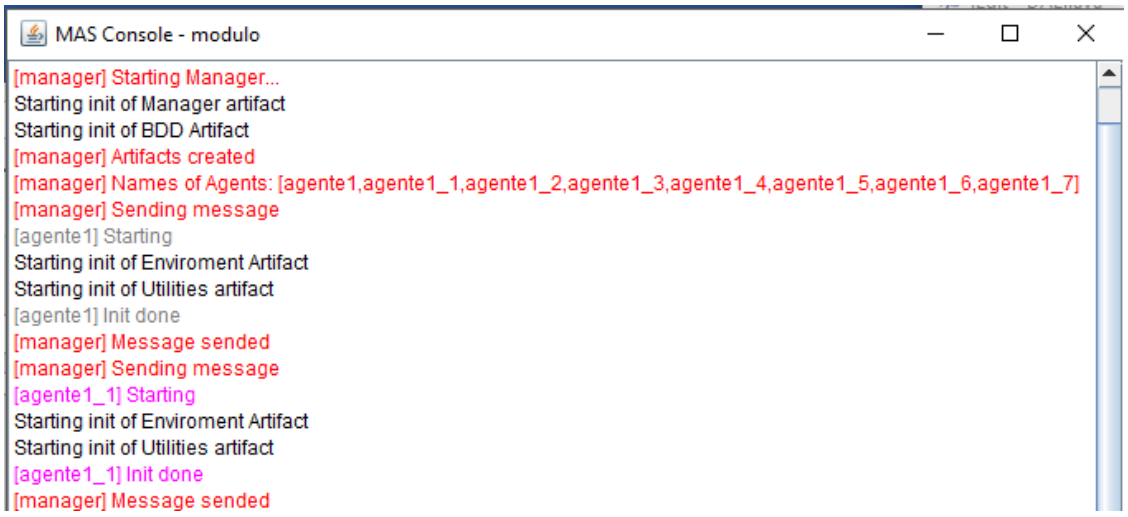


Ilustración 40 Inicialización del sistema multiagente

5.6.2.b Prueba de selección de un caso y retorno de este

Para comprobar la comunicación con la base de datos, se creó un agente temporal adicional para comprobar la conexión. Se utilizó un agente que no pertenece al sistema general, hacía una petición Request al Artefacto que se comunica con la base de datos. En esta petición se pasan los datos de una situación, un código de valor *false* y la codificación de una planta. Al enviarse *false* se realiza la búsqueda de casos similares. El resultado fue el envió de la acción. Esta primera aproximación tenía la estructura mostrada en la siguiente ilustración.

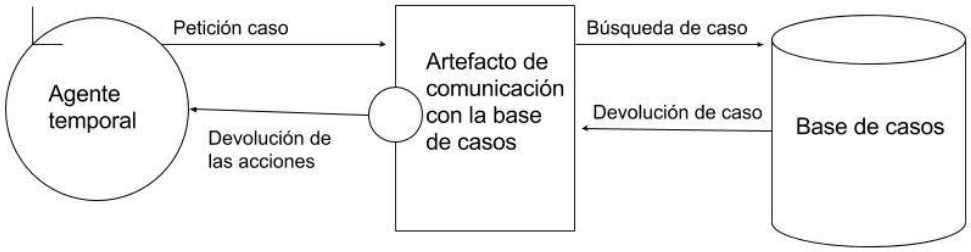


Ilustración 41 Arquitectura de prueba del Artefacto

5.6.3 Pruebas de acceso al API.

Para realizar las pruebas sobre el API se ha decidido utilizar un cliente REST (Postman). El motivo de esta decisión es que los componentes que acceden a esta API no están implementados.

5.6.3.a Prueba de crear Chombo

Para comprobar la funcionalidad de creación de chombos se han generado diversas peticiones a la URL especificada en el punto 5.4. Entre estas peticiones se ha enviado la información de gama como descripción, un uno como id y la planta codificada como 9. Ha esta petición como se puede ver en la ilustración 36, se ha contestado con un 200 representado por el ok. Además, si se observa

el *log* puede verse la operación que se ha ejecutado en MySQL y el campo de error de la respuesta de MySQL como null, que representa un ok.

