Escola Politècnica Superior de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO DE FIN DE CARRERA

TÍTULO DEL TFC: Sistema de Gestión de Regadío mediante una arquitectura distribuida J2EE.

TITULACIÓN: Ingeniería Técnica de Telecomunicaciones especialidad en Telemática.

AUTOR: Silvia Florensa Labrador.

DIRECTOR: Pablo Royo Chic, Juan López Rubio.

FECHA: 28 de julio de 2008.

Resumen

El sector agrícola es uno de los mayores consumidores de agua, la modernización de sistemas de riego se considera una respuesta para alcanzar y mantener un uso eficiente del agua. El cambio climático está alterando las precipitaciones del planeta en periodos de grandes sequías y de lluvias torrenciales. Este hecho obliga al agricultor a tener un seguimiento minucioso del agua suministrada; dependiendo del periodo en el que se encuentra y de esta forma no suministrar más agua de la necesaria. Realizar una correcta gestión de regadío comienza a tomar mucha importancia cuanto mayor sea la superficie a controlar y lo distribuida que este.

La mayoría de los programas informáticos que controlan la gestión de regadío no poseen una tecnología para la obtención de datos en tiempo real y de forma remota.

Para solucionar este problema se plantea una aplicación Web de gestión de fertirrigación convencional de regadío. Al ser una aplicación Web, el cliente podrá conectarse al ordenador de riego desde su casa u oficina y obtener los datos de regadío en tiempo real. La aplicación Web ofrece la posibilidad de tener superficies de cultivos separadas a cientos de kilómetros y gestionarlas a la vez.

Las funcionalidades de esta aplicación Web son control de riego, fertilización, pH, bombeo, limpieza de filtros y detección de averías; pudiendo ser configuradas por tiempo (segundos) o por volumen (es un contador de litros suministrados). Gracias a estos datos sabemos como se está realizando el riego, el estado de las bombas, si ha habido alguna anomalía. Por ejemplo, si se va a terminar los fertilizantes que se van a inyectar a los cultivos; es necesario repostarlos para seguir con su tratamiento. Si este tratamiento fuera suspendido, se podría llevar a cabo un incorrecto crecimiento del cultivo o un aumento de plagas innecesarias. Otra funcionalidad importante es la de limpieza los filtros ya que un incorrecto mantenimiento podría provocar un imperfecto filtrado de agua dejando pasar partículas peligrosas para el sistema electrónico de regadío.

La aplicación Web permite incluir información sobre la empresa, de modo que cualquier usuario de Internet pueda consultar: información sobre la empresa, que productos cultivan, localización de sus fincas y un formulario de contacto para cualquier consulta que tengan. Además la aplicación permite realizar compras por Internet.

En este documento vamos a describir la arquitectura, diseño y la implementación de la aplicación. Además finalizaremos explicando el manual de instalación de la misma.

Title: TFC/PFC Model
Author: Silvia Florensa Labrador
Director: Pablo Royo Chic, Juan López Rubio
Date: July, 28th 2008

Overview

The agricultural sector is one of the biggest consumers of water, modernization of irrigation systems is considered as a response to attain and maintain an efficient water use. Climate change is altering rainfall on the planet in periods of great droughts and torrential rains. This fact obliges the farmer to have a careful monitoring of water supplied; depending on the period in which lies and not provide more water than necessary. Perform management of irrigation proper begins to have great importance the larger the surface to control and what it is distributed.

Most computer programs that control the management of irrigation not possess a technology to obtain data in real time and remotely.

To solve this problem raises Web application for conventional irrigation systems. Being a Web application, the customer will be able to connect to the computer irrigation from home or office and obtain data irrigated in real time. The Web application offers the possibility of having separate areas of crops and hundreds of miles manage at once.

The functionalities of this Web application are irrigation control, fertilization, pH levels, pumping, filter cleaning and system trouble shooting. These functions can be controlled by time (per second) or volume (per litre). Thanks to these data as we know is being conducted irrigation, the status of pumps, and if there had been any anomaly. For example, if you are going to finish fertilizers to be injected to crops; it is necessary to refuel and continue your treatment. If this treatment is suspended, it could carry out a wrong crop growth or an increase in pests unnecessary. Another important feature is the cleaning filters since an improper maintenance could result in an imperfect filtering water particles letting dangerous to the electronic system of irrigation.

The Web application allows you to include information about the company, so that any Internet user can consult: information about the company, products grown, location of their farms and a form of contact for any queries they have. Besides the application allows sales over the Internet.

In this document we will describe the architecture, design and application deployment. In addition we will finish explaining the installation manual of the same.

Dedico este proyecto a toda la gente que confía en mí.
A mis padres, que siempre han estado a mi lado,
porque a ellos les debo
todo lo que he conseguido, he sido y lo que soy.
A mis hermanos, que han apostado por mí,
y me han dado sus mejores consejos
a nunca desistir de mis sueños.
A todos mis amigas/os, que estando cerca o lejos,
han sabido demostrarme su amistad y su cariño.
En especial a Beatrile, Cruxe y Sus.
También le dedico este proyecto,
a mis dos tutores.
A Oscar Illescas, que me ha dado muy buenos consejos
y que sin él este proyecto no hubiese sido posible.
A Mari Jose Callizo por enseñarme el mundo del diseño.
Y a Santiago Díaz por mostrarme el maravilloso mundo de J2EE.

ÍNDICE

FIGURAS

INTRODUCCIÓN

La agricultura española se desenvuelve en unas condiciones físicas bastante difíciles, las precipitaciones en territorio español cada vez están siendo más escasas y la demanda de agua de riego es cada vez mayor, lo que obliga a ser mejor gestionada.

La incorporación de las nuevas tecnologías en el mundo del riego están facilitando el ahorro de agua; gracias a la ayuda programas informáticos.

Hoy en día hay una gran variedad de programas informáticos para la gestión de regadío, pero apenas existe la opción de disponer la información en tiempo real de los cultivos existentes desde cualquier punto del mundo. Este proyecto ofrece una solución a este problema.

Quiero hacer especial hincapié que la idea de este TFC ha sido de “cosecha propia”, ya que pertenezco a una familia de agricultores y conozco el problema de cerca y sus necesidades.

El controlador de riego para el cual está referido este proyecto es el Agronic 4000. Incorpora un equipo para el envío de mensajes cortos SMS a teléfonos móviles GSM para el envío de informes de la gestión de regadío. También dispone de un programa para conexión a PC permitiendo manejar el controlador de riego con mayor facilidad. Pero el inconveniente es que obliga al usuario a tener instalado el programa para dicha gestión.

El objetivo de este proyecto se basa en poder satisfacer las necesidades de los agricultores, para poder acceder al controlador de riego desde cualquier ordenador y desde cualquier punto del mundo, sin la obligación de tener instalado el programa de ordenador que proporciona Agronic. La solución planteada en este proyecto es una aplicación Web para la gestión de fertirrigación utilizando la tecnología J2EE.

En los inicios del TFC me puse en contacto con la empresa Progres, es la que suministra estos ordenadores de riegos. Les expliqué el problema con la solución que propongo y si me podían facilitar algo de código poder atacar el ordenador de riego vía Internet, pero me cerraron las puertas. A pesar de esto seguí con el proyecto ya que es un avance tecnológico para los agricultores. Por este motivo la base de datos propuesta en el TFC simula al ordenador de riego.

La arquitectura del diseño de la aplicación Web a implementar será el modelo, vista, controlador (MVC), con el principal objetivo de separar la lógica del negocio y los datos asociados a la aplicación (modelo) de la presentación de los datos (vista) y el procesamiento de las peticiones y componentes para toma de decisiones de la aplicación (controlador). Se decidió trabajar en diferentes capas para facilitar la programación, mantenimiento y control de errores de a la aplicación.

La aplicación estará empaquetada en dos módulos independientes (pueden estar en una o dos máquinas). El primer modulo es el de aplicación/Web, en él se ubicará la vista y el controlador. El controlador se conectará mediante el protocolo RMI (*Java Remote Method Invocation*) con el segundo módulo basado en EJBs. El servidor de aplicaciones que se utilizará como software base para ubicar tanto el módulo Web como el de EJBs será BEA WEBLOGIC 8.1.

El funcionamiento interno de la aplicación Web (a lo largo del TFC se explica detalladamente) es el siguiente: el Cliente Web se conectará vía HTTP al modulo de Aplicaciones/Web. La respuesta del servidor serán JSP (Java Server Pages) y el cliente podrá realizar todas las peticiones y respuestas requeridas para la gestión del Regadío mediante dichas JSP's (vista), permitiendo generar contenido dinámico Web. Los Servlets (controlador) recibirán las peticiones del cliente, realizarán las operaciones requeridas invocando a la capa modelo y los resultados obtenidos se visualizarán en JSP's. El modelo constará de EJB (Enterprise JavaBeans) y DAO (capa de acceso a datos). Los EJB permiten ser invocados remotamente vía RMI (*Java Remote Method Invocation*) y realizar la lógica de negocio. Para tener acceso a la capa de almacenamiento y administración de datos los EJB utilizarán un modelo de objetos DAO.

El documento se organiza de la siguiente manera: Se comienza con un estudio del contexto tecnológico y análisis del problema, luego se especifica, diseña e implementa. Se acaba con las conclusiones.

En el primer capítulo se sitúa el problema, se analiza el contexto general todos los componentes necesarios de una finca para la realización del trabajo.

Ya en el segundo capítulo se realiza un análisis detallado de la arquitectura implementada.

En el tercer se describe el diseño completo de la aplicación, con los casos de uso, diagramas de clases y de secuencia. Se comentan los aspectos más interesantes a la hora de programar la aplicación.

El cuarto capítulo se explica la implementación de la aplicación, se muestran imágenes del resultado de la aplicación

Ya en el quinto capítulo, en el manual de instalación se explica los pasos a seguir para que cualquier persona pueda instalar la aplicación en cualquier ordenador windows XP, este capítulo tiene su importancia porque hay que seguir unos pasos muy rigurosos para que la aplicación funcione.

Finalmente el capítulo de conclusiones agrupa todos los aspectos que han sido de importancia para el proyecto y se habla con detalle del cumplimiento de objetivos y requisitos funcionales.

Se puede encontrar un capítulo de anexos en el que se evalúa toda la información relacionada con el trabajo.

CAPÍTULO 1. CONTEXTO TECNOLÓGICO Y ANALISIS DE OBJETIVOS.

1.1. Motivación

Hoy en día estamos sufriendo los achaques del cambio climático, donde la variación de temperatura que afecta a la evapotranspiración del suelo son tan variantes de unos días a otros, exige al agricultor a modificar el plan de riego constantemente, teniendo en cuenta el máximo ahorro de agua posible. Una gestión inteligente del regadío ayudaría a combatir los efectos del cambio climático.

El proyecto surge de una empresa de agricultores, con el fin de diseñar una aplicación Web para la gestión de regadío. La idea es poder realizar la automatización y control del regadío sin tener la necesidad de ir personalmente a la finca donde se encuentra el Agronic 4000 o tener que instalar en el ordenador el programa Agronic PC para su gestión.

1.2. Escenario de la finca e inconvenientes

En esta sección se va a explicar las partes principales que se pueden configurar desde la aplicación Web y forman parte para la gestión de regadío.

Una finca de frutales, principalmente consta de árboles y un cabezal de gestión de regadío. Los campos de frutales se dividen en diferentes sectores para poder facilitar el subministramiento de agua y fertilizantes. El cabezal de gestión de regadío es cerebro de la finca, en la figura 1 se muestra un esquema del cabezal de riego. En él, se encuentra el ordenador de riego Agronic 4000, es el encargado de controlar el sistema de riego ya que está conectado a todos los componentes del cabezal. Otros componentes son los fertilizantes, lugar donde se vierten todos los productos químicos. Para la mezcla de los productos químicos con el agua son necesarios unos agitadores para su perfecta disolución. Una vez mezclados son inyectados al agua que ha sido filtrada. El ordenador de riego a partir de las electroválvulas es capaz de ordenar por que sector ha de salir el agua con o sin fertilizantes.
En el apartado 1 de los anexos se muestran imágenes reales del cabezal de riego.

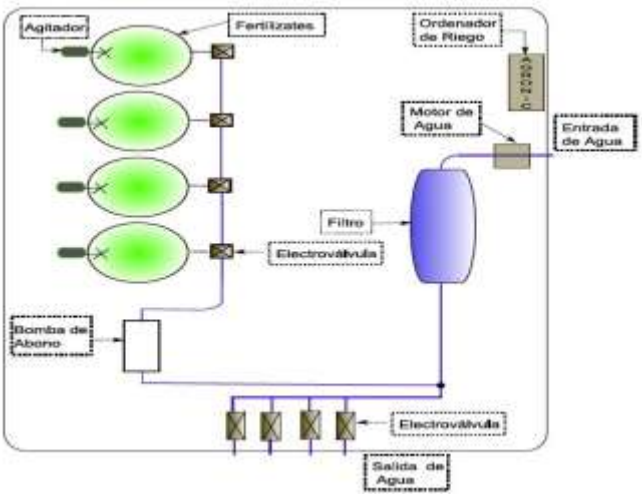

Figura 1 Cabezal de riego

El Agronic 4000 debe conectarse a un MODEM GSM para el envío de los mensajes a móviles o a equipos que tengan instalados Agronic PC.

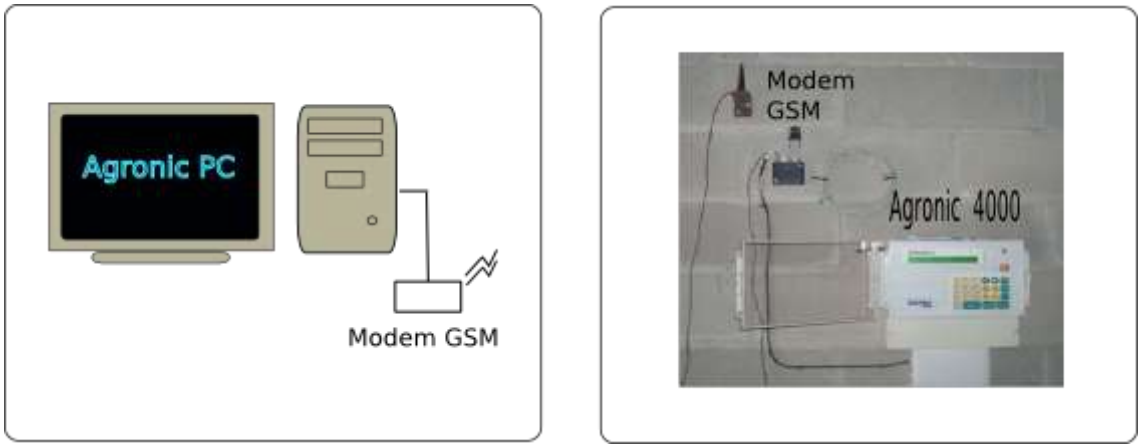

Figura 2 Conexión GSM

El Agronic PC es un programa que permite manejar los controladores de riego Agronic 4000 desde un PC. Tienes tres inconvenientes:

- Primero, necesita ser instalado en el PC.
- Segundo, obliga al cliente a trabajar siempre desde el mismo PC.
- Y tercero, Agronic PC no es un programa multiplataforma, sólo es soportado por el sistema operativo Windows.

1.3. Objetivo y solución

El objetivo es sustituir el software de Agronic por la aplicación Web para tener un control de toda la finca desde cualquier lugar con acceso a Internet (oficina, casa, etc); y poder conectarse desde cualquier maquina independientemente del sistema operativo instalado en el ordenador.

La solución planteada es el diseño de una aplicación Web para la gestión de regadío, la tecnología a utilizar es J2EE (Java 2 Enterprise Edition).

Al ser una aplicación Web, se diferencian dos partes en el portal:

- Una es la parte pública, donde cualquier usuario de la red podrá acceder a ella y obtener toda la información sobre la empresa.
- Y otra parte es la privada con acceso restringido por el administrador de la Web. Para el acceso a la zona privada del portal donde se realizar la gestión de regadío, es necesario un nombre de usuario y una contraseña.

1.4. Requisitos funcionales

Los requisitos funcionales para la zona pública, donde cualquier usuario de la red puede accedes a ella, son los siguientes:

- Quienes somos: descripción de la empresa.
- Donde estamos: descripción de donde se encuentran las fincas, con un plano para ubicarse.
- Productos: una descripción de los diferentes productos que se cultivan.
- Contactar: un formulario para que el usuario se pueda poner en contacto con la empresa.
- Álbum fotográfico de todos los productos y de toda la plantación que se tiene.

Se plantean los siguientes requisitos funcionales para la zona privada, gestión de regadío:

- Configuración del número de sectores en que se divide toda la finca controlada por un cabezal de riego.
- El administrador debe decir al sistema el número de cubas de fertilizantes que se utilizarán.
- Limpieza de filtros, ya que al filtrar el agua todas las impurezas se quedan en el filtro y es necesario ser limpiados para el buen funcionamiento de la gestión.
- El administrador tiene que gestionar las salidas del sistema de regadío, hay cuatro salidas diferentes, son las siguientes:
    - Salidas de las alarmas, se utilizan en caso de fallo del sistema.
    - Salidas de fertilizantes son para elegir que cuba de fertilizante se inyectará al sector.
    - Salidas de motores, el tiempo y paro de cada motor.
    - Salidas de generales, asignar la salida del cabezal de riego.
- Otro requisito funcional es, gestionar el caudal del agua para minimizar su consumo. Para el control de este consumo se debe controlar:
    - El caudal inyectado en las cubas de los fertilizantes para poder mezclarlo con las materias químicas.
    - Y el caudal de riego suministrado a todos los sectores.

- Crear un programa para activar toda la configuración dependiendo de los días deseados por el usuario.
- Y por último, realizar control de errores.