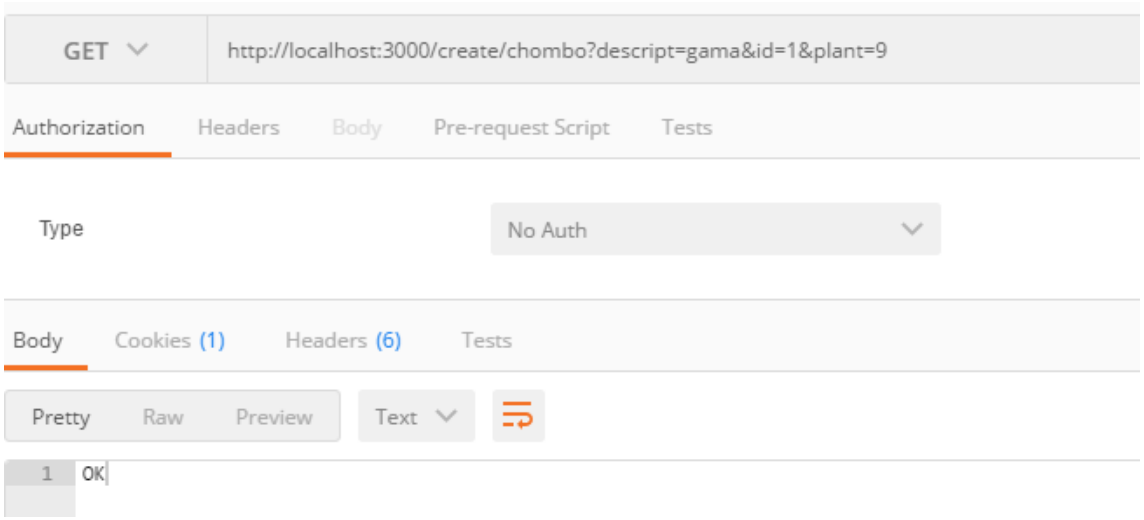


Ilustración 42 Cliente REST al Crear Chombo

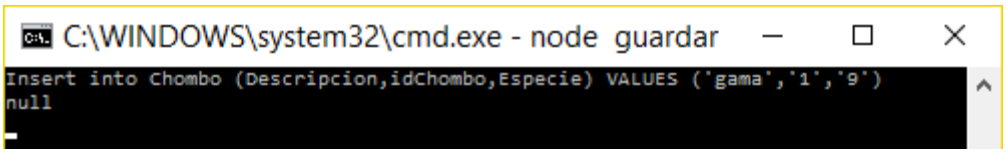


Ilustración 43 Log al crear Chombo

5.6.3.b Prueba de crear Actuador

Después de crear un Chombo se procede a probar la creación de actuadores. Se siguió la misma metodología realizando diversas peticiones. Entre estas peticiones se envió la que es visible en la ilustración 39. La respuesta devuelta fue un ok. Por otra parte, se puede apreciar en el *log* de la ilustración 38 la información que ha llegado y el JSON producido con ella, el retorno de ok de la base de datos, el id del actuador y el valor del JSON después de borrarlo.

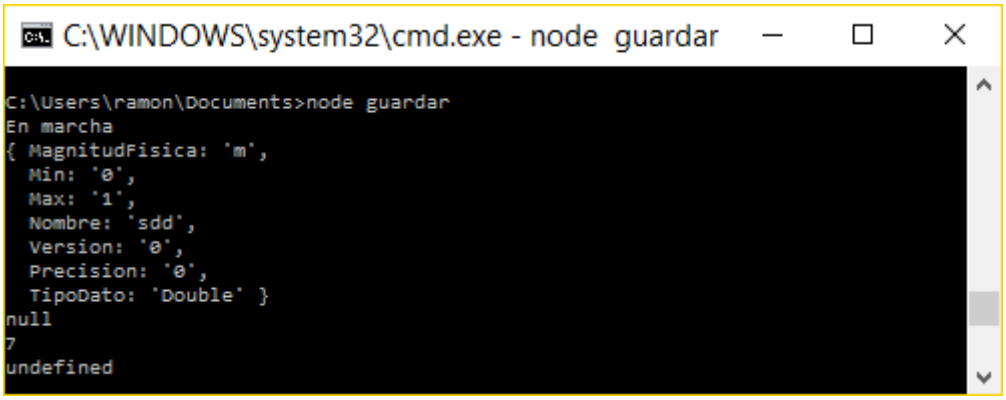


Ilustración 44 Log de crear Actuador

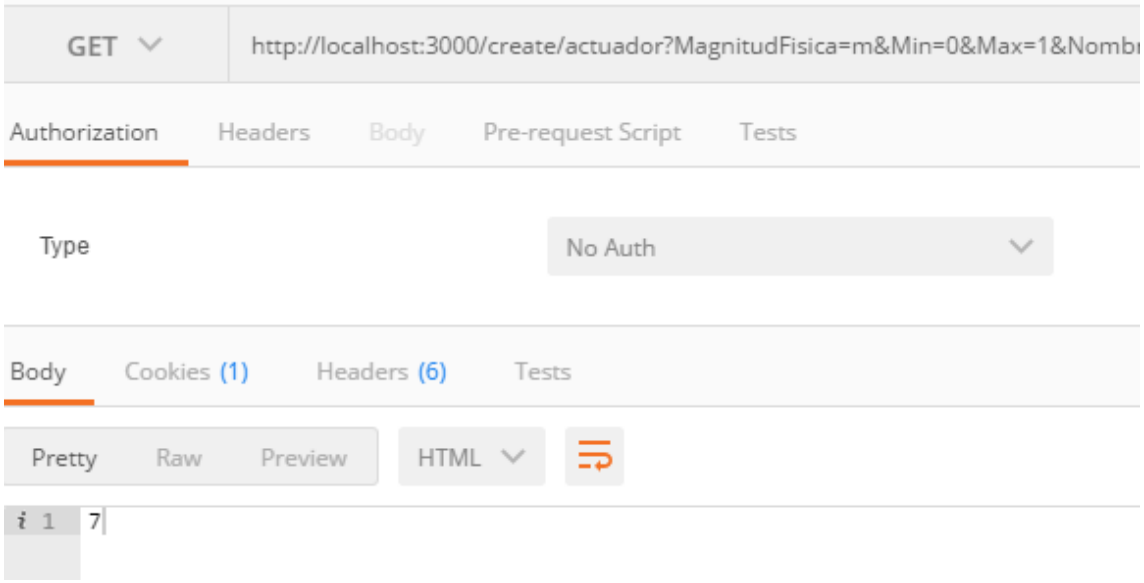


Ilustración 45 Cliente REST al crear el Actuador

5.6.3.c Prueba de insertar datos de sensores.

En esta sección se presenta una de las pruebas de inserción mediante el uso de la URL especificada en el punto 5.4. En esta prueba se envió como valor dos, dado que se restringe el valor entre 0 y 100 de este campo, desde el Chombo 0 desde el sensor 1. La primera vez y la segunda vez recibieron las mismas contestaciones, expuestas en la ilustración 42 desde la perspectiva del cliente.

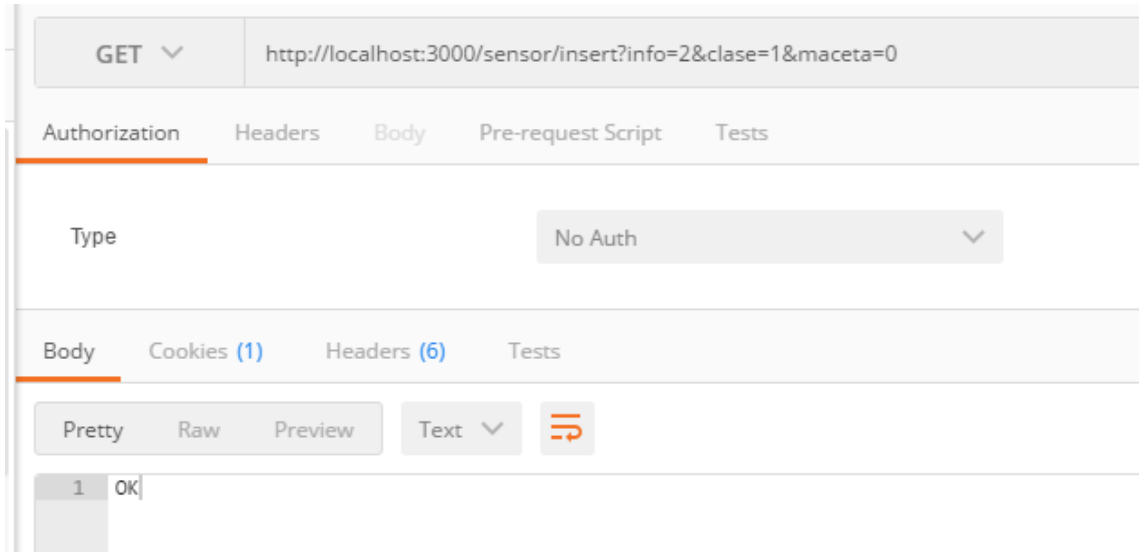


Ilustración 46 ***Cliente REST al Insertar datos históricos***

5.6.3.d Prueba de consultar información histórica de sensores

La prueba que se presenta en este apartado está vinculada a la información del apartado anterior. Por lo tanto, se espera que se devuelvan dos objetos JSON. Estos representan las instancias de la tabla HistoricoSensores con los valores vistos en el apartado anterior (5.6.3.c). El rango de visualización elegido es entre el año 1999 y 3000 por lo tanto debería sacar toda la información de la tabla. En este caso solo tiene dos elementos, por esto se puede obviar el limit con valor 50. Si se observa la respuesta que recibe el cliente REST en la siguiente ilustración, se pueden ver los dos documentos JSON que verifican la funcionalidad.