En la zona privada hay unos requisitos funcionales referenciados al administrador:

- Control de sesión: crear una sesión cuando el administrador entre a la gestión de regadío y cerrarla cuando él lo desee.
- Recordar contraseña: este requisito funcional es importante, por si el administrador olvida su contraseña pueda recordarla mediante un envío de un mail con la contraseña a su correo electrónico.

1.5. Tecnología utilizada

La zona pública se ha programado es HTML incorporando algunos elementos Flash y Ajax. Se decidió utilizar HTML ya que esta parte de la aplicación es estática. Flash porque se han creado varias animaciones y se cargan bastante rápido. Se muestra entre otras la página de inicio, la cabecera de todas las páginas y los menús. Por último, la tecnología Ajax para no tener que recargar toda la página ya que la cabecera, el menú y el pie de la página se repiten en toda la zona pública y sólo es necesario recargar el Div[1] que contiene la información deseada. Dedicar

La tecnología utilizada para la zona privada es la siguiente:
J2EE (Java 2 Enterprise Edition) se trata de una plataforma completa para construir aplicaciones Web basadas en el lenguaje Java. Todas las tecnologías que componente J2EE fueron desarrolladas por **Sun Microsystems** y otros vendedores de software entre los que se incluye BEA Systems.

Hay cuatro componentes que conforman el núcleo de una aplicación J2EE: los **Servlet**, las **JavaServer Pages** (JSP), los **Enterprise JavaBeans** (EJB) y **JDBC** (Java Database Connectivity).

Un concepto muy importante que se ha de tener en cuenta, es que J2EE resuelve el problema del coste y la complejidad del desarrollo de servicios multi-capa, basada en el patrón **Modelo-Vista-Controlador** (MVC).

Figura 3 Modelo-Vista-Controlador (MVC)

[1] Div: es un elemento HTML que permite asignar ciertos atributos a bloques de contenido.

La vista se compone de **JSPs**, interactúa con el usuario presentándole los datos y realizando todas las peticiones deseadas a la capa controlador. El controlador son los Servlets responsables de atender las peticiones de la vista e interactuar con el modelo. Los Servlets son quienes realizan la toma de decisiones de la aplicación. El modelo es la lógica del negocio y de los datos asociados de la aplicación. Los EJB se utilizan para construir componentes del modelo.

La **JSP** es una tecnología Java que permite generar contenido dinámico en forma de documentos Web estáticos. Contienen el componente de presentación y la lógica de procesamiento. El componente de presentación define el contenido que muestra el cliente. La lógica de procesamiento (javascript) define las reglas de negocio que se aplican cada vez que el cliente quiere enviar peticiones al servidor de aplicaciones.

Un **Servlet** es una clase Java que recibe peticiones que envía un cliente y devuelve información al mismo como respuesta. Haciendo de intermediario entre la JSP y el EJB.

Un **EJB** en un componente de la arquitectura J2EE cuya función principal es proporcionar la lógica de negocio de una aplicación. El tipo que se utiliza es el **beans de sesión** sin estado, no es necesario guardar ninguna relación entre las diferentes llamadas de un mismo cliente.

Se decidió utilizar EJB porque permite la **reusabilidad** de componentes e interoperabilidad, es decir se puede usar el mismo EJB en varias aplicaciones distintas. La arquitectura EJB hace posible que las aplicaciones se desplieguen de forma distribuida entre distintos servidores de una red; Y son mucho más sencillas de manejar (arrancar, parar, configurar, etc.) debido a que existen herramientas de control más elaboradas como ANT o Maven.

La **ventaja** de una aplicación J2EE es que no tiene por que ejecutarse dentro de una única máquina virtual java (JVM). Esto se debe a que una aplicación J2EE y sus componentes pueden llamar objetos ubicados en una JVM distinta mediante el sistema de invocación remota de métodos java (**RMI**). Los componentes de la capa controlador se comunican con la capa EJB mediante el API de RMI, implementando una interfaz remota.

La gestión con los datos del sistema se realiza por medio del API **JDBC**, proporcionando una conexión a la base de datos para luego poder consultar, insertar, modificar o eliminar datos almacenados en una base de datos. El controlador JDBC es independiente de la base de datos y de la plataforma utilizada.

En la aplicación también se utiliza el API **JavaMail** para el envío de correos electrónicos. Cuando un usuario de la red quiere ponerse en contacto con el administrador rellenando el formulario de contacta se envía un correo electrónico al administrador. También se envía un correo para recordar la contraseña al administrador. JavaMail permite que una aplicación cree

mensajes de E-mail y los envíe a través de un servidor SMTP (Simple Mail Transfer Protocol) a la red.

Una de las justificaciones de utilizar esta tecnología es porque es multiplataforma, se puede ejecutar en un entorno unix, mac o windows. El único requisito es tener instalado la maquina virtual de java (JVM), para interpretar el código en el momento de la ejecución. El administrador de la aplicación no se a de preocupar en ningún momento del sistema operativo que hay instalado en el ordenador.

Java es un lenguaje **orientado** a **red**; soporta protocolos TCP/IP, UDP, HTTP, FTP y RMI. El código desarrollado puede ser **reutilizado**, hay una simplificación de los procesos de desarrollo por lo que tiene un rápido mantenimiento debido a que son pequeñas unidades de código. J2EE es una combinación de varias tecnologías que permite unir los sistemas y servicios de los servidores, además de producir un entorno **escalable** y **robusto** en las aplicaciones Web. Esta escalabilidad la proporciona el uso de capas (patrón MVC).

Java es una tecnología orientada a **objetos,** permite reutilizar código, agilizando el desarrollo del software. Acercar el sistema al mundo real. Los objetos facilita el mantenimiento del software en caso de fallo del sistema.

Permite tener un sistema **distribuido**, ya que interesa separar el código en maquinas diferentes. Este interés es porque facilita el mantenimiento de la aplicación, en caso de fallos es más fácil encontrar el punto donde falla la aplicación.

Otra tecnología utilizada es **Log4j**, es una librería libre desarrollada en java por Apache Software Foundation. Es utilizado para monitorizar la aplicación en un archivo log. Permite elegir la salida y el nivel de granularidad de los mensajes a tiempo de ejecución y no a tiempo de compilación. La configuración de salida y granularidad de los mensajes se realiza en fichero logj4.properties para el controlador y un log4j_EJB.properties para el modelo. Dependiendo de la configuración que el programador desee puede habilitar y deshabilitar ciertos logs. Log4j es una forma elegante de sustituir los mensajes por consola, dejando de consumir recursos del sistema.

Log4j facilita el control de errores del sistema, en este proyecto se crean dos ficheros logs, uno para la parte del modelo (frumansaEJB.log) y otro para el controlador (frumansaServlet.log). Al crear dos ficheros logs, facilita todavía más encontrar los errores del sistema y saber exactamente donde falla. Los mensajes que se muestran en los ficheros logs son de información y de errores. Estos ficheros tienen un tamaño máximo 500kB, cuando se exceda este tamaño se creará un nuevo fichero log con otro nombre, con un máximo de 10 ficheros logs. Cuando el número de ficheros creados sean 10 se elimina el primer fichero creado y se crea uno nuevo.

En la aplicación también se ha optado por tener otros tres ficheros properties, en ellos se guarda toda la configuración de la aplicación con el fin de tenerlas separadas del código y modificarlas cuando se desee. Estos ficheros son:

- bbdd.properties: es donde está toda la configuración de la conexión a la base de datos necesaria para las clases.
- mail.properties: se guardan los datos necesarios que Javamail utiliza para el envió de correo electrónico, como la dirección de correo del administrador, su contraseña, el servidor para el envío de correo…
- conf.properties: El servidor de aplicaciones utiliza este fichero de configuración para localizar el resto de ficheros properties mencionados anterior mente. Ya que en conf.properties es donde se guarda las rutas de los demás ficheros properties.

Estos fichero Properties facilita el mantenimiento de la aplicación, porque si algún día es necesario cambiar el nombre de la base de datos o la dirección de correo del administrador o la ruta de los ficheros properties… no es necesario volver a modificar el código de las clases, compilarlo y empaquetarlo. Tan solo se tendría que modificar este fichero properties.

En el apartado 2 de los anexos se incluye más información sobre J2EE.

1.6. Herramientas y software base

Para el desarrollo de la aplicación, se han utilizado diferentes herramientas: El servidor de aplicaciones **BEA Weblogic 8.1.** Para el desarrollo UML el software **Together 6.1,** para el desarrollo del código java **eclipse 3.3**, para la base de datos se utiliza el software **MySQL-5** y para el diseño de la aplicación se han utilizado **Macromedia DreamWeaver 8** y **Agilator Flash Designe 7.**

En el apartado 3 de los anexos se incluye más información sobre este apartado.

CAPÍTULO 2. ARQUITECTURA

En este apartado, se comentan los diagramas de despliegue, arquitectónico, y de componentes. Estos diagramas sirven para mostrar los aspectos de la arquitectura física del sistema.

2.1. Arquitectura

En la figura 4 muestra la arquitectura distribuida de la aplicación y en la figura 5 se muestra aplicación local. Durante el desarrollo de la aplicación se han hecho todas las pruebas en local, pero está pensado que sea distribuido.

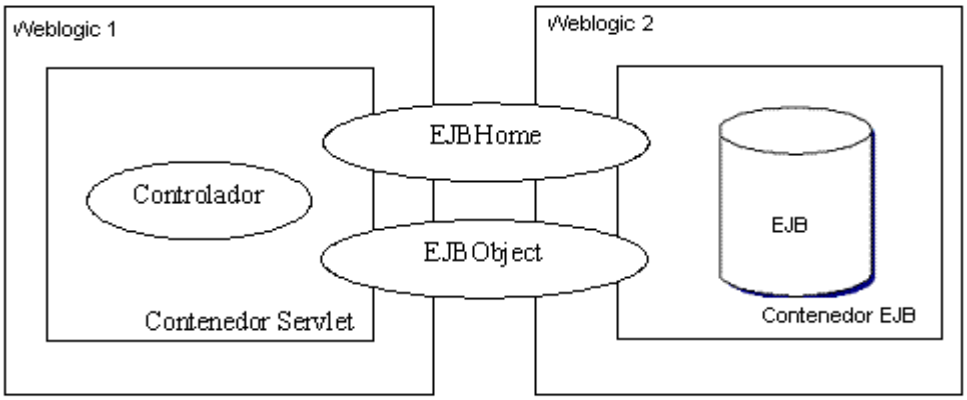

Figura 4 Arquitectura distribuida.

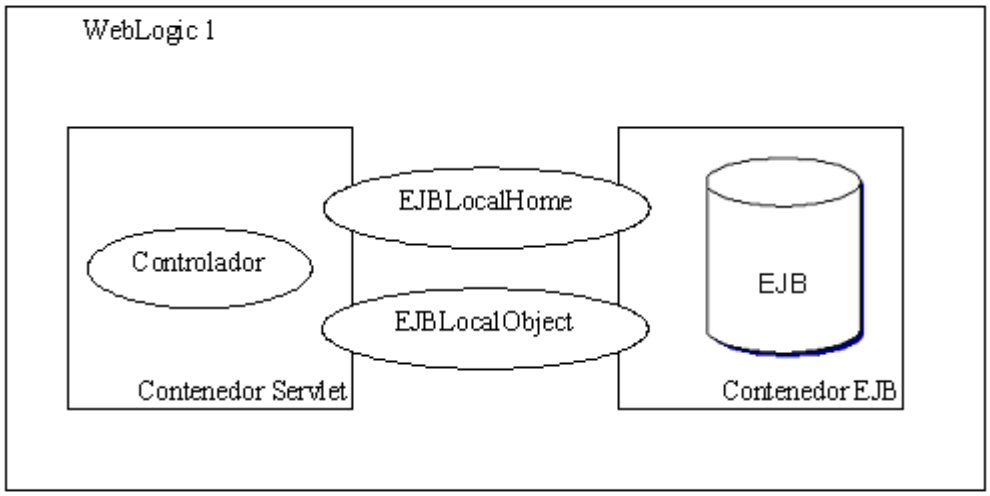

Figura 5 Arquitectura local.

Los motivos por los cuales se ha decidido que sea una arquitectura distribuida son los siguientes:

- **Funcional**: Cada ordenador tiene distinta funcionalidad.
- Es una arquitectura **escalable**, ya que se usa el patrón MVC.
- **Tolerante a fallos:** esta arquitectura permite la duplicidad por tanto si uno falla, los demás siguen funcionando.
- **Independiente** del sistema operativo instalados en los servidores.
- **Dispersos geográficamente**: se pueden tener los servidores separados a cientos de kilómetros.
- **Distribución** del trabajo: Menos carga para el servidor por lo que pueden ser menos potentes y más baratos.
- **Rápido mantenimiento**: Si hay un fallo en el sistema, al ser una arquitectura distribuida es más fácil localizarlo y repararlo.

2.2. Diagrama de componentes y despliegue

En el diagrama de despliegue (figura 6) se muestra las relaciones físicas entre los componentes software y hardware en el sistema final y la estructura del sistema en ejecución. Hay cuatro nodos. Un nodo es el PC del cliente, otros dos nodos son los servidores BEA WebLogic y el último nodo es el MYSQL, donde está instalada la base de datos del sistema. Los componentes WAR[2] y JAR[3] se ejecutan en su nodo BeaWeblogic correspondiente.

El WAR es un paquete que contiene los controladores y las vistas, y en el JAR los EJBs. La vista (JSP) se conecta mediante el protocolo HTTP al controlador, los EJBs son invocados de forma remota por el controlador. EL acceso a la base de datos se realiza mediante el protocolo JDBC. Como es una **arquitectura distribuida** permite colocar en una maquina el WAR y en otra el JAR, siempre y cuando haya conectividad RMI entre los 2 nodos.

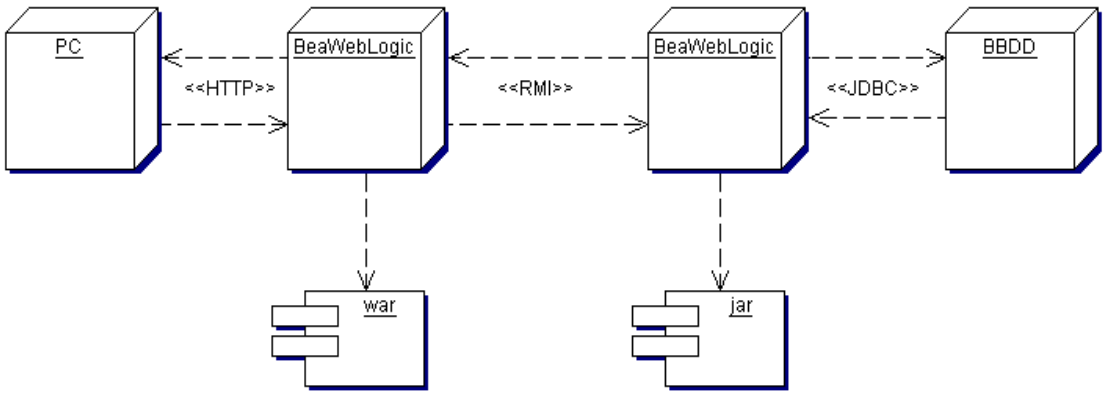

Figura 6 Diagrama de despliegue

[2] WAR: Es un paquete que permite agrupar un conjunto de clases y documentos que conforman una aplicación Web.
[3] JAR: Es un paquete que permite empaquetar clases de los EJB y los archivos relacionados para su despliegue.

Se ha decidido este diagrama ya que coincide con el despliegue de la aplicación, es decir, el usuario desde un PC con Internet (es el nodo PC) se conecta vía HTTP a la aplicación Web, como es distribuida, en el primer servidor de aplicaciones se hospeda el paquete WAR, este atiende las peticiones del usuario. BeaWeblogic1 invoca remotamente a los EJB (BeaWeblogic2) y este es quien se conecta a la base de datos mediante el protocolo JDBC y realiza las acciones solicitadas por el controlador.

RMI es un mecanismo ofrecido en Java para invocar un método remotamente con la independencia de JVM o el servidor en el que se encuentra, como si fuera un objeto local. El servidor BeaWeblogic ya tiene instalado todas las clases e interfaces necesarias para la conexión RMI.

La ventaja de usar RMI es que permite distribuir una aplicación de forma muy transparente, es decir, sin que el programador tenga que modificar apenas el código. Además las invocaciones remotas son más eficientes que las peticiones vía HTTP.

En la figura 7 se describe el diseño arquitectónico, dicho diseño está formado por cuatro subsistemas:

- El **subsistema gestor vista**, es la parte que se mostrada al usuario.
- El **subsistema controlador** es el encargado de la toma de decisiones de las peticiones de la vista decidiendo que hacer con ellas.
- El **subsistema modelo** realiza la gestión con la base de datos.
- Y por último, el **subsistema utilidades** se encarga de escribir ficheros Log en el sistema operativo.

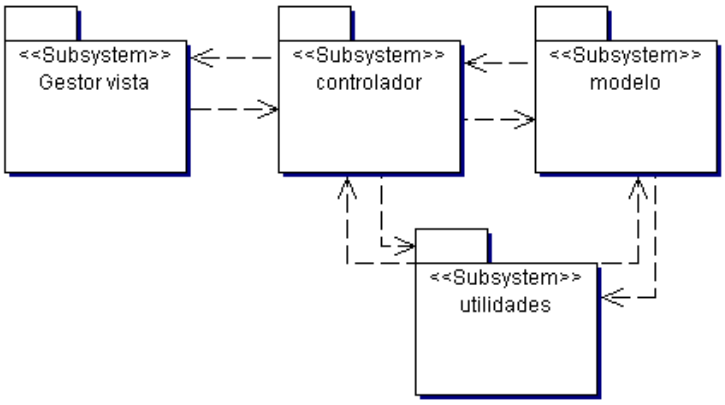

Figura 7 Diagrama arquitectónico.