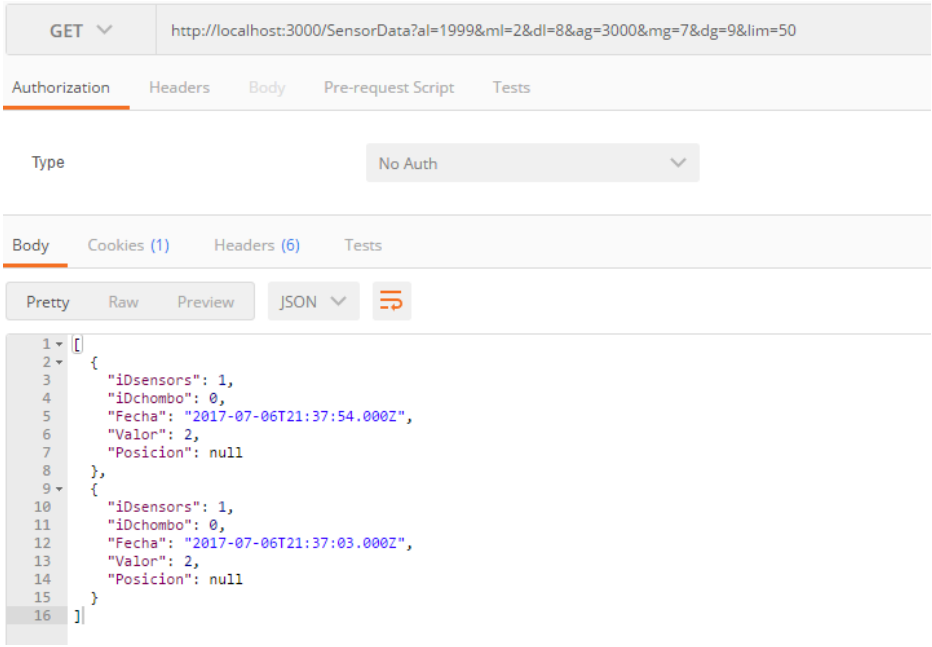


Ilustración 47 Cliente REST en la consulta de datos históricos

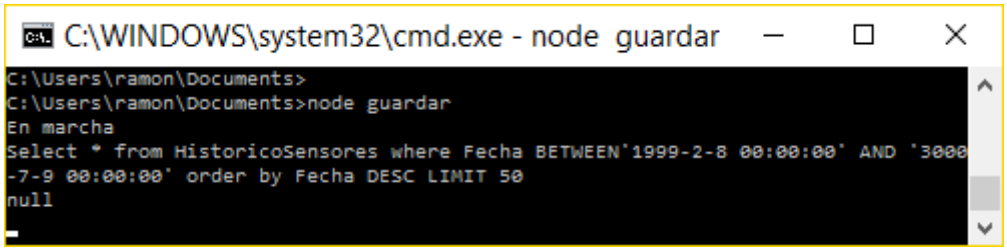


Ilustración 48 Log en la consulta de datos históricos

5.6.3.e Prueba de inserción de redundancia

En este caso si se ha probado a insertar la misma información que se ha visto en el apartado anterior. Con la finalidad de verificar que se evitan las inserciones de información redundante o de poco valor, como la información con una diferencia inferior al 0.5. Para realizar esto se ha realizado la misma petición que en el apartado 5.6.3.c. La respuesta ha sido un 200 pero con el mensaje de Información redundante (Ilustración 43) para avisar que no se ha insertado y no es ningún error interno.

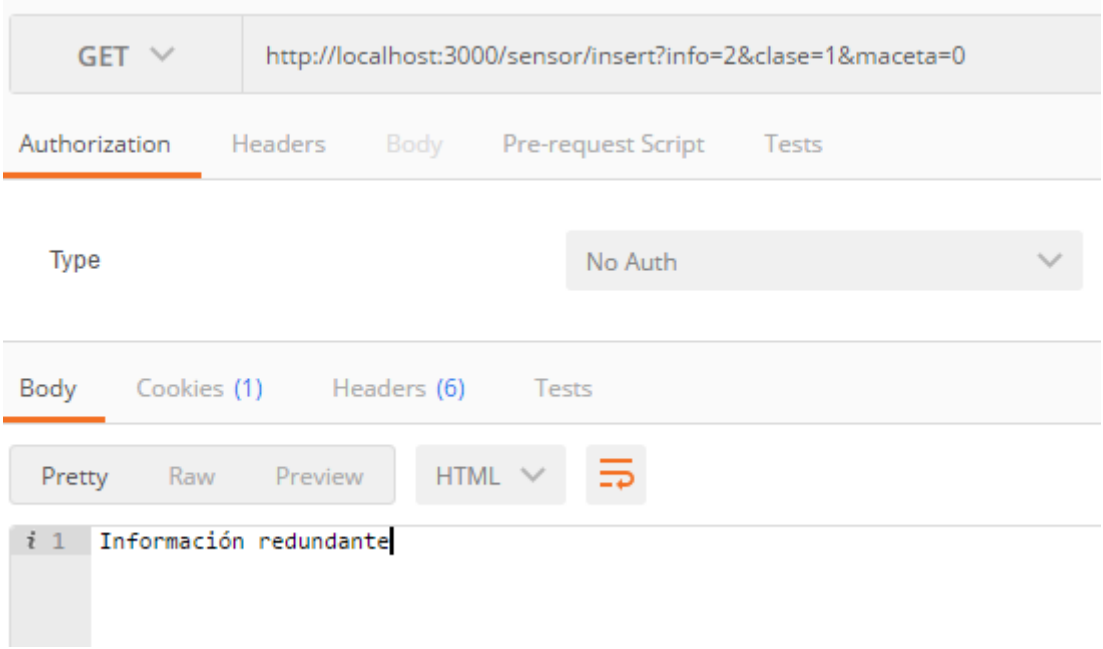


Ilustración 49 Cliente REST al insertar información redundante

Finalmente se exponen la agrupación de las pruebas realizadas por funcionalidad y los resultados obtenidos de forma resumida en la siguiente tabla.

N.º Test	Descripción	Comprobación
Test 1	Conexión del simulador con Artefacto BDD. Responde a la característica CA5.	Comprobación con éxito
Test 2	Creación de un Chombo vía API.	Comprobación con éxito
Test 3	Creación de un actuador vía API.	Comprobación con éxito
Test 4	Creación de un sensor vía API.	Comprobación con éxito
Test 5	Inserción de información de sensores.	Comprobación con éxito
Test 6	Inserción de información de actuadores.	Comprobación con éxito
Test 7	Consultar información histórica. Responde a la característica CA8.	Comprobación con éxito
Test 8	Consultar información de Chombos.	Comprobación con éxito
Test 9	Selección de Casos. Se ha probado la devolución de los valores -1, acción positiva, acciones negativas.	Comprobación con éxito
Test 10	Inserción de Casos y Plantas. Responde a las características CA1, CA6, CA7.	Comprobación con éxito

Tabla 11 Pruebas realizadas

Adicionalmente se han realizado pruebas de la base de datos histórica y el API REST. Pruebas de stress para valorar las características CA3 y CA9.

5.7 Conclusiones

En este apartado se han presentado los objetivos logrados mediante la implementación de la base de datos encargada de la información histórica y la base de datos encargada de realizar el apoyo al conocimiento basado en casos, junto a sus módulos de comunicación con el resto de componentes del sistema. Además, se han mostrado algunas de las pruebas del sistema y las características que verifican. Algunas de estas características eran deseables como en el caso de CA4(Escalable con el incremento de macetas en el sistema), pero para este primer prototipo demasiado ambiciosa.
Por este motivo las siguientes características quedan para futuro CA4, CA9(Comunicación escalable y flexible entre los distintos componentes del sistema) ha sido verificada con los componentes que han estado listos para las pruebas, dejando para el futuro la integración con el *hardware* y la aplicación móvil, y CA3(Disponibilidad 24x7 del histórico de casos) ha sido verificada mediante peticiones masivas desde un cliente, pero no se ha verificado el nivel de concurrencia que podría aguantar.

6. Conclusiones

En este proyecto se han analizado diversos enfoques para llegar a ofrecer las funcionalidades requeridas por los distintos componentes desde el apartado de análisis de tecnologías hasta el apartado de implementación, pasando por el apartado de diseño. En este capítulo se profundizará en los objetivos logrados, el estado del sistema global y las diferentes líneas de trabajo futuro que puedan partir de este primer prototipo.

6.1 Trabajo realizado

En este apartado se presentan las funcionalidades que sido han implementado en el sistema durante este proyecto. Estas funcionalidades se agrupan por los objetivos, presentados en la Introducción de este trabajo, que resuelven.

6.1.1 Apoyo al conocimiento basado en casos

Se ha logrado crear un prototipo que permite el apoyo al razonamiento basado en casos mediante la devolución de casos útiles para las situaciones y la capacidad de inserción de nuevas reglas que no existan en la tabla de casos.

El sistema tiene la capacidad de realizar inserciones de nuevos casos y modificaciones sobre la valoración de los casos (Índice de Confianza del caso). Por lo tanto, con la implementación del sistema se ha tenido en consideración la existencia de estas variaciones en la entidad de casos y situación, para la creación de estructuras de optimización de consultas(Índices).

El sistema permite la devolución de las listas de casos referentes a una especie, las acciones con mejor índice de confort dada la exactitud necesaria o requerida de similitud de la situación actual en que se encuentre el agente, y los datos característicos de las plantas como pueden ser los pesos o atributos ideales. Además, se ha implementado pensando en la flexibilidad como se explicó en el apartado de diseño del artefacto.

6.1.2 Acceso a los datos históricos de la red de Chombos

Se ha logrado la creación de la base de datos histórica para sensores y actuadores junto con una interfaz de acceso que permite limitar las operaciones que otros componentes del sistema pueden realizar sobre esta.

El sistema implementado permite la comunicación de la interfaz de usuario con la información histórica generada por la red de sensores subyacente al jardín. Funcionalidad independiente al tipo de dispositivo donde se encuentre

implementada la interfaz, dado que se ha utilizado para la comunicación REST, decisión explicada en el análisis de tecnologías.

Además, se permite la inserción de Chombos, tipos de sensores y actuadores de la forma más flexible que un sistema fuertemente dependiente de esquema como es el relacional habilita. Como se explicó en el apartado de diseño de la base de datos histórica, se ha decidido apostar por un diseño con homogeneidad de datos dejando el control de la consistencia de los datos a la interfaz que envuelva el acceso.

6.1.3 Acceso a las características de las especies

Se ha logrado que el sistema multiagente pueda acceder a la información de pesos, temperatura ideal y humedad ideal relativas a una especie. Este acceso puede ser tanto de consulta como de inserción.

6.1.4 Análisis de los objetivos logrados

Siendo estos logros el resultado de la coordinación con el resto de los trabajos simultáneos para el proyecto general. De esta forma se han llevado a cabo una serie de decisiones y modificaciones constantes en todas las iteraciones del ciclo de vida.

Se ha alcanzado la integración con todos los proyectos paralelos que se han implementado. Dejando preparadas las pruebas para verificar la integración con los otros componentes. Por otro lado, al llevarse a cabo este prototipo se podría pensar en una futura comercialización. En este caso, será necesario un sistema distribuido y escalable por número de usuarios, macetas y jardines. Entorno difícil de abordar desde una base relacional como MySQL que no es demasiado escalable, como se explica en el análisis de tecnologías. Por este motivo se ha dejado preparado el diseño de la estructura de la información para el uso de MongoDB. Dado que esta tecnología destaca por su capacidad de escalado horizontal, debida en gran parte a permitir la distribución de la información entre diferentes nodos.

6.2 Trabajo futuro

Dadas las circunstancias que han estado presentes durante la realización del proyecto, han aparecido ideas que no han podido ser implementadas, pero que serían de gran interés para un futuro, ya que mejorarían sustancialmente algunos aspectos del objetivo final.

6.2.1 Trabajo futuro para el apoyo del razonamiento basado en casos

Sería interesante la realización de un sistema de persistencia de la información que no dependa de un único nodo para evitar convertirlo en un punto único de fallo. Por lo tanto, es interesante realizar una división de la información más acercada a los interesados que van a utilizarla. De forma que quede un diseño del sistema con información local útil replicada para cada Chombo que la necesite, pero con una interfaz global a la que se puedan realizar consultas sobre toda la información. Siendo así una base para un sistema de almacenamiento federado.

Otro punto de interés para el futuro es dotar de mayor funcionalidad a las bases de datos mediante comunicaciones reactivas a las operaciones de las alteraciones de las entidades. De forma que se puedan advertir de ciertos comportamientos o patrones de alteraciones de información coherente pero incorrecta o que usualmente genere errores. Por ejemplo, se podría aplicar a la modificación constante de la valoración de algún caso, dado que en teoría los casos rara vez deberían modificarse.