Hay cuatro subsistemas porque antes de empezar a programar la aplicación Web, se decidió utilizar el patrón modelo-vista-controlador, por tanto cada uno de estos tiene su subsistema, a parte se pensó en el subsistema utilidades que es usado por los subsistemas controlador y modelo para escribir los ficheros Log e informar de las acciones realizadas.

En la figura 8 se muestra el **diagrama de componentes** de la aplicación. En este diagrama quedan reflejadas las interfaces y las relaciones entre los subsistemas descritos en la figura 7. Estos subsistemas están formados por los siguientes componentes:

- El **subsistema gestor vista**: está formado por el componente JSP**.**
- El **subsistema controlador:** está formado por dos componentes:
  - **Componente servlets:** recibe las peticiones de las JSP
  - **Componente instanciaEJBs:** instancia a las interfaces home y remote para conectar remotamente con el componente EJBs.
- El **subsistema modelo** está formado por el componente EJBs.
- Y por último, el **subsistema utilidades** está formado por el componente Log4j.

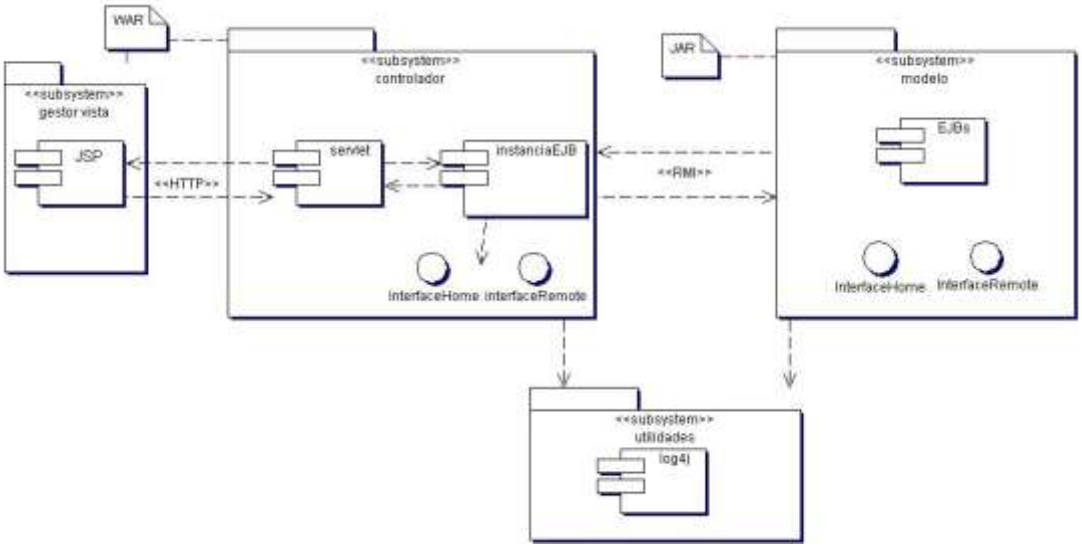

Figura 8 Diagrama de componentes.

En el subsistema controlador tiene estos dos componentes para separar el código de los Servlets de la instancia de los EJB, con esto se facilitó la programación y la detección de errores. Los subsistemas vista y modelo contienen estos componentes para seguir el patrón MVC, ya que la vista es la encargada de la presentación de los datos (JSP) y el modelo de la lógica de negocio (EJB). Por último, el subsistema utilidades contiene el componente Log4j porque es donde se escriben los ficheros Log.

CAPÍTULO 3. DISEÑO DE LA APLICACIÓN

3.1. Descripción del proyecto

El proyecto consiste en desarrollar la aplicación Web mencionada anteriormente. Hay dos partes claramente diferenciadas. Una parte es la zona de acceso público y otra de acceso privado.
La zona de acceso público permite al usuario navegar por la Web e informase sobre la empresa, los productos que tienen, ver un álbum de fotos de los productos, donde se ubica las fincas y un formulario de contacto.
La zona de acceso privado, es necesaria un nombre de usuario y una contraseña, previamente esta contraseña ya ha sido facilitada al administrador.
Una vez el administrador ha iniciado sesión, podrá gestionar el riego. Esta gestión de riego se especifica en los requisitos funcionales.
En la siguiente figura se muestra un mapa de la aplicación Web, diferenciando la zona pública y la privada.

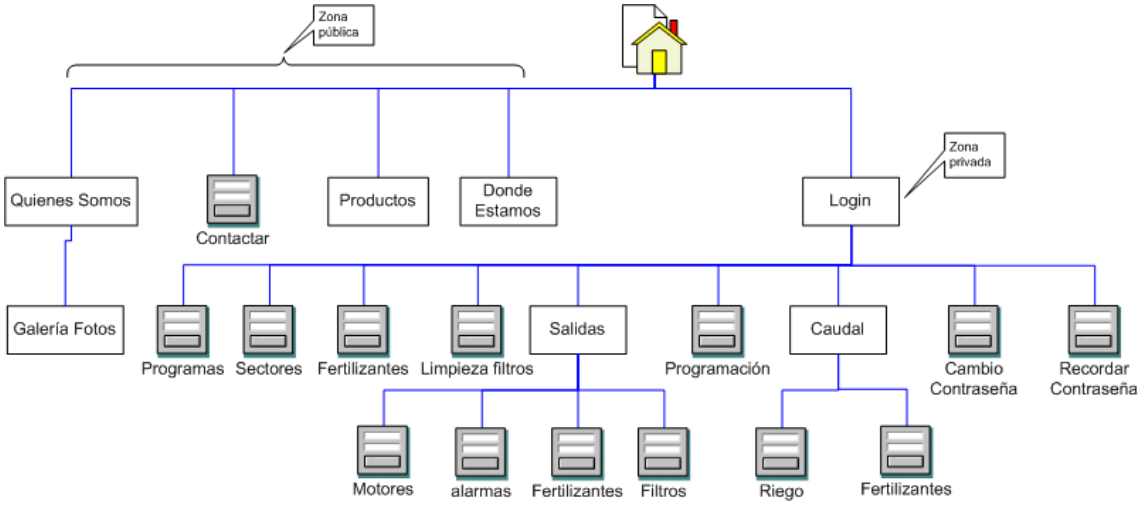

Figura 9 Mapa Web

3.2. Casos de uso.

El modelo de Casos de Uso cubre la especificación detallada de los requisitos del producto software. Se representa la forma en como un Actor opera con el sistema.

Para empezar se debe identificar los actores que interactúan en la aplicación. Hay dos actores, el actor usuario que puede acceder a toda la parte pública de la aplicación Web. Y otro actor es el administrador, como su nombre indica es quien administra la gestión de regadío, tiene permiso para acceder a la parte privada de la aplicación Web.

El primer diagrama de uso es la figura 10, se describe la interacción de los usuarios con el sistema de regadío. Se dividen en tres casos de usos: zona pública, gestión de regadío y gestión de administración. Estos casos de uso se explican más adelante.

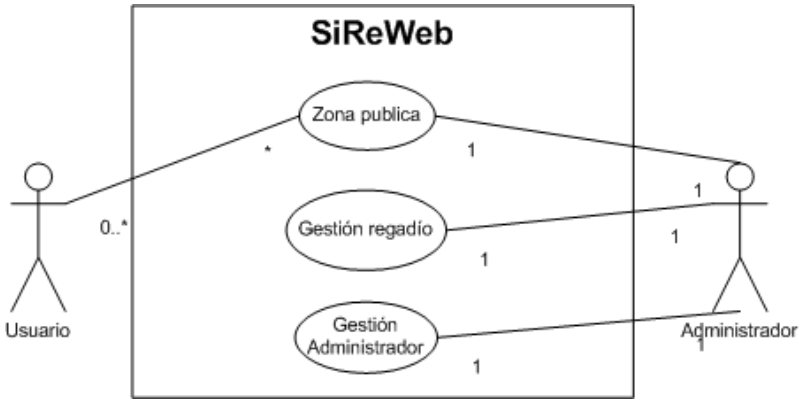

Figura 10 Caso uso Sistema Regadío Web (SiReWeb)

Un usuario cualquiera únicamente se le permite acceder a la zona pública. El administrador tiene acceso a toda la aplicación Web, a la zona pública, al la gestión de regadío y a la gestión del administrador.

En la figura 11 se puede observar el diagrama de caso de uso de la zona pública que se compone de los siguientes casos de uso: gestión contactar, ver información sobre la empresa, ver productos, ver mapa y ver fotos.

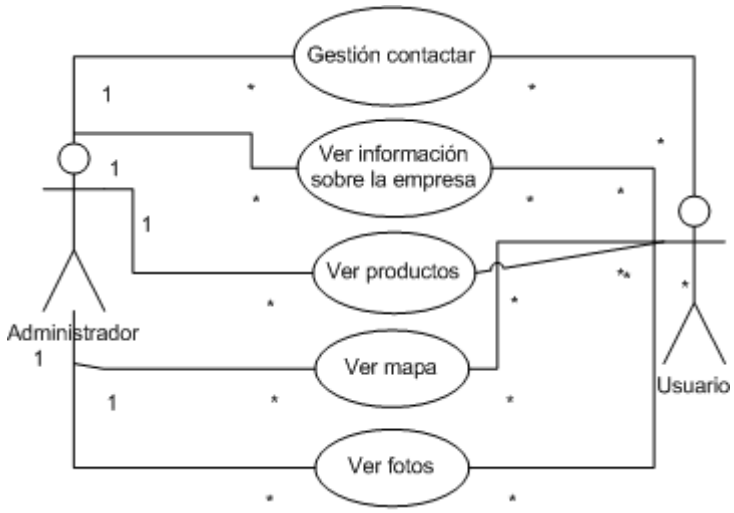

Figura 11 Caso uso zona pública

A continuación, en la figura 11 se muestra los casos de usos referentes al caso uso gestión administrador; hacen referencia al inicio y cierre de sesión del cliente. Si el cliente ha olvidado la contraseña, puede pedir que la aplicación le

recuerde la contraseña por correo electrónico. Una vez iniciada sesión, el administrador tiene la opción de cambiar su contraseña.

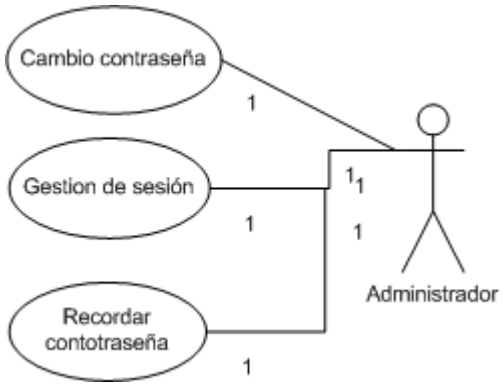

Figura 12 Caso uso gestión administrador

El diagrama de caso uso gestión de sesión, la figura 13, esta formado por varios casos de usos, inicio de sesión y cierre de la sesión. Estas funcionalidades se dan cuando el administrador inicia sesión y la cierra. Estos casos de usos son bastante importantes ya que en el inicio de sesión se comprueba que los datos introducidos por el administrador sean correctos y no pueda acceda cualquier usuario de la red. El cierre de sesión asegura al administrador de que no pueda acceder otro usuario con su sesión.

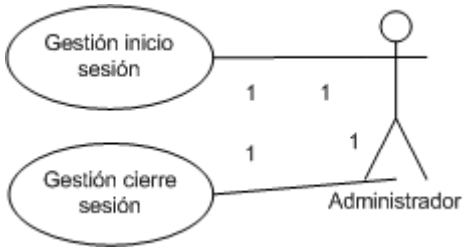

Figura 13 Caso uso gestión sesión

En el último diagrama uso, figura 14, se muestra el diagrama de uso gestión de regadío, con todas las funcionalidades descritas en el apartado 1.4. Como es una zona privada el administrador es el único actor que interactúa con este diagrama. Estos casos de uso son: Gestión de fertilizantes, gestión de limpieza de filtros, gestión de salidas de fertilizantes, gestión de salidas de alarmas, gestión de salidas de filtros, gestión de salidas de motores, gestión de caudal de riego, gestión de caudal de fertilizantes, gestión de sectores, gestión de programas y gestión de programación.

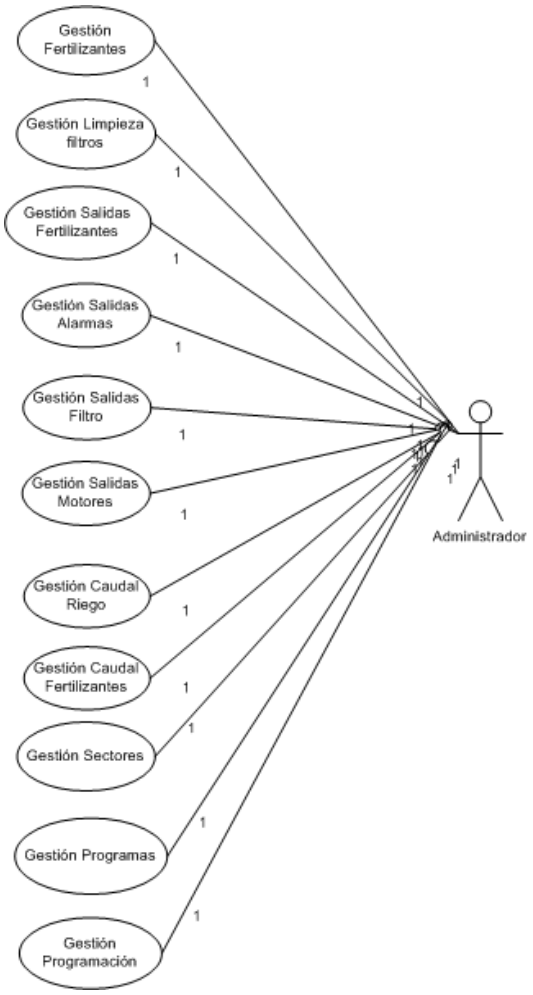

Figura 14 Caso uso gestión Regadío

3.3. Diagramas de clases

En este apartado se describe la estructura de la implementación del sistema mostrando sus clases con sus relaciones estructurales y de herencia. El modelo de casos de uso diseñados anteriormente aporta información para establecer las clases, objetos, atributos y operaciones.

En la figura 15 se muestran los paquetes de la aplicación. Se decidió dividir la aplicación en 5 paquetes para facilitar la programación de la aplicación y trabajar en diferentes capas. Son los siguientes paquetes:

- El paquete controlador es donde se encuentran todos los Servlets.
- El paquete instanciarEJB como su nombre indica se encarga de instarciar los EJB.
- El paquete utilidades se encuentra la clase encargada de cargar las configuraciones del sistema.
- En el paquete BBDD consta de dos paquetes:
    - El paquete cliente que es donde están las interfaces remotas y las interfaces Home.

- El paquete bean compuesto por los EJBs. El paquete instanciarEJB son las clases que instancia las interfaces de bbdd.cliente.

- Por último en el paquete datos están los objetos donde se guardan los todos los datos manipulados en la aplicación

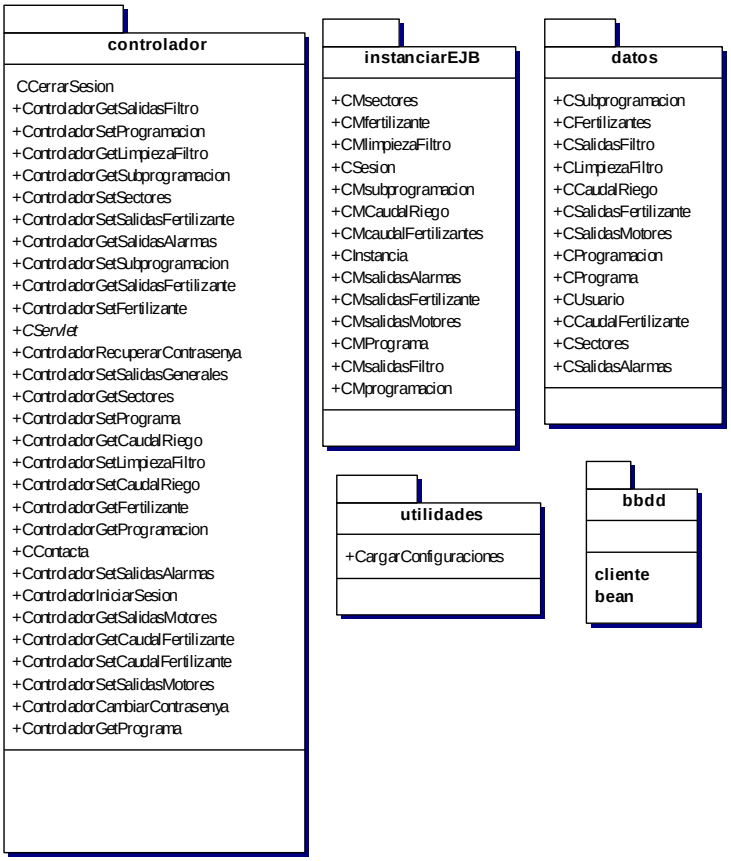

Figura 15 Paquetes del sistema

En las dos siguientes figuras, 16 y 17 se describen en detalle el paquete controlador. Todas las clases heredan de CServlet, es una clase abstracta que extiende de httpservlet. Esta clase contiene los métodos comunes para todos los controladores. Con esto se evita repetir código. La clase CServlet contiene los siguientes métodos:

- Método Init: es un método público, se llama desde el servidor de aplicaciones cada vez que se crea un Servlet. Se inicializa la configuración del los logs y se prepara el objeto de Logger para su uso.
- Los métodos doGet y doPost controlan la sesión. Estos dos métodos son prácticamente iguales pero se ejecuta uno u otro dependiendo del tipo de petición HTTP generada del navegador Web.
- Los métodos hacerPost y hacerGet son abstractos. Estos Métodos tiene que ser implementados por cualquier clase que extienda de Servlet.
- El método enviarDatosCorrectos se utiliza para confirmar al administrador que la inserción de datos en la base de datos se ha efectuado correctamente. Y si la inserción ha sido incorrecta se utiliza el método enviarDatosErroneos.