6.2.2 Trabajo futuro para el sistema global

Para una mayor capacidad de análisis del entorno, se podría desarrollar un sistema de aprendizaje automático, para que con un sistema en funcionamiento y con los datos que este provee, se pueda mejorar la toma de decisiones. Así como hacer un análisis estadístico de los datos históricos.

Para evitar que el servidor sea un único punto de fallo (aunque los agentes físicos sigan trabajando, se corta toda la comunicación con los usuarios), podría establecerse de forma distribuida, ya sea usando replicación o completa distribución entre los agentes.

Si se dispone del sistema distribuido anterior, podría llevarse al siguiente nivel, en el cual toda la comunicación se cedería a los propios agentes, mejorando su coordinación al informar estos directamente de las condiciones de entorno individuales al resto del sistema, sin necesidad de usar al simulador como intermediario.

Implementar una gestión de usuarios dentro de la aplicación, de forma que se puedan crear usuarios desde la propia aplicación e identificarse con las credenciales correspondientes. Esto permitiría gestionar un mismo jardín desde distintos dispositivos siempre y cuando se disponga de las credenciales del usuario propietario de la red.

En este punto también podrían incluirse distintos niveles de usuario, ya que el usuario experto puede necesitar mayor control sobre su jardín (por ejemplo, alterando los pesos de los parámetros de las plantas) mientras que un usuario con menos conocimientos no iría más allá de la monitorización de sus chombos.

La posibilidad de parametrizar y gestionar distintas redes de chombos, pues un mismo usuario puede tener la necesidad de gestionar dos jardines independientes el uno del otro y en distintas localizaciones.

Disponer de un render en 3 dimensiones que represente de forma más fiel el movimiento de los chombos dentro del jardín, con posibilidad de hacer *zoom* y rotar la vista para obtener una mejor perspectiva.

Dotar a la aplicación de características que mejoren su usabilidad y funcionalidad para así resultar atractiva a un mayor número de usuarios: introducción de gestos para la navegación y realización de acciones, filtros de búsqueda, subir fotos propias de plantas, visionado en *streaming* de la planta a través de su cámara cenital, conexión con redes sociales, multilenguaje, etc.

Una forma de facilitar el despliegue del sistema completo podría ser el uso de la plataforma Docker, dado que nos permite automatizar la inyección de dependencias del sistema y poner en marcha todos los componentes mediante docker-compose.

Como se explicó en el apartado de análisis de tecnologías, hay otras tecnologías que podría aportar beneficios en entornos más complejos, ya sea por un aumento en la complejidad de la distribución del sistema o por un entorno de red enormemente limitante en ancho de banda. Por estos motivos podría ser interesante utilizar para la generación de arquitecturas *pub/sub* ZeroMQ o MQTT como módulos de comunicación para el guardado de la información histórica y consulta de datos en tiempo real.

7. Bibliografía

1. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M. and Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials*, [online] 17, pp.7-10. Available at: http://ieeexplore.ieee.org/Xplore/home.jsp [Accessed 1 Jun. 2017].
2. https://github.com/mqtt/mqtt.github.io/wiki. (2017). server support. [online] Available at: https://github.com/mqtt/mqtt.github.io/wiki/Server%20support [Accessed 5 Jun. 2017].
3. Delgado Sanchis, A. (2016). Comunicación entre módulos para la construcción de jardines inteligentes. Universidad Politécnica de Valencia, ETSINF.
4. MongoDB-Based Repository Design for IoT-Generated RFID/Sensor Big Data. (2015). *IEEE Sensors Journal*, [online] 16. Available at: http://ieeexplore.ieee.org/ [Accessed 3 May 2017].
5. Happ, D., Karowski, N., Menzel, T., Handziski, V. and Wolisz, A. (2016). Meeting IoT platform requirements with open pub/sub solutions. *Annals of Telecommunications*, 72.
6. websystique. (2017). *Spring MVC 4 RESTFul Web Services CRUD Example+RestTemplate*. [online] Available at: http://websystique.com/springmvc/spring-mvc-4-restful-web-services-crud-example-resttemplate/ [Accessed 25 Jun. 2017].
7. Benchmark of MQTT servers. (2017). [ebook] Available at: http://www.scalagent.com/IMG/pdf/Benchmark_MQTT_servers-v1-1.pdf [Accessed 22 May 2017].
8. Open GIS Consortium, Inc. OpenGIS Simple Features Specification For SQL. (1999). p.16.
9. Oracle Corporation, (2017). *MySQL 5.7 Reference Manual*. [online] Available at: https://dev.mysql.com/doc/refman/5.7/en/ [Accessed 9 Jan. 2017].
10. Microsoft,(2017) Chapter 8: Communication.[online] Available at: https://msdn.microsoft.com/en-us/library/hh404091.aspx [Accessed 5 Jan. 2017]
11. Docker. (2017). What is a Container. [online] Available at: https://www.docker.com/what-container [Accessed 1 Jun. 2017].
12. Hong, K. (2017). Docker container vs Virtual machine - 2017. [online] Bogotobogo.com. Available at: http://www.bogotobogo.com/DevOps/Docker/Docker_Container_vs_Virtual_Machine.php [Accessed 1 Jun. 2017].
13. Cloudera, I. (2017). *Chicago Data Summit: Apache HBase: An Introduction*. [online] Slideshare.net. Available at: https://www.slideshare.net/cloudera/chicago-data-summit-apache-hbase-an-introduction [Accessed 5 Jan. 2017].

14. Developer.couchbase.com. (2017). *Data Management*. [online] Available at: https://developer.couchbase.com/documentation/server/4.6/concepts/data-management.html [Accessed 5 Jan. 2017].

15. Developer.couchbase.com. (2017). *High Availability and Replication Architecture*. [online] Available at: https://developer.couchbase.com/documentation/server/current/architecture/high-availability-replication-architecture.html [Accessed 5 Jan. 2017].

16. Developer.couchbase.com. (2017). *Providing transactional logic*. [online] Available at: https://developer.couchbase.com/documentation/server/3.x/developer/dev-guide-3.0/transactional-logic.html [Accessed 6 Jan. 2017].

17. Es.wikipedia.org. (2017). *Comparación de sistemas administradores de bases de datos relacionales*. [online] Available at: https://es.wikipedia.org/wiki/Anexo:Comparaci%C3%B3n_de_sistemas_administradores_de_bases_de_datos_relacionales [Accessed 3 Jan. 2017].

18. Hbase.apache.org. (2017). *Apache HBase – Apache HBase (TM) ACID Properties*. [online] Available at: https://hbase.apache.org/acid-semantics.html [Accessed 5 Jan. 2017].

19. Kkovacs.eu. (2017). *Cassandra vs MongoDB vs CouchDB vs Redis vs Riak vs HBase vs Couchbase vs Hypertable vs ElasticSearch vs Accumulo vs VoltDB vs Scalaris comparison :: Software architect Kristof Kovacs*. [online] Available at: https://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis [Accessed 3 Jun. 2017].

20. Merriman, D. (2017). *About M102*. [online] University.mongodb.com. Available at: https://university.mongodb.com/courses/M102/about [Accessed 15 Jan. 2017].

21. Merriman, D. (2017). *sharding and data distribution*. [online] YouTube. Available at: https://www.youtube.com/watch?v=xvWzS9j7IIY [Accessed 20 Jan. 2017].

22. Docs.mongodb.com. (2017). Atomicity and Transactions — MongoDB Manual 3.4. [online] Available at: https://docs.mongodb.com/manual/core/write-operations-atomicity/ [Accessed 26 Jul. 2016].

23. Orientdb.com. (2017). *Distributed Architecture | OrientDB Manual*. [online] Available at: http://orientdb.com/docs/2.0/orientdb.wiki/Distributed-Architecture.html [Accessed 5 Jun. 2017].

24. Richard, S. (2017). Introduction to NoSQL and DBaaS. [online] BigDataUniversity. Available at: https://archive.bigdatauniversity.com/courses/introduction-to-nosql-and-dbaas/ [Accessed 5 Jul. 2016].

25. Cartago.sourceforge.net. (2017). CArtAgO. [online] Available at: http://cartago.sourceforge.net/ [Accessed 5 Jan. 2017].

26. Planthardiness.ars.usda.gov. (2017). USDA Plant Hardiness Zone Map. [online] Available at: http://planthardiness.ars.usda.gov/PHZMWeb/ [Accessed 6 Jun. 2017].

27. Seleniumhq.org. (2017). Selenium - Web Browser Automation. [online] Available at: http://www.seleniumhq.org/ [Accessed 16 Jan. 2017].

28. Delgado Sanchis, A. (2016). Comunicación entre módulos para la construcción de jardines inteligentes. Universidad Politécnica de Valencia, ETSINF.

29. Guzman Godia, A. (2016). Informatización de un módulo para la construcción de jardines inteligentes. Universidad Politécnica de Valencia, ETSINF.

30. Gil Rodríguez, C. (2016). Desarrollo de aplicación web para gestionar módulos inteligentes para la construcción de jardines. Universidad Politécnica de Valencia, ETSINF.

31. Ferrando Ferragut, L. (2016). Monitorización y control de los módulos de un jardín inteligente. Universidad Politécnica de Valencia, ETSINF.

32. http://www.emse.fr/~boissier/enseignement/maop11/doc/cartago-main-api/index.html. server support. [online] Available at: http://www.emse.fr/~boissier/enseignement/maop12/doc/c4jason-api/cartago/invoke_obj.html [Accessed 25 Jun 2017].

33. http://www.telegraph.co.uk. [online] Available at: http://www.telegraph.co.uk/gardening/tools-and-accessories/the-best-apps-to-identify-unknown-plants-and-flowers/ [Accessed 1 Dec 2016].

34. http://blog.parrot.com. (2015) [onliine] Available at: http://blog.parrot.com/2015/01/05/ces-2015-flower-power-pot-the-smart-pot-that-grows-healthy-plant/ [Accessed 1 Dec 2016].

35. https://www.alibaba.com [online] Available at: https://www.alibaba.com/showroom/smart-flower-pot.html [Accessed 1 Dec 2016].

36. http://es.aliexpress.com [online] Available at: http://es.aliexpress.com/popular/smart-flower-pot.html [Accessed 1 Dec 2016].

37. https://www.google.com [online] Available at: https://www.google.com/patents/US3961444 [Accessed 2 Dec 2016].

38. https://www.google.com [online] Available at: https://www.google.com/patents/US4403443 [Accessed 2 Dec 2016].

39. https://www.google.com [online] Available at: https://www.google.com/patents/US6070359 [Accessed 2 Dec 2016].

40. https://www.google.com [online] Available at: https://www.google.com/patents/US4108350 [Accessed 2 Dec 2016].

41. https://www.google.com [online] Available at: https://www.google.com/patents/US3069807 [Accessed 2 Dec 2016].