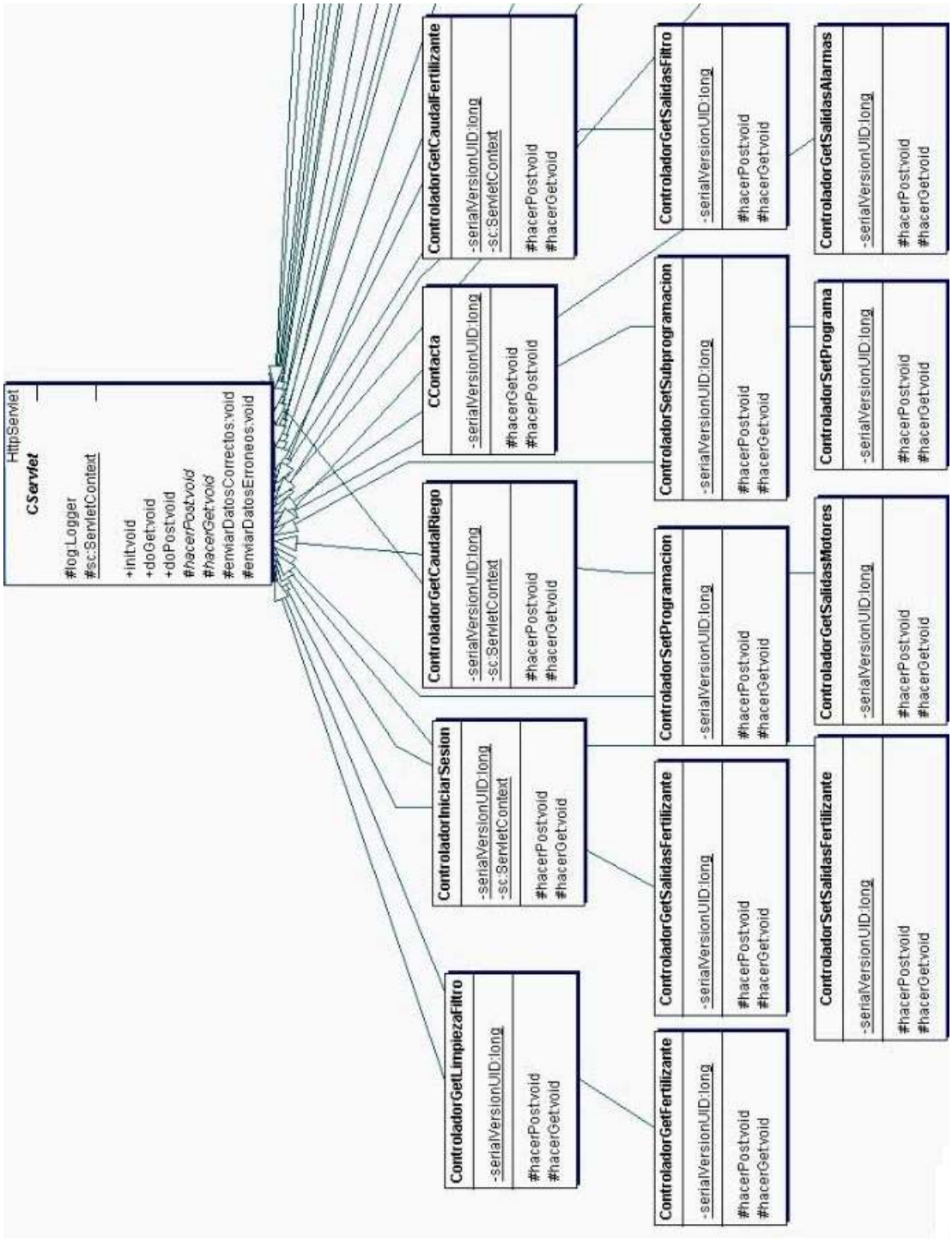

Figura 16 Paquete Controlador

En la figura 16, las clases que empiezan con el nombre ControladorGet se encargan de llamar a un EJB para consultar datos de la base de datos y devolverlos a la vista para mostrarlos al administrador.
Las clases que empiezan con el nombre ControladorSet se encargan recibir los datos de la vista de la gestión de regadío para guardarlos en la base de datos.
La clase CContacta recibe los datos de contacta.jsp y se encarga de enviar un correo al administrador con la información introducida en el formulario.

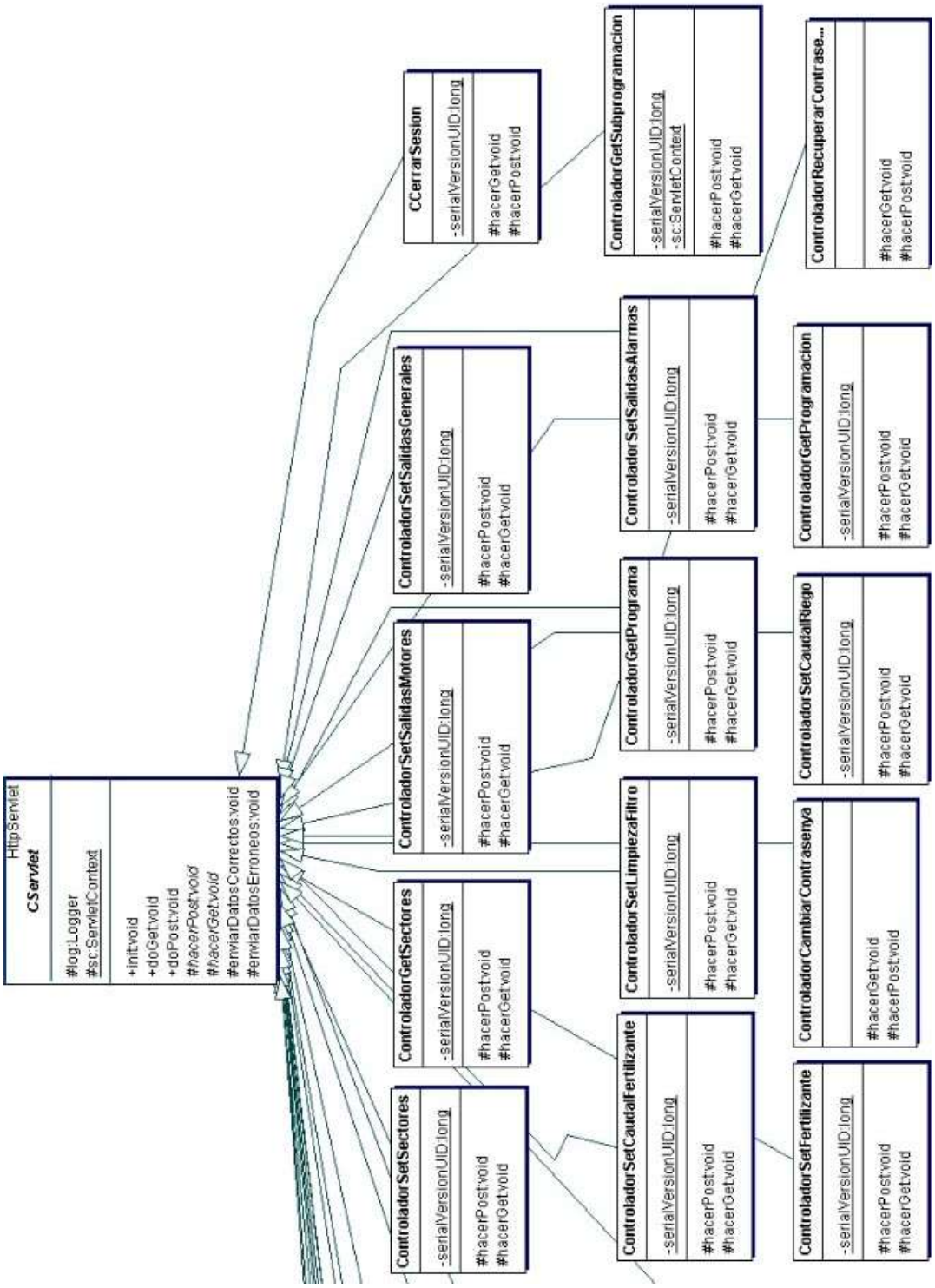

Figura 17 Paquete Controlador

En este diagrama, la figura 17, las clases que empiezan por ControladorGet y ControladorSet tienen el mismo significado que en la figura 16.
La clase ControladorRecuperarContraseña se encarga de hacer la petición al EJB perteneciente para recuperar la contraseña y el correo electrónico de la base de datos, y esta clase envíe un correo electrónico al administrador con la contraseña. La clase ControladorCambiarContraseña, como su nombre indica,

recibe la petición de cambio de contraseña. Y por último, la clase CCerrarSesion es la encargada de cerrar la sesión del administrador.

Casi todas las clases del paquete controlador tienen una instancia a las clases del paquete instanciarEJB, este paquete se muestra en la figura 18.

La clase de la que heredan todas las demás clase es Cinstancia. En esta clase se encuentra la parte común necesaria para instanciar un EJB.

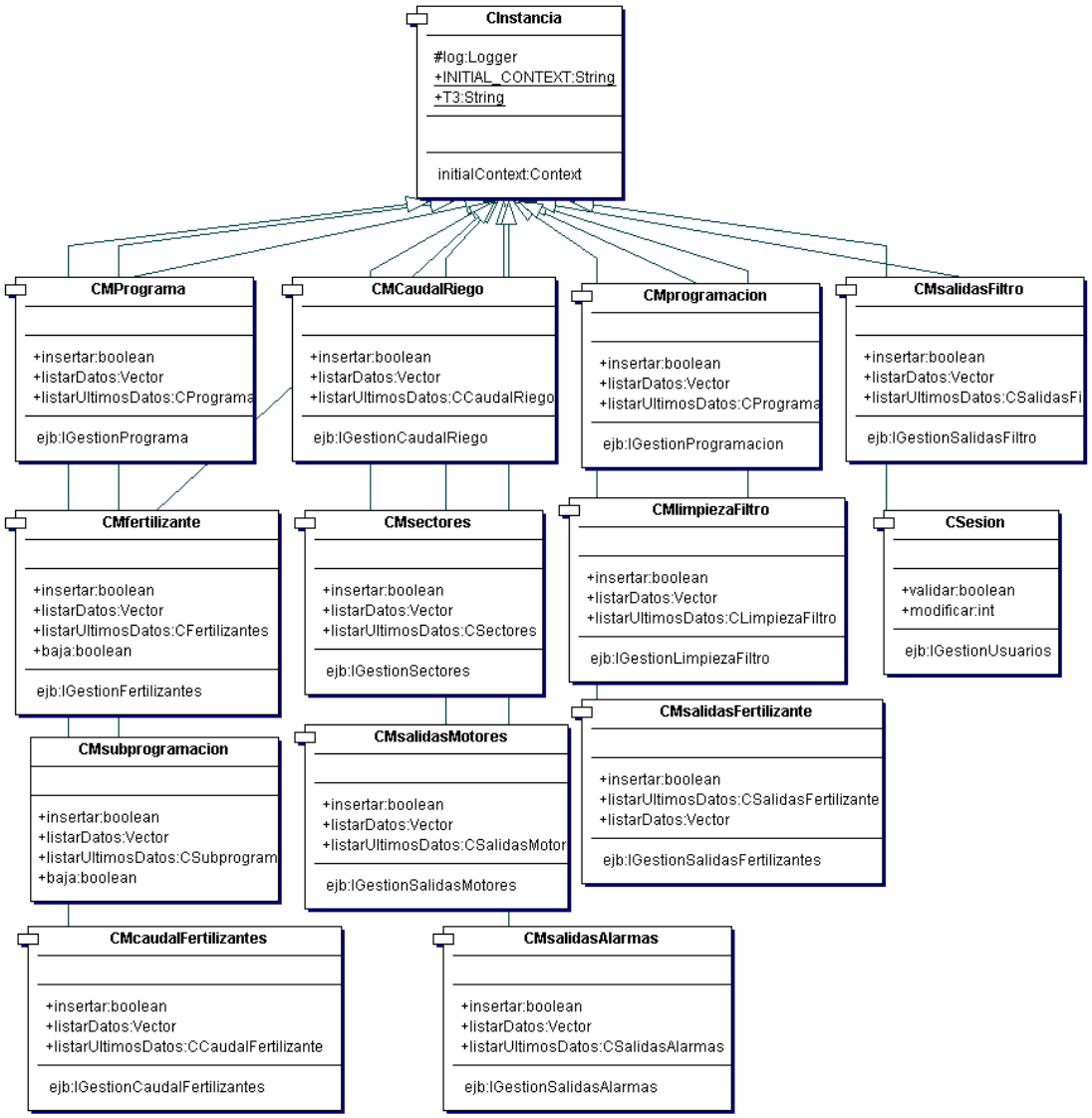

Figura 18 Paquete instanciarEJB

En la figura 19 se describe las interfaces de los EJBs. Hay dos tipos de interfaces: La Interface HOME nos permite crear las instancias de EJB de entidad o sesión a través del método create y la interfaz REMOTA especifica los métodos de instancia públicos encargados de realizar las operaciones.

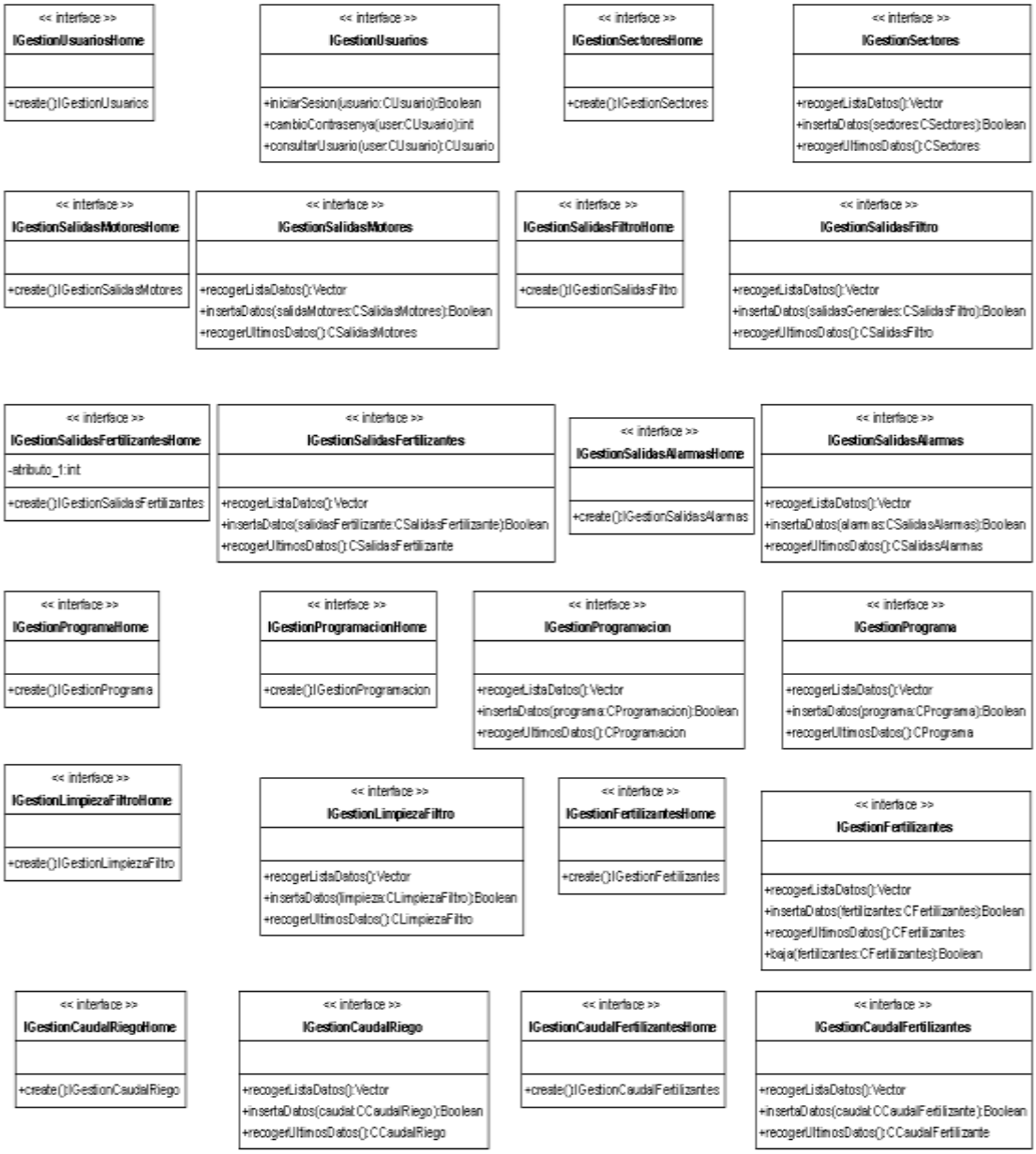

Figura 19 Paquete bbdd.cliente

En el diagrama de clases del paquete BBDD.Beans se haya los EJBs. Estos BEANs son la capa de acceso a la base de datos. En cada Bean se usa una instancia de la clase CConexionBaseDatos, como su nombre indica esta clase se usa para conseguir una conexión JDBC con la base de datos. Todas las clases de este paquete excepto CConexionBaseDatos heredan de CEJBComun. Esta clase tiene el método CREATE que se llama la primera vez que se instancia un EJB, en él se inicializa la configuración del los logs.

El diagrama de clases BBDD.Beans se muestra en el apartado 4.1 del anexo, porque es irrelevante ya que en la figura 19 se encuentran las interfaces encargadas de llamar a este paquete y contienen los mimos métodos.

A continuación, en la figura 20 se muestran los objetos de persistencia de datos, estas clases se usan para almacenar los datos procedentes de la base de datos y los datos introducidos por el usuario.

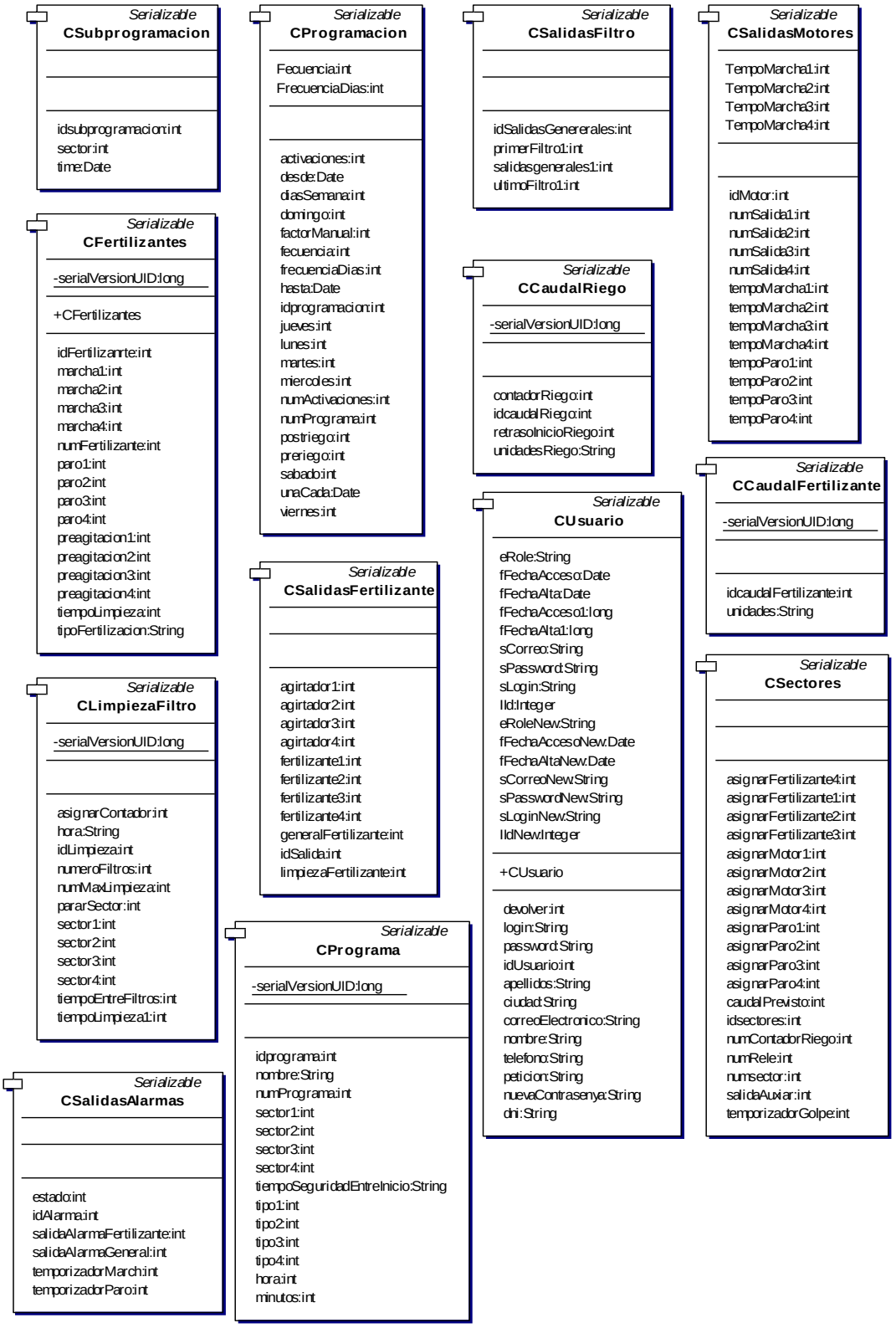

Figura 20 Paquete datos.

Y por último, en la figura 21 se representa la clase usada para la carga de los archivos de configuración. Dado que la configuración es común para toda la aplicación, esta clase implementa el patrón de diseño Singleton (instancia única), consiste en garantizar que la clase sólo tenga una instancia y proporcionar un punto de acceso global a ella.

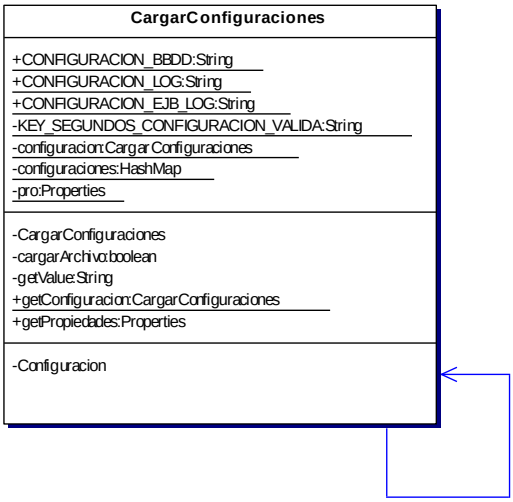

Figura 21 Paquete utilidades.

3.4. Diagramas de secuencia

Se describe un diagrama de secuencia por cada caso de uso o por cada grupo de caso de uso. Los diagramas de clases y el mapa Web ayudan a modelar las distintas secuencias de acciones descritas en los casos de uso como interacciones entre objetos, que envían y reciben mensajes para implementar las funcionalidades requeridas del sistema.

El siguiente diagrama de secuencia (figura 22) pertenece al caso de uso público. El usuario o administrador navegan por la Web, se muestran las llamadas javascript Ajaxpage que realizan cambios sobre las páginas sin necesidad de recargarlas.

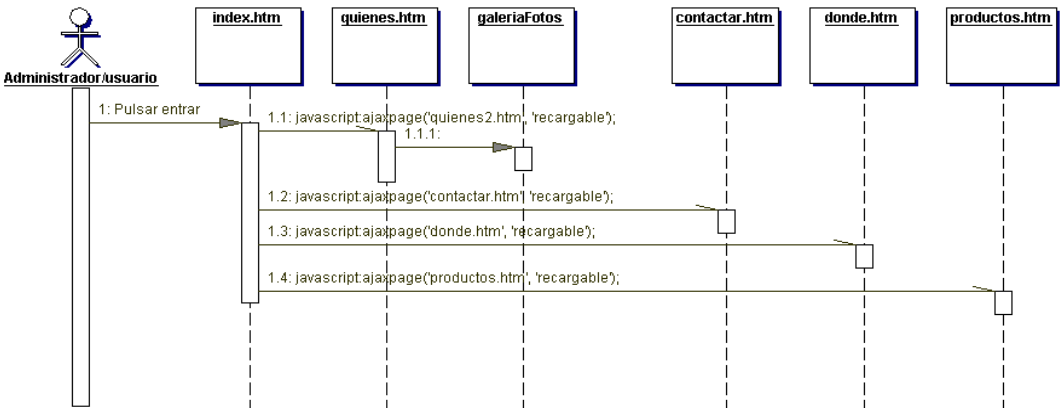

Figura 22 Diagrama secuencia público.

El caso de uso gestión contactar tiene el siguiente diagrama de secuencia, el usuario quiere ponerse en contacto con el administrador rellenando el formulario que aparece en la pagina contactar.htm, una vez que el usuario pulsa enviar se envía un correo electrónico al administrador con los datos detallados en el formulario.

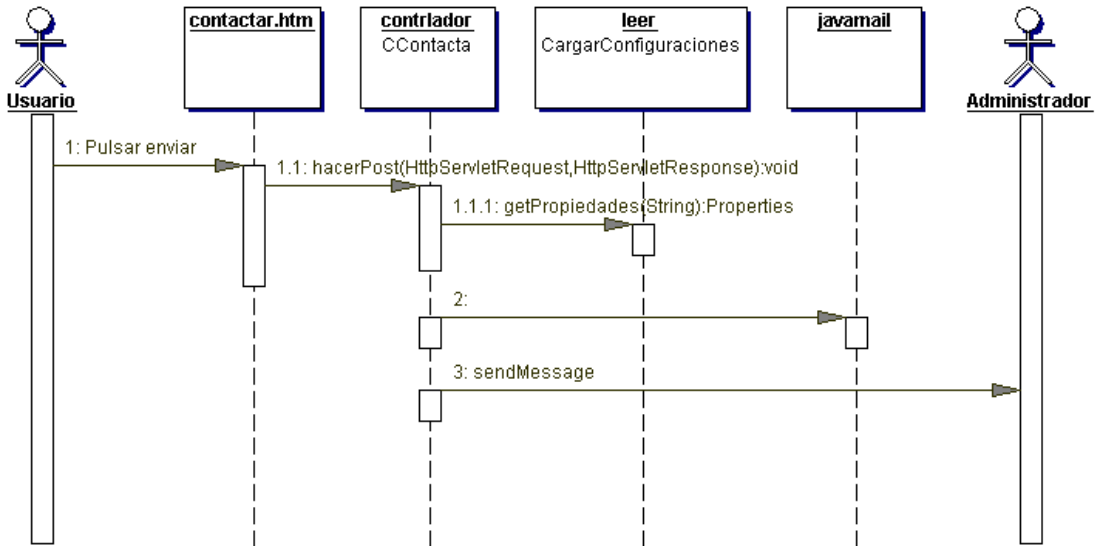

Figura 23 Diagrama secuencia contactar.

Inicio de sesión pertenece a la figura 24, el administrador introduce los datos en el formulario login.jsp, envía los datos al Servlet y este al EJB, compara los datos del administrador registrado en la base de datos, si son correctos inicia sesión y puede entrar la gestión de regadío. Si no son correctos, se le muestra la página de login.jsp para que vuelva introducir los datos de inicio de sesión.

En la figura 25 se muestra el diagrama de secuencia de cambio de contraseña del administrador, el formulario que debe de rellenar es el de cambiocontrasenya.jsp con la contraseña antigua y la nueva contraseña, el servlet controladorCambiarContrasenya recoge los datos, se los envía a la EJB, si la contraseña introducida es la misma que la que hay guardada en la base de datos los cambios se realizarán satisfactoriamente.

Si el administrador ha olvidado su contraseña para iniciar sesión, puede solicitar que la aplicación le recuerde la contraseña, esto se muestra en el diagrama de secuencia 26, la aplicación le envía un correo electrónico con la contraseña.

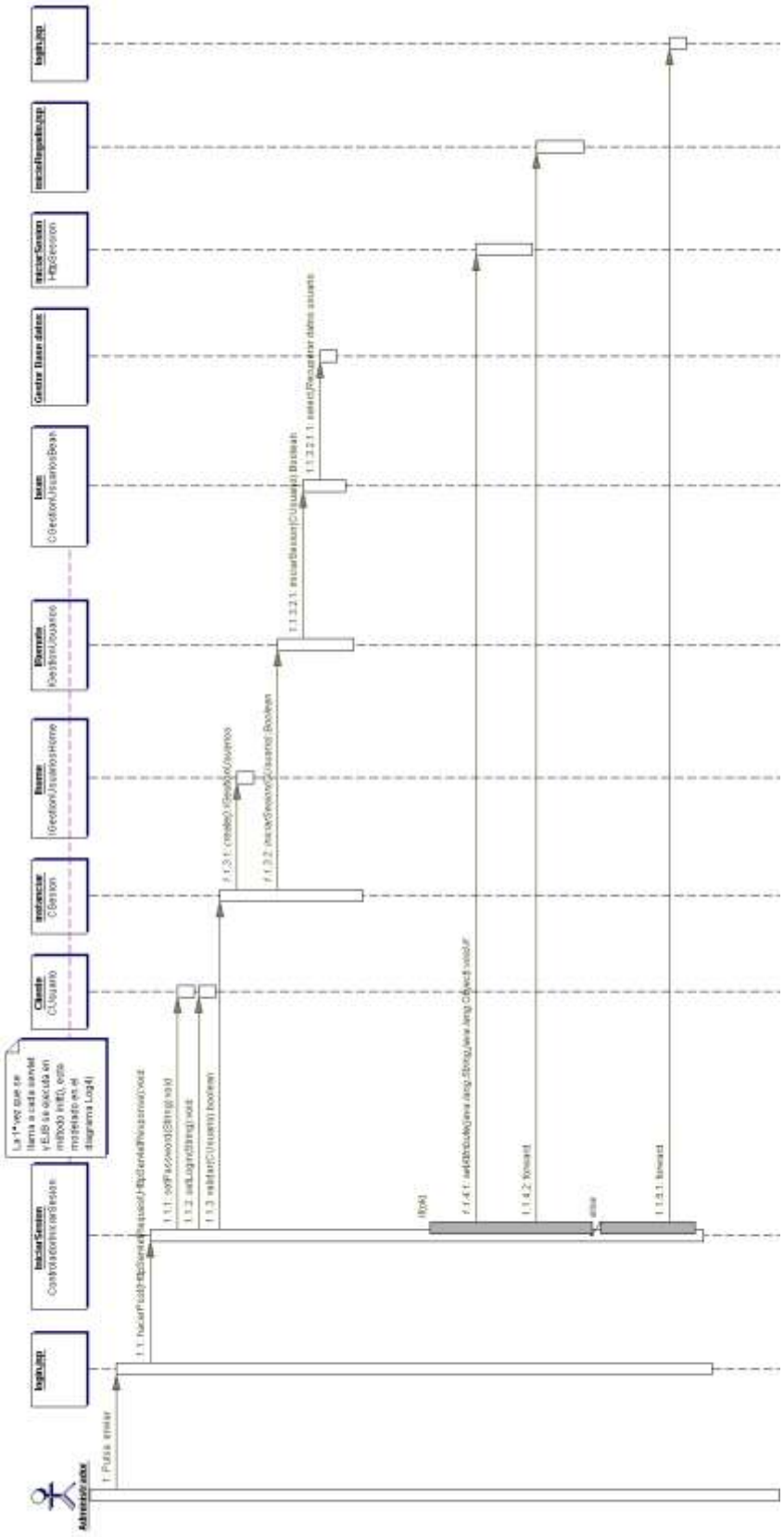

Figura 24 Diagrama secuencia inicio de sesión.