42. https://www.google.com [online] Available at: https://www.google.com/patents/US20060272210 [Accessed 2 Dec 2016].

43. https://www.google.com [online] Available at: https://www.google.com/patents/US6243986 [Accessed 2 Dec 2016].

44. http://www.actahort.org [online] Available at: http://www.actahort.org/members/showpdf?booknrarnr=417_4 [Accessed 4 Dec 2016].

45. http://www.fliwer.com [online] Available at: http://www.fliwer.com/# [Accessed 4 Dec 2016].

46. http://delivery.acm.org [online] Available at: http://delivery.acm.org/10.1145/1360000/1357335/p1797-consolvo.pdf?ip=158.42.174.22&id=1357335&acc=ACTIVE%20SERVICE&key=DD1EC5BCF38B3699%2E016407C0B79CBB66%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=677700695&CFTOKEN=86003516&__acm__=1475767424_edd7e7a77bbde675bf3f0ef7fc1ec83d [Accessed 4 Dec 2016]

47. Ministerio de cultura y pesca. (2013) Proyecto sigAGROasesor. Available at: http://agroasesor.es/es/ [Accessed 7 Dec 2016]

48. Emmi Pizzella, L. A. (2011). Entorno de simulación para flota de robots orientado a la gestión de malas hierbas en cultivos. Universidad Complutense de Madrid, Máster en Investigación en Informática, Facultad de Informática, Departamento de Ingeniería del Software e Inteligencia Artificial.

49. https://myflowerpower.parrot.com. (2016) [online] Available at: https://myflowerpower.parrot.com/#plantdb [Accessed 10 Dec 2016].

50. Oddity Central - Collecting Oddities. (2017). Bad with Plants? This High-Tech Flower Pot Can Keep Any Plant Alive. [online] Available at: http://www.odditycentral.com/technology/bad-with-plants-this-high-tech-flower-pot-can-keep-any-plant-alive.html [Accessed 7 Oct. 2017].

51. Amazon.com. (2017). Amazon.com: Parrot Flower Power - Wireless Indoor/Outdoor Bluetooth Smart Plant Sensor with Free dedicated App - Green: Electronics. [online] Available at: https://www.amazon.com/Parrot-

Flower-Power-Bluetooth-dedicated/dp/B00FOM2Y6W [Accessed 7 Oct. 2017].

52. Kawakami, A., Tsukada, K., Kambara, K. and Siio, I. (2017). PotPet: Pet-like Flowerpot Robot. [ebook] Available at: http://sappari.org/pdf/potpet_tei2011.pdf [Accessed 8 Oct. 2017].

53. Docs.oracle.com. (2017). Tables and Table Clusters. [online] Available at: https://docs.oracle.com/cloud/latest/db112/CNCPT/tablecls.htm [Accessed 8 Jan. 2017].

Anexo A: Glosario

- **Actuador**: Dispositivo capaz de transformar energía eléctrica en la activación de un proceso con la finalidad de generar un efecto sobre un proceso automatizado.

- **Agente**: Aplicación informática con capacidad para decidir cómo actuar para alcanzar sus objetivos.

- **API:** Conjunto de funciones llamadas mediante un protocolo

- **APP**: Acrónimo de la palabra aplicación, al que se recurre para referirse a la aplicación móvil que se encargará de la monitorización e interacción del sistema.

- **Base de casos**: Conjunto estructurado de información que representa los casos del sistema.

- **Chombo**: Contenedor de una o varias plantas, con sensores, actuadores y capacidad de movimiento, que es controlado por un agente.

- **Fuentes externas**: Origen de los datos que no son obtenidos directamente por el sistema.

- **Fuentes internas:** Componentes del sistema que generan y entregan acceso a información de este.

- **Integración de datos**: Combinación coherente de datos originarios de diferentes fuentes en una vista unificada.

- **Jardín:** Red de chombos asociada a un mapa.

- **Maceta-robot:** Contenedor de una o varias plantas, con sensores, actuadores y capacidad de movimiento.

- **Razonamiento basado en casos**: Metodología de la inteligencia artificial con la finalidad de resolver problemas recordando situaciones previas similares y reutilizando la información y el conocimiento sobre esa situación.

- **Sensor**: Dispositivo capaz de detectar magnitudes físicas y transformarlas en valores binarios.

- **Simulador**: Reproducción del sistema en un entorno con unas características específicas.

- **Sistema de agentes**: Modelo de programación conformado por agentes en comunicación, que toman el papel de los usuarios, tomando control de las acciones necesarias que se deben llevar a cabo para cumplir los objetivos marcados. Pudiendo crear nuevos objetivos y variaciones para cada caso particular, en función del entorno.

- **Sistema gestor de base de casos(SGBC)**: Software utilizada para la extracción e inserción de casos en la base de casos.

- **Sistema**: Se usa este término para referirse globalmente a todas las partes que componen el proyecto, desde la APP y el sistema de agentes hasta los sensores y actuadores, pasando por las conexiones y el sistema de persistencia.

- **Situación:** conjunto de valores tomados del estado actual que manejan los agentes, incluyendo valores derivados como pueden ser distancias a localizaciones específicas.

- **UML:** Lenguaje de Modelado Unificado estandarizado por Object Management Group®.

- **Web scrapper:** Componente software cuya funcionalidad es la extracción de datos de sitios web.

- **Wrapper:** Interfaz que da acceso a las funciones de un componente.