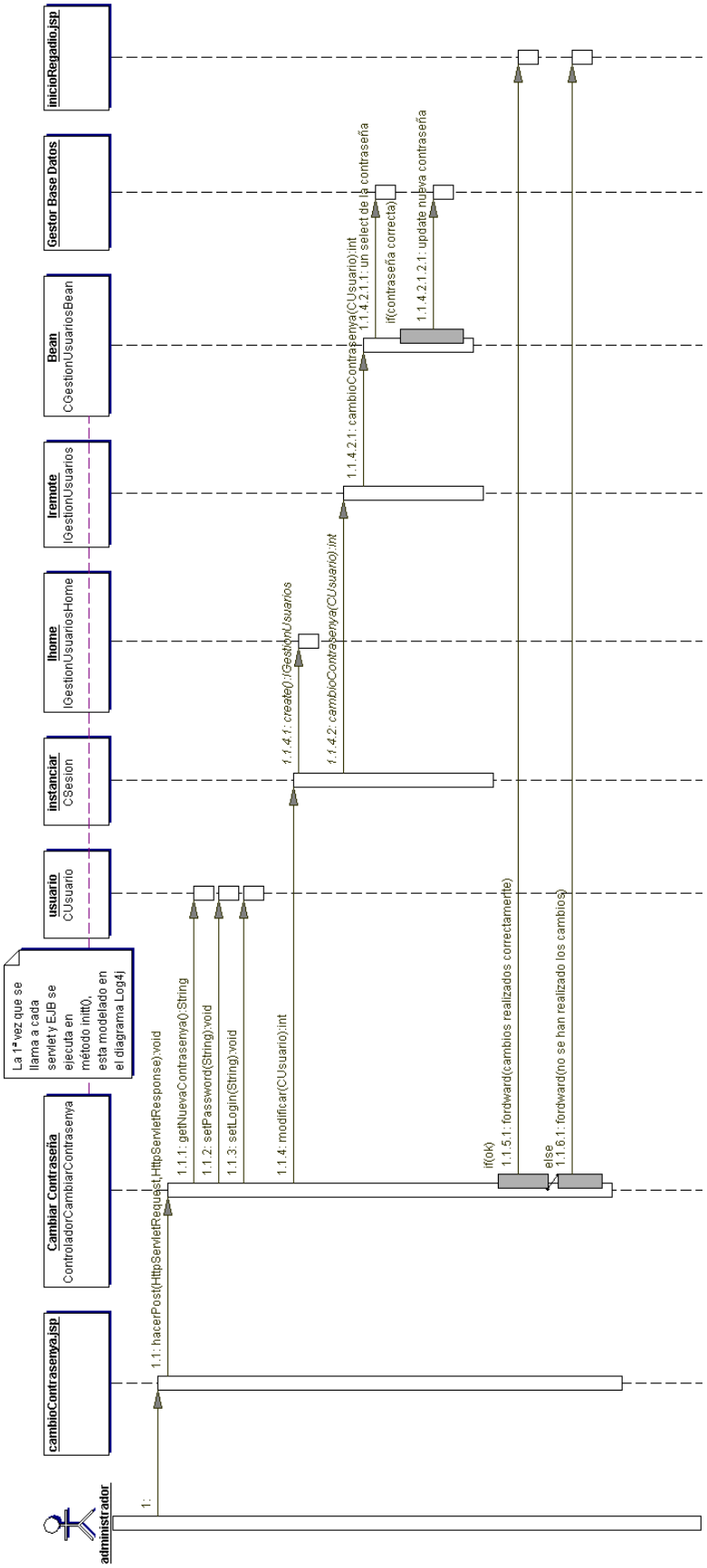

Figura 25 Diagrama secuencia cambio contraseña

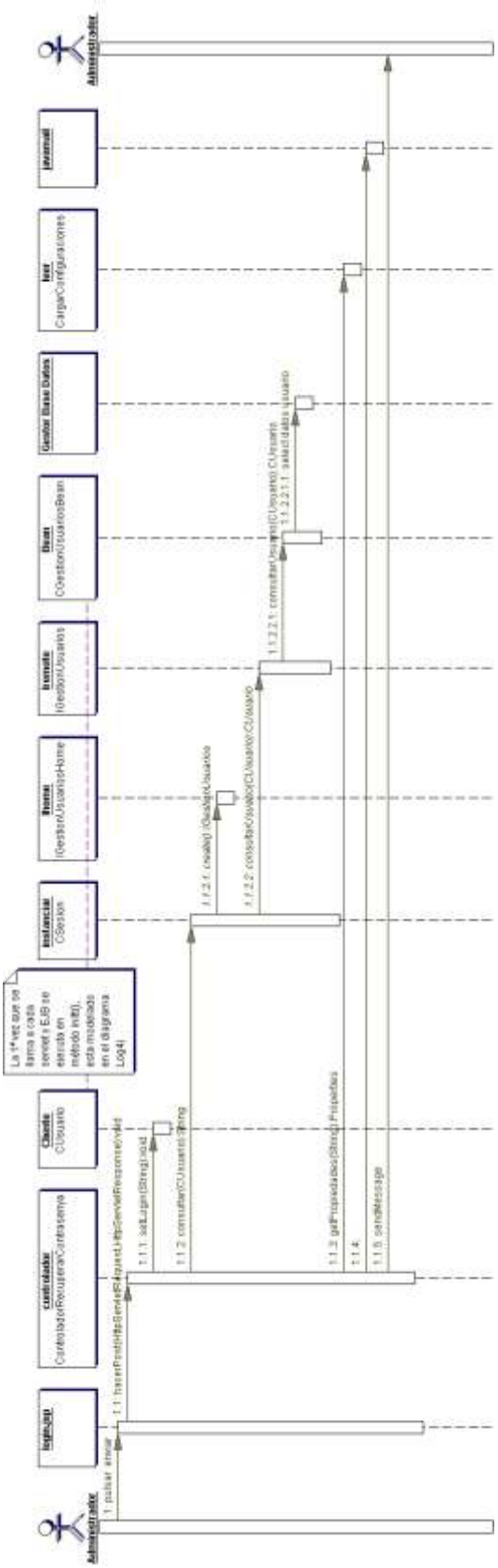

Figura 26 Diagrama secuencia recuperar contraseña.

Durante la gestión de regadío la sesión está inicializada, el administrador puede cerrarla cuando él lo desee. Hay un botón cerrar sesión, que pulsando sobre él invalida la sesión y seguidamente muestra la pagina login.jsp. Esto se muestra en la figura 27.

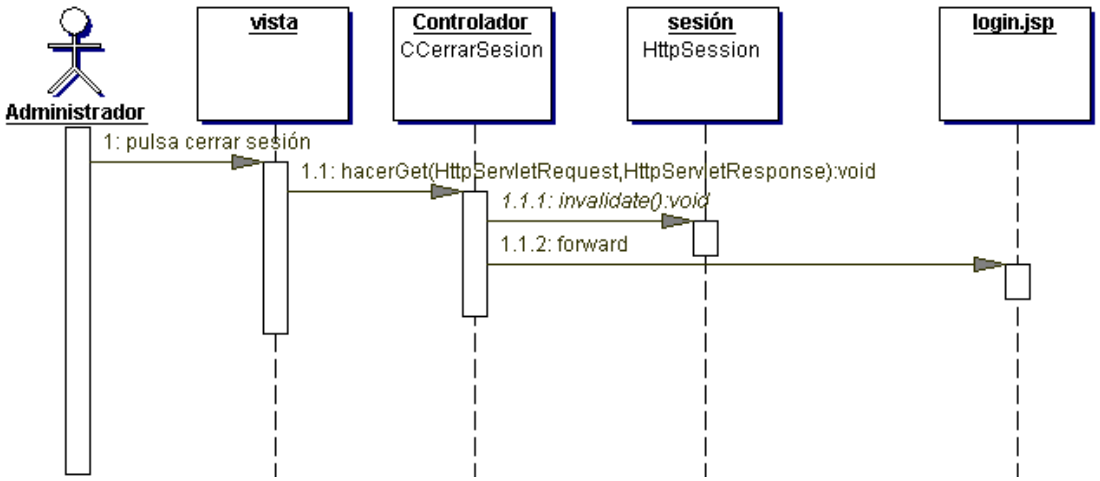

Figura 27 Diagrama secuencia cerrar sesión.

Por cada caso de uso de gestión de regadío hay dos diagramas de secuencia. Un diagrama donde se refleja la consulta de los datos para mostrarlos al usuario (get) y otro para insertar los datos (set). Dado que los procesos de lectura de base datos (ejemplo: getfertilizantes) y los de escritura (ejemplo: setfertilizante) son prácticamente idénticos para todos los casos de usos de gestión de regadío, con la diferencia de que cambian los nombres de las clases y objetos pero siguen la misma estructura. Solo se incluye un diagrama de secuencia por cada tipo (get y set).

En el apartador 4.2 de los anexos se han incluido estos diagramas de secuencia que no se muestran en este apartado.

En la figura 28 se muestra el diagrama de secuencia getFertilizantes, se recogen los datos de fertilizantes de la base de datos y a continuación se muestran al administrador. Si desea modificar los datos, sólo con cambiar los valores del formulario y pulsando enviar, estos datos serán introducidos en la base de datos, en el diagrama setFertilizantes, figura 29, se puede ver la descripción completa.

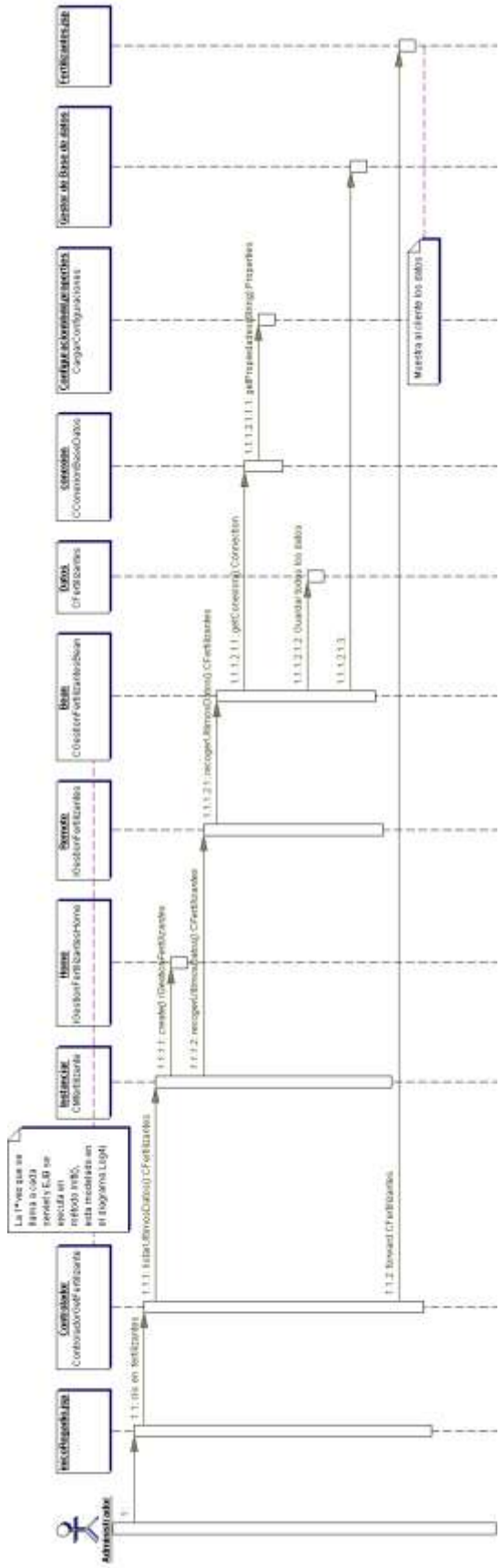

Figura 28 Diagrama secuencia getFertilizantes.

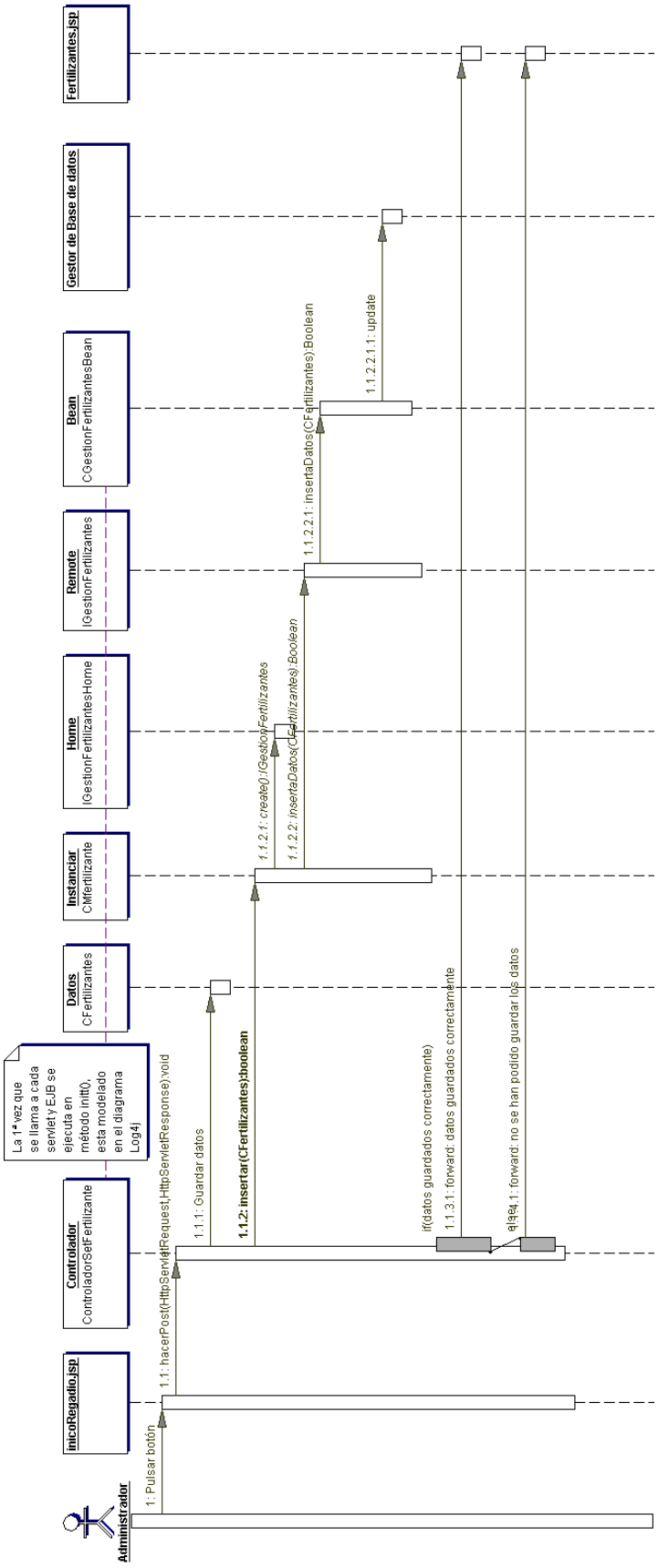

Figura 29 Diagrama secuencia setFertilizantes.

La primera vez que se invoca un servlet y un EJB, se implementa el método init en los Servlets y el CREATE en los EJB, inicializando la configuración del Log4j, mediante la clase cargarConfiguraciones. Esto se muestra en la figura 30 para los Servlets y la figura 31 para los EJB

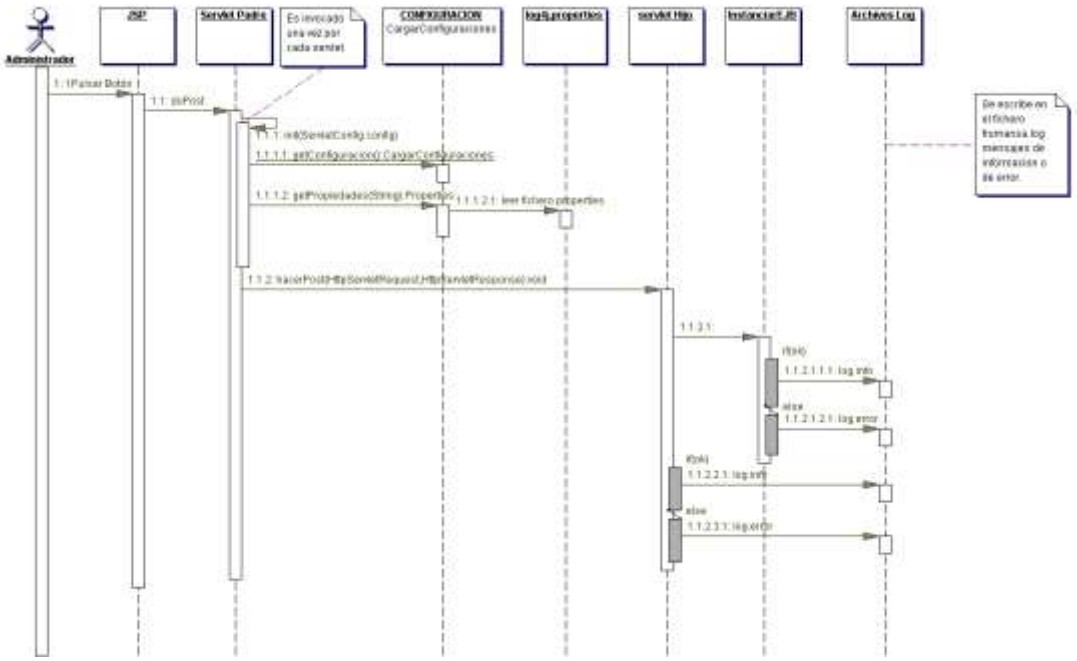

Figura 30 Diagrama secuencia log4jServlet.

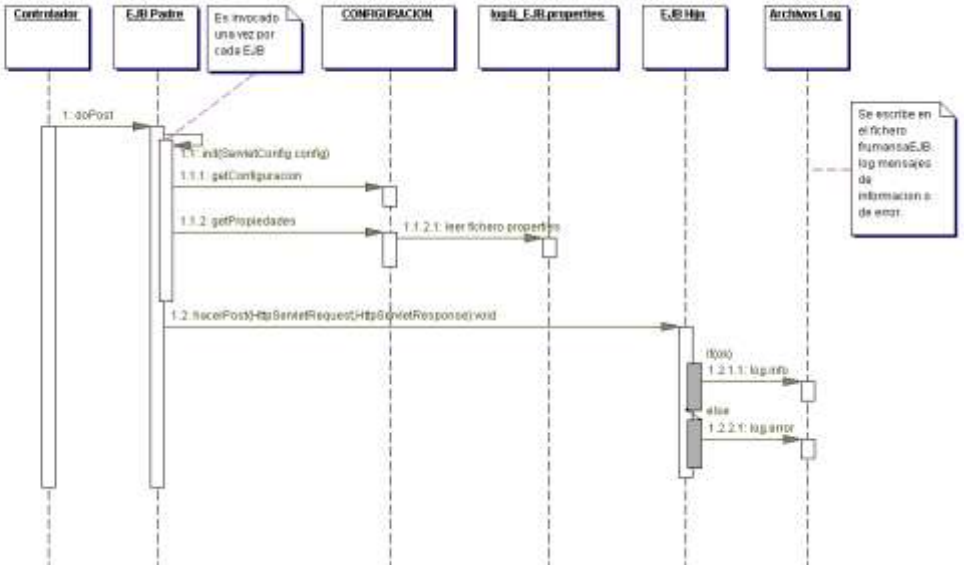

Figura 31 Diagrama secuencia log4jEJB.

3.5. Base de datos

En la base de datos se encuentran las siguientes tablas:

- Tabla usuario: En ella se guarda los datos relacionados con el administrador.
- Tabla Sectores: En esta tabla se asigna el motor, el caudal previsto, la entrada de paro y el fertilizante a un número de sector.
- Tabla fertilizante: En esta tabla se define el número de fertilizantes que tiene el cabezal de riego, por cada fertilizante hay datos relacionados sobre el tiempo que esta en marcha y el tiempo que está parado, también tiene relacionado un agitador con tiempo de preagitación. Además se puede guarda un tiempo de limpieza final de fertilizantes.
- Tabla limpieza de filtros: Se guarda en número de filtros que tiene el cabezal de riego, el tiempo de limpieza y la hora que se procederá hacer esta limpieza, además guarda el tiempo de paro entre la limpieza de cada uno de los filtros. También se guarda la información de si se ha de parar el riego mientras se efectúa la limpieza.
- Hay cuatro tablas que se refieren a las salidas del sistema de regadío, son las siguientes:
    - Tabla salida de las alarmas: En esta tabla se almacena el estado de las alarmas en reposo (abiertas/cerradas). Se guarda el temporizador en marcha de la alarma, el temporizador en paro, la salida de alarma general y la salida de alarma fertilizante.
    - Tabla salida de fertilizantes: Guarda que fertilizante se inyectará al sector.
    - Tabla salida de motores: Se almacena la configuración de salida de cada motor, con el número de salida con el tiempo de marcha y paro del motor.
    - Tabla salida generales: Se guarda que salida general del cabezal de riego se desea asignar y por el filtro que va a pasar.
- Gestionar el caudal del agua para minimizar su consumo. Para el control de este consumo hay dos tablas relacionadas:
    - Tabla caudal fertilizante: Guarda la unidad en que se va medir el riego.
    - Tabla caudal de riego: Almacena la unidad de riego, el contador de riego que se le asigna para contabilizar la cantidad de agua suministrada y el tiempo de retraso al inicio de riego
- Tabla programa: Se guarda el nombre y número de programa con los condicionantes y sensores que se desee asignar.
- Tabla programación: En ella se almacena toda la programación que se aplica en este momento con referencia con las tablas anteriores, a parte se almacena las horas y fecha que se activara el riego.

Hay que tener en cuenta que esta base de datos ha sido creada para simular el ordenador de riego Agronic 4000. En la figura 32 se muestra el diagrama de la base de datos.

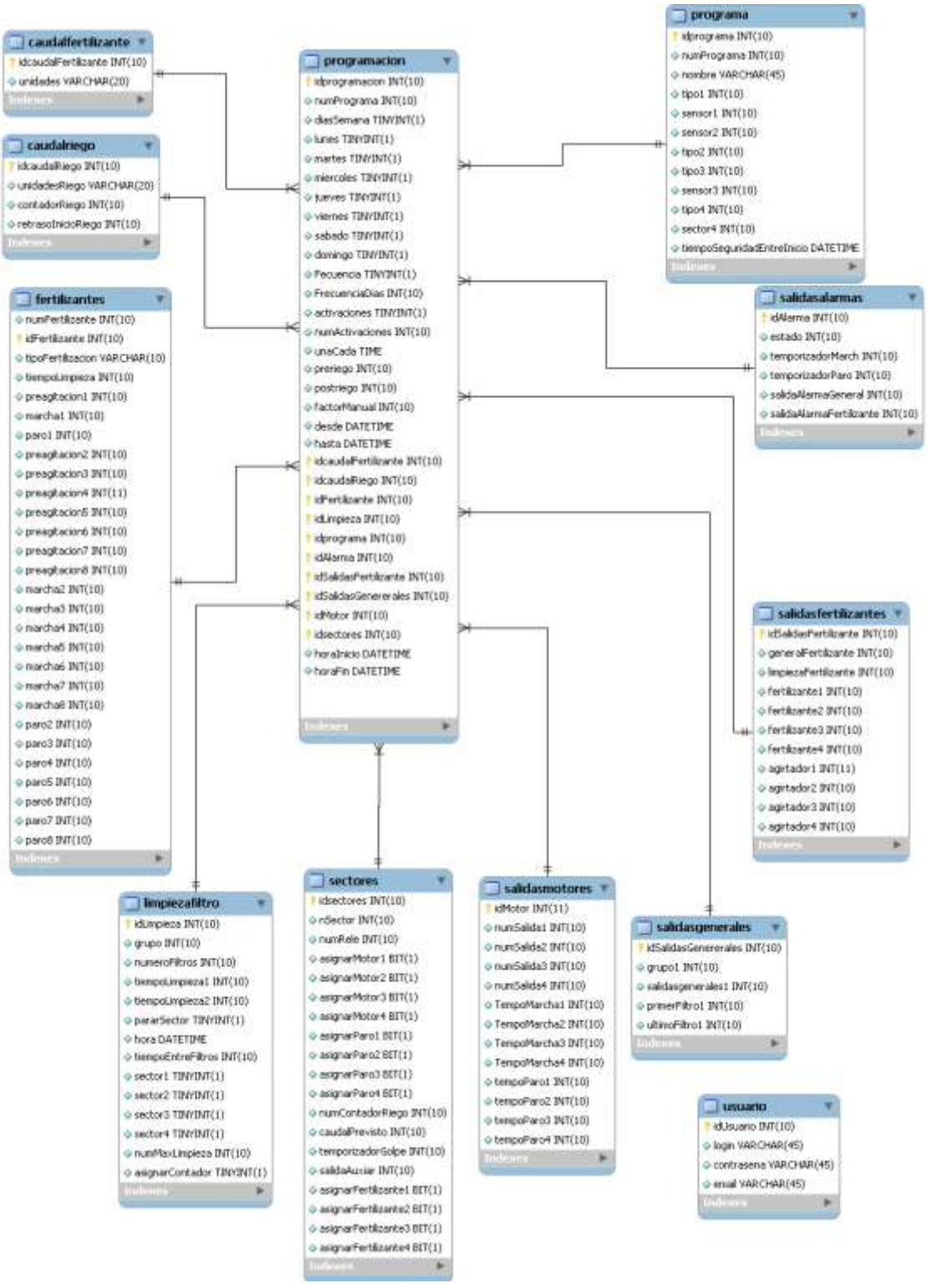

Figura 32 Diagrama base de datos

CAPÍTULO 4. IMPLEMENTACIÓN

Para implementar la aplicación y teniendo en cuenta lo que se ha explicado anteriormente, se han desarrollado diferentes pantallas que se muestran en la implementación visual. Además se han generado unos paquetes de instalación que se explican con detalle en la implementación de los paquetes WAR y JAR.

4.1. Implementación visual

En este apartado se muestran unas capturas de pantalla de la aplicación Web. Se diferencian dos partes en la aplicación Web: zona pública y zona privada.

4.1.1. Zona pública

La primera página que se muestra al usuario es la figura 33, es la página de inicio de la aplicación. Una vez que el flash se ha cargado, se puede pulsar a "entrar", si no se quiere esperar a que se cargue, se puede pulsar sobre "saltar intro".

Figura 33 Página de inicio

Seguidamente se muestra la figura 34, el contenido de esta página es el quienes somos. En el menú superior hay diferentes enlaces donde el usuario puede acceder.

Figura 34 Quienes somos

Cuando el usuario pulsa en el enlace “Contactar” situado en el menú superior; se muestra la figura 35. Es un formulario para contactar con el administrador. La aplicación se encarga de enviar un correo al administrador con los datos introducidos y comentarios introducidos.

Figura 35 Formulario contacta

Hay figuras de la zona pública que son irrelevantes y no se muestran en este apartado, pero se pueden encontrar en el apartado 5 de los anexos.

4.1.2. Zona privada

En la zona privada solo tiene acceso el administrador de la aplicación de regadío. Para acceder a esta zona, se ha pulsar sobre el enlace “Login”, aparece la página de la figura 36, en este caso el administrador debe rellenar este formulario con el usuario y contraseña asignados previamente.

Figura 36 Formulario Login

Si el administrador no recuerda la contraseña, pulsando sobre el enlace “No puedo acceder a mi cuenta”, se muestra la figura 37. Introduciendo el usuario y pulsando enviar, se envía un correo electrónico al administrador con la contraseña.

Figura 37 Formulario recordar contraseña

Si el administrador ha introducido bien el usuario y la contraseña, ha entrado a la zona privada, al gestor de regadío (figura 38). En esta zona siempre está el menú superior de “Gestor de Regadío”, “Cambiar contraseña” y “Cerrar sesión”. El menú que hay en izquierda de la figura 38 “Parámetros”, es para la gestión de regadío y la “Programación”.

Figura 38 Gestor de Regadío

Cuando el administrador pulsa sobre “Fertilizantes” se muestra los parámetros a configurar (figura 39), se muestran los que se introdujeron la última vez que se configuraron, con la posibilidad de modificarlos.

Figura 39 Formulario gestión Fertilizantes

Si el administrador pulsa sobre el boton “Limpieza filtros”, aparece la figura 40.

Figura 40 Formulario gestión limpieza filtros

En el menú de parámetros, si se pulsa sobre salidas, el menú se despliega 4 submenus: salidas de alarmas se puede ver en la figura 41. Salidas de fertilizantes, salidas de filtros y salida de motores

Figura 41 Formulario salida de alarmas

Pulsando sobre “Caudal” del menú de la izquierda, también se despliega en dos submenús, gestión de caudal de riego (figura 42) y de caudal de fertilizantes.

Figura 42 Formulario caudal de Riego

Para la gestión de sectores se muestra la figura 43.

Figura 43 Formulario gestión de Sectores

Si se pulsa en el menú parámetros “programa” se puede configurar el programa con todos los parámetros guardados anteriormente modificados. En la figura 44 se muestra esta configuración.

Figura 44 Formulario parámetros programas

La programación de la aplicación se puede configurar que fechas, que horas y días de la semana se apliquen los parámetros guardados previamente en la configuración del regadío. Siempre se muestra la última programación guardada, con la posibilidad de modificarla a conveniencia del administrador. Se accede a ella pulsando sobre el botón “Configuración” del menú de la izquierda. Se muestra en la figura 45.

Figura 45 Formulario configuración programación

Por último, el administrador puede cambiar la contraseña, pulsando en "cambiar contraseña" que aparece en el menú de la parte superior, la pantalla que se muestra es la figura siguiente:

Figura 46 Formulario cambio de contraseña

En el apartado 5 de los anexos se incluyen todas las capturas de la aplicación Web.

4.2. Implementación de los paquetes WAR y JAR

Al ser un sistema distribuido, la aplicación se ha empaquetado en un **WAR** y en un **JAR** (figura 46). EL motivo por el cual van en paquetes separados, es porque se pueden instalar en servidores de aplicaciones distintos, es decir en distintas máquinas. Al tener dos paquetes, hay independencia de fallos de la aplicación.

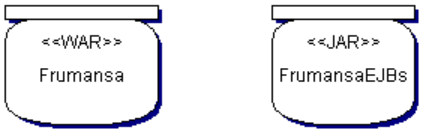

Figura 47 Empaquetado de la aplicación.

Cada archivo puede estar ubicado en un mismo servidor o en diferentes. Su formato es un archivo de compresión para facilitar su transporte. Si los archivos se ubican en servidores distintos, la CPU puede ser menos potente ya que hay menos carga al sistema.

WAR es un paquete cuya especificación fue desarrollada por Sun que permite agrupar un conjunto de clases y documentos que conforman una aplicación Web. Todos los ficheros debajo de la raíz /Frumansa pueden servirse al cliente excepto aquellos ficheros que están bajo los directorios especiales **META-INF** y **WEB-INF** ya que son directorios privados.
El paquete frumansa.WAR tiene la siguiente estructura:

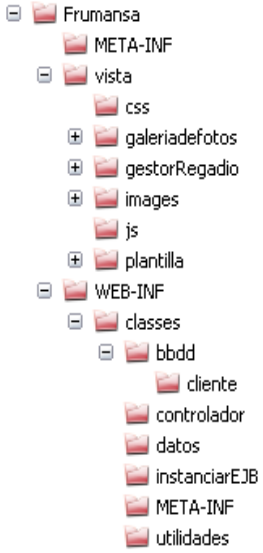

Figura 48 Estructura Frumansa.WAR

Dentro de frumansa.WAR se encuentra:

- **WEB-INF**: están los ficheros privados, contiene los paquetes con sus clases pertenecientes y el web.xml, es el descriptor de despliegue, se indican los Servlets contenidos en la aplicación.
- **Vista**: contiene los directorios públicos que el usuario tiene acceso.
- **META-INF**: contiene información sobre la aplicación.

Un **JAR** permite empaquetar clases de los EJB y los archivos relacionados para su despliegue. Contiene las clases de los EJB, interfaz remota, interfaz home y las clases dependientes. La estructura del JAR es la siguiente:

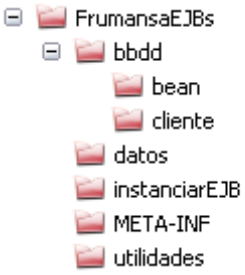

Figura 49 Estructura FrumansaEJBs.jar

FrumansaEJBs.jar contiene:

- **META-INF** se encuentra el descriptor de despliegue de los EJB, **ejb-jar.xml**, este es el estandar de SUN pero el servidor de aplicaciones Bea WebLogic necesita su propio descriptor de despliegue (**weblogic-ejb-jar.xml**) para su propia configuración.

El empaquetado de los archivos se ha realizado mediante la tecnología **ANT**, es una herramienta Open-Source utilizada para la realización de tareas mecánicas, compilando las clases Java y las coloca en un WAR o JAR.

CAPÍTULO 5. MANUAL DE INSTALACIÓN

En este capítulo se explica el procedimiento a seguir para instalar la aplicación en un ordenador[4]. La importancia de este capítulo proviene de lo complicada y extensa que es la instalación de una aplicación J2EE en un servidor de aplicaciones BeaWeblogic e importación de la base de datos al MySQL. Este manual se divide en ocho pasos que se explican detalladamente en cada apartado.

Los ficheros necesarios para instalar la aplicación son los siguientes.

- **Frumansa.WAR:** paquete con el desplegable que contiene los Jsps, las clases del controlador y las interfaces EJBs.
- **FrumansaEJBs.jar**: paquetes con el desplegable con los EJBs.
- **configuracionFrumansa.rar:** contiene todos los archivos properties de configuración.
- **Frumansa.sql**: script sql que crea la base de datos.
- Librería **Connector MySQL**: directorio con librería mysql-connector-java-necesario para conectar Bea Weblogic con MySQL
- Log4j.jar

5.1 Configuración de un dominio Weblogic

Para la configuración de un dominio WebLogic se supone que la máquina tiene instalado el servidor de aplicaciones Bea WebLogic 8.1. A continuación se procederá a la creación de un dominio del servidor Bea Weblogic.
En el menú de inicio → todos los programas → BEA weblogic platform 8.1→Configuration Wizard y aparecerá una ventana. Hay que seguir los siguientes pasos:

5.1.1. Selecciona "Create a new WebLogic configuration" y pulsa siguiente. En la figura 50 se muestra el aspecto que tiene la ventana.

Figura 50 Configuration Wizard

[4] El sistema operativo que se hace referencia es Windows XP

5.1.2. En la siguiente pantalla se selecciona la plantilla de configuración que se desea utilizar de BEA, selecciona “Basic Weblogic Server Domain” y pulsa siguiente.

5.1.3. Elija configuración Express para continuar la configuración pulsa siguiente.

5.1.4. En la siguiente ventana se debe de introducir el usuario: “**system**” y contraseña: “**weblogic**” para poder acceder al dominio y pulsa siguiente.

5.1.5. Seleccione “Development Mode” y el SDK de Sun y pulsa siguiente.

5.1.6. En la siguiente ventana donde pone “configuration Name” escribe el nombre del dominio “**frumansa**”. En él es donde se guardaran los archivos WAR y JAR. Pulsa siguiente.

5.1.7. La última ventana informa que el dominio ha sido creado. Seleccione la casilla “Start admin Server” y pulse sobre done. Con esta opción se inicia el servidor.

5.1.8. Al iniciar el servidor aparece la siguiente ventana:

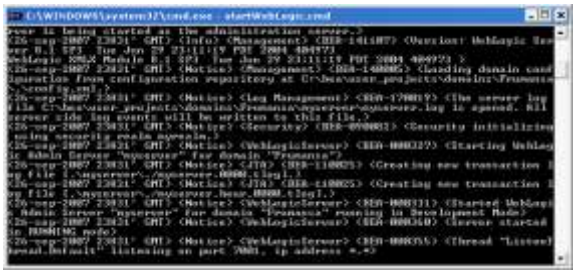

Figura 51 Consola de BeaWeblogic**.**

Nota: Si no ha seleccionado la casilla Start admin Server puedes encender el servidor desde: menú de inicio→todos los programas→BEA weblogic platform 8.1→user Project→Frumansa→Start Server

Una vez que la instalación del dominio es correcta, apague el servidor para poder realizar la configuración del dominio creado anteriormente.

5.2 Configurar los valores de ruta de clase

Tenga en cuenta que, en una configuración predeterminada de un dominio de servidor BEA WebLogic en un sistema Microsoft Windows XP, el archivo de comandos para iniciar un dominio se encuentra en el:

***directorio_inicial_BEA\weblogic81*`\user_projects\domains\`*`nombre_dominio`*`\startWebLogic.cmd.`**

directorio_ inicial_BEA es la ubicación de instalación del servidor WebLogic y nombre_dominio es el nombre del dominio WebLogic que quiere utilizar.
En plataformas UNIX y Linux, el archivo de inicio se encuentra en:
***directorio_inicial_BEA*`/weblogic81/user_projects/domains/`*nombre_dominio*`/startWebLogic.sh`**

5.2.1. En el cd hay una carpeta con el nombre de JAR, debe de localizar los siguientes archivos*:*

mysql-connector-java-5.0.3-bin.jar
log4j-1.2.15.jar
Cópialos y pégalos en el **directorio_inicial_BEA\weblogic81\server\lib**

5.2.2. Edite el archivo startWeblogic.cmd (startWeblogic.sh en plataformas unix) de su dominio WebLogic y agregue la línea siguiente justo antes de la línea en la que se define por primera vez la variable `CLASSPATH` (ruta de clase).

Set CLASSPATH_MYSQL*=C:\bea\weblogic81\server\lib\mysql-connector-java-5.0.3-bin.jar*
set CLASSPATH_LOG*=C:\bea\weblogic81\server\lib\log4j-1.2.15.jar*
set CLASSPATH_PROPERTIES*=C:\configuracionFrumansa*

5.2.3. En la línea **set CLASSPATH=%WEBLOGIC_CLASSPATH%; …** añada las dos variables anteriores*:*

Set CLASSPATH**=**
%CLASSPATH_PROPERTIES%;%CLASSPATH_LOG%;%CLASSPATH_MYSQL%;%WEBLOGIC_CLASSPATH%;…

5.2.4. En la última línea:

%JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS% -Dweblogic.Name=%SERVER_NAME% …
Se debe de añadir -classpath %CLASSPATH%; quedaría:

%JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS% -Dweblogic.Name=%SERVER_NAME% ***-classpath %CLASSPATH%...***

5.2.5. descomprima el ***configuracionFrumansa.rar*** en el disco local C.

5.2.6. Reinicie el servidor WebLogic.

5.3 Instalación MySQL

El programa de instalación del MySQL se puede bajar de la Web:
http://dev.mysql.com/downloads/mysql/5.0.HTML#win32

Para la administración de la base de datos debe de bajarse de la página Web :
http://mysql-gui-tools.malavida.com/d1083-descargar-windows
Las siguientes herramientas:

- MySQL Administrator 1.2 Generally Available (GA)
- MySQL Query Browser 1.2 Generally Available (GA)
- MySQL Migration Toolkit 1.1 Generally Available (GA)

Instalando el **MySQL:CONFIGURACION WIZARD**:
En las siguientes capturas se muestra la instalación de MySQL, debe los siguientes pasos:

5.3.1. Selecciona “Standard configuration” y pulsa siguiente. El aspecto del programa de instalación es el siguiente:

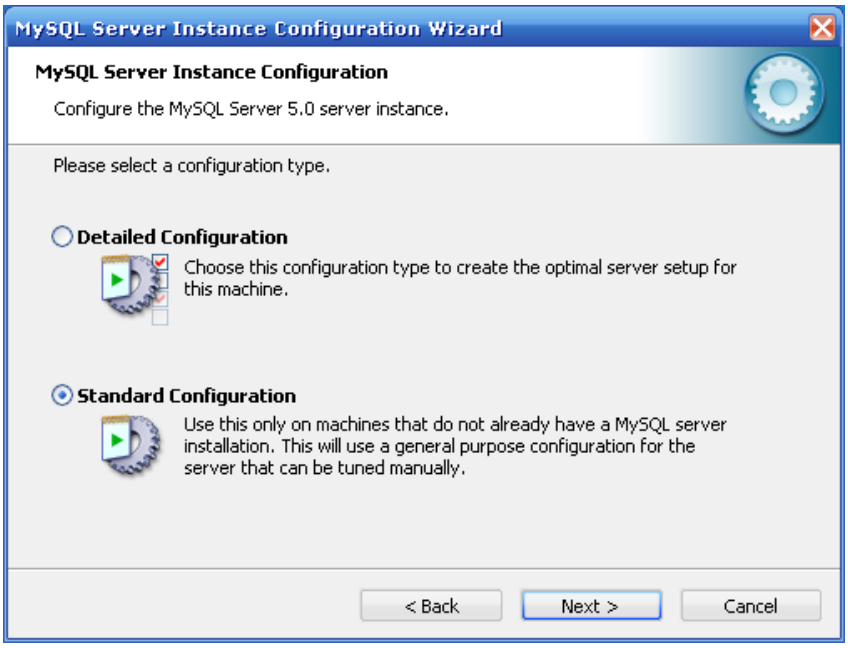

Figura 52 Configuration Wizard MySQL

5.3.2. Seleccione todas las casillas y el “service name” es MySQL, pulsa siguiente.

5.3.3. Selecciona “Modif. Security settings” es el usuario de ROOT una contraseña: **1234**. Debe de recordar esta contraseña para iniciar la sesión a la base de datos y para crear el conector JDBC con el Bea Weblogic. Pulsa siguiente.

5.3.4. En la ultima ventana pulsa en “execute” para llevar a cabo la ejecución de la configuración del MySQL.

5.3.5. Una vez instalado MySQL se debe de instalar las herramientas mysql-gui- tools-5.0.

5.4 Importación de la Base de Datos

5.4.1. Para importar la base de datos se debe ejecutar la herramienta Administrador MySQL instalada anteriormente.

5.4.2. Una vez ejecutado, introduzca los siguientes datos: Server host:localhost, port:3306, username:root y password:1234. Pulsa ok para entrar en la administración de la base de datos.

5.4.3. En el cd adjunto hay una carpeta BBDD que dentro contiene el script **frumansa.sql.** En la ventana de mysql, a la izquierda hay un botón que **Restore,** pulsa sobre él. En la pestaña general pulsa sobre el botón “Open Backup File” busca el strip frumansa.sql y ábrelo. Esto permite importar el script.

5.4.4. La base de datos ya ha sido importada. Si pulsas en el botón “catalogs” en la parte inferior izquierda debe de aparecer el nombre de la base de datos importada. La figura 53 muestra la base de datos importada en el MySQL.

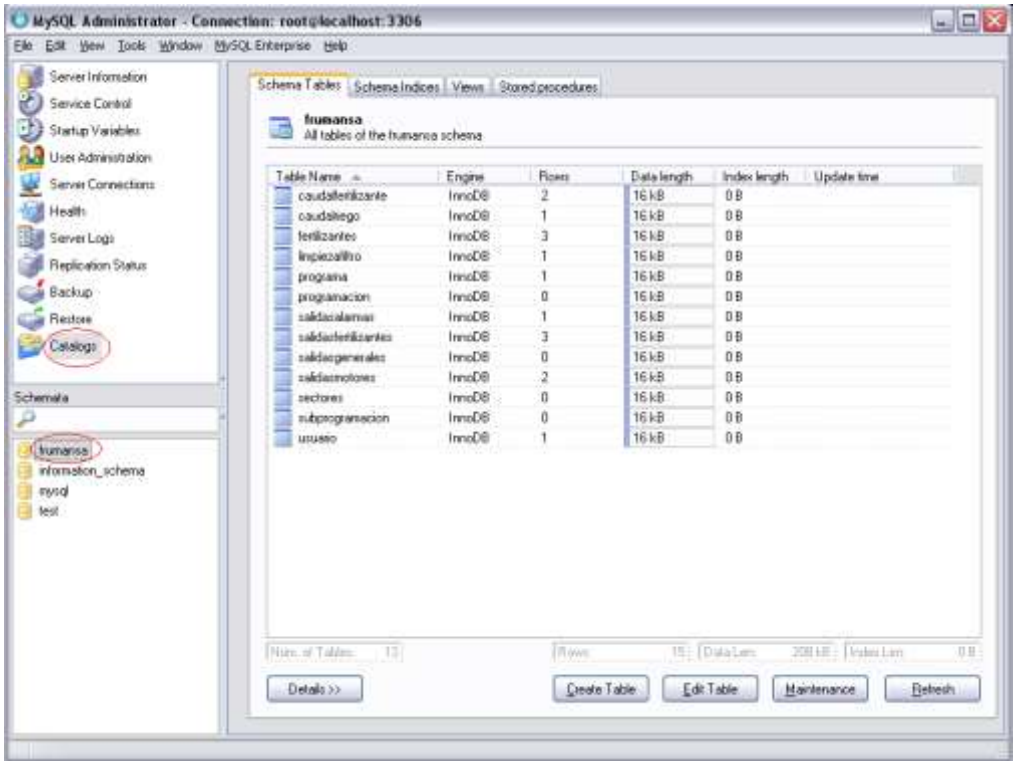

Figura 53 Script frumansa importado**.**

5.5 Configurar de conexiones JDBC

La aplicación utiliza un origen de datos JDBC, deberá configurar en primer lugar una agrupación de conexiones JDBC en el servidor WebLogic antes de poder crear un origen de datos JDBC. Encienda el servidor Bea Weblogic.

5.5.1. En el navegador Web, inicie la WebLogic Administration Console usando el URL `http://`*nombre_host*`:`*número_puerto*`/console/`; *nombre_host* es el nombre del sistema en el que está instalado el servidor WebLogic y *número_puerto* es el puerto en el que se está ejecutando el servidor. El puerto predeterminado es 7001.

5.5.2. Escriba el nombre de usuario: **system** y la contraseña: **weblogic**. Pulsa en sign in.

5.5.3. Haga clic para expandir el nodo “Services” después en “JDBC” en el panel izquierdo de la Administration Console.

5.5.4. Haga clic con el botón derecho en el nodo “Connection Pools” y seleccione “Configure a New JDBC Connection Pool” (Configurar una nueva agrupación de conexiones JDBC).

Se abre el asistente JDBC Connection Pool (Agrupación de conexiones JDBC) en el panel derecho.

5.5.5. En la lista “Database Type” (Tipo de base de datos) selecciona **MySQL** y en “Database Driver” (Controlador de base de datos) selecciona **com.mysql.jdbc.Driver** Haga clic en Continue.

5.5.6. Complete los datos en el cuadro de diálogo Data Connection Properties (Propiedades de la conexión de datos) con los siguientes datos:

En las propiedades de configuración han de coincidir los datos con los introducidos en el MySQL y la base de datos importada, En el campo Database Name (nombre base datos) ha de ser **frumansa**, ya que es importada anteriormente en el MySQL. En el campo Host Name ha de ser **localhost**, el puerto **3306**, que ya viene por defecto. El nombre del usuario es **root** y la contraseña es la introducida en el MySQL es **1234**.

5.5.7. En el cuadro de diálogo Test Database Connection (Probar conexión de base de datos), compruebe las propiedades de la conexión y haga clic en Test Driver Configuration (Probar configuración de controlador).

Si la prueba es satisfactoria se abrirá la página Create and Deploy (Crear y ejecutar). Si no se ha superado la prueba se mostrará un mensaje de error en la parte superior de la página. Compruebe los valores de la página, corrija los posibles errores y vuelva a probar la conexión.

5.5.8. Haga clic en el botón Create and Deploy.

Su nueva agrupación de conexiones aparecerá en una tabla en el panel derecho y en el nodo Connection Pools en el panel izquierdo.

5.6 Configurar un origen de datos JDBC

Si la aplicación que quiere ejecutar en el servidor WebLogic utiliza un origen de datos JDBC, deberá configurar dicho origen de datos antes de poder ejecutar la aplicación en el servidor. Una vez configurada una agrupación de conexiones de base de datos deberá configurar el origen de datos para esa conexión.

5.6.1. Expanda el nodo Services > JDBC en el panel izquierdo de la WebLogic Administration Console.

5.6.2. Haga clic en el nodo Data Sources (Orígenes de datos) y haga clic en Configure a new JDBC Data Source en el panel derecho. En el panel derecho se abre la página Configure a JDBC Data Source.

5.6.3. Escriba un nombre= **frumansa** y el nombre JNDI=**TFC** para el origen de datos JDBC.

5.6.4. Haga clic en Continue para asociar el origen de datos JDBC con una agrupación de conexiones.

5.6.5. En la lista desplegable “pool name”, selecciona la agrupación de conexiones **TFC**. Haga clic en Continue.

5.6.6. Seleccione el servidor de destino del nuevo origen de datos JDBC. Y haga clic en Create (Crear).

5.6.7. El origen de datos recién creado aparece en una tabla en el panel derecho.

En la siguiente figura se muestra la configuración realizada en este apartador:

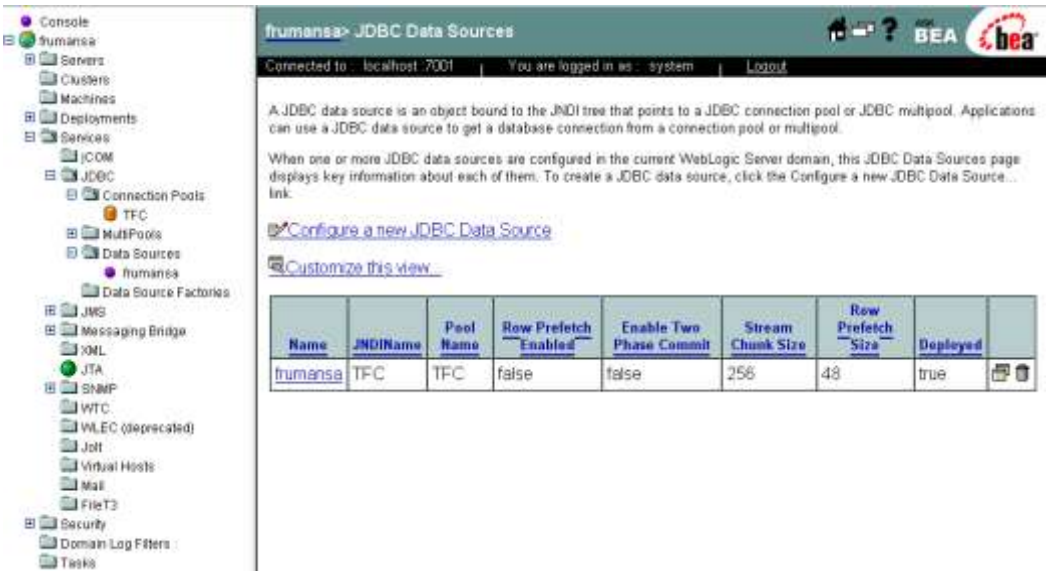

Figura 54 Configuración JDBC.

5.7 Configuración para los EJBs

La aplicación requiere la configuración de los EJB en el servidor.

5.7.1. Expanda el nodo deployments > startup & shutdwon

5.7.2. En el panel derecho, haga clic en configure a “new Startup Class” y añada la siguiente línea en ClassName:

bea.jolt.pool.servlet.weblogic.PoolManagerStartUp

Y haga clic en create.
Expanda de nuevo el nodo deployments > startup & shutdwon. Haz clic en el panel derecho en “shutdwon Class” y añada la línea en ClassName:

bea.jolt.pool.servlet.weblogic.PoolManagerShutDown

Y haga clic en create. La configuración ha sido creada.

5.7.3. En el nodo EJB Module, una vez se que coloquemos el JAR en el dominio (se realiza más a delante), el mismo servidor coge los EjBs. En el panel derecho se muestra el JAR. Clica sobre él.

5.7.4. En la pestaña Monitoring del panel derecho, se muestran todos los EJBs de la aplicación:

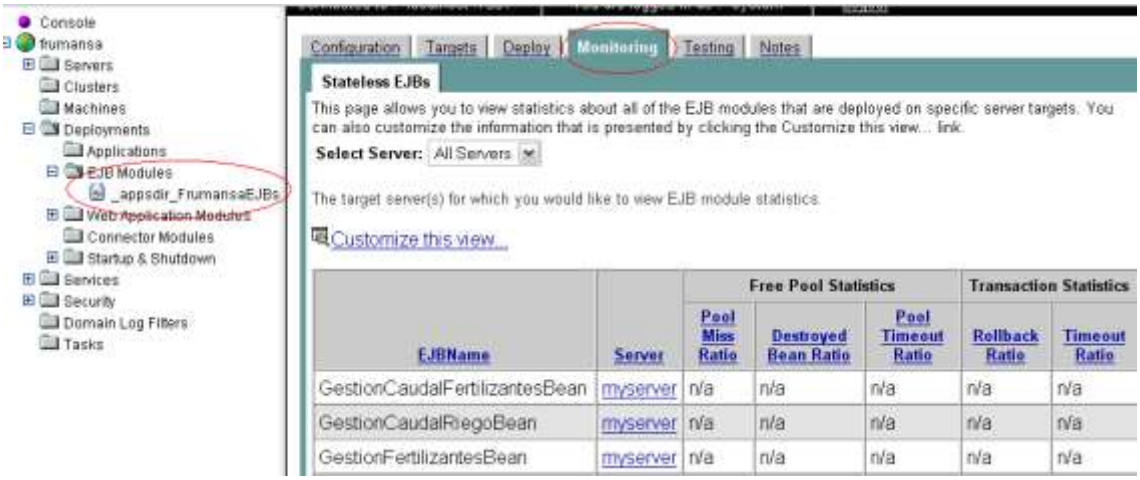

Figura 55 Configuración EJBs.

5.8 Añadir WAR y JAR en el dominio J2EE

5.8.1. Apaga el servidor de aplicaciones.

5.8.2. En el cd hay una carpeta llamada Aplicación, en ella se encuentran dos ficheros: frumansa.WAR y frumansaEJBs.jar, copialos y pegalos en la carpeta applications que está dentro del dominio “frumansa”

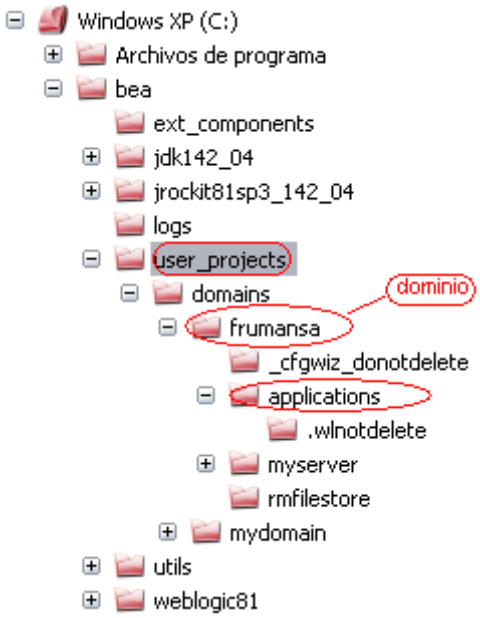

Figura 56 añadir JAR y WAR en el dominio.

5.8.3. Reinicia el servidor

5.8.4. En el navegador Web, inicie la WebLogic Administration Console usando el URL `http://`*`nombre_host`*`:`*`número_puerto`*`/console/`; *`nombre_host`* es el nombre del sistema en el que está instalado el servidor WebLogic y *número_puerto* es el puerto en el que se está ejecutando el servidor, 7001.

5.8.5. Escriba el usuario y contraseña.

5.8.6. En los nodo EJB module debe aparecer el JAR y en Web aplication Module debe de aparecer el WAR. Dentro del nodo Web aplication Module clica en appsdir_Frumansa_WAR, en la pestaña Testing del panel derecho puedes ver que la aplicación ya está corriendo dentro del servidor.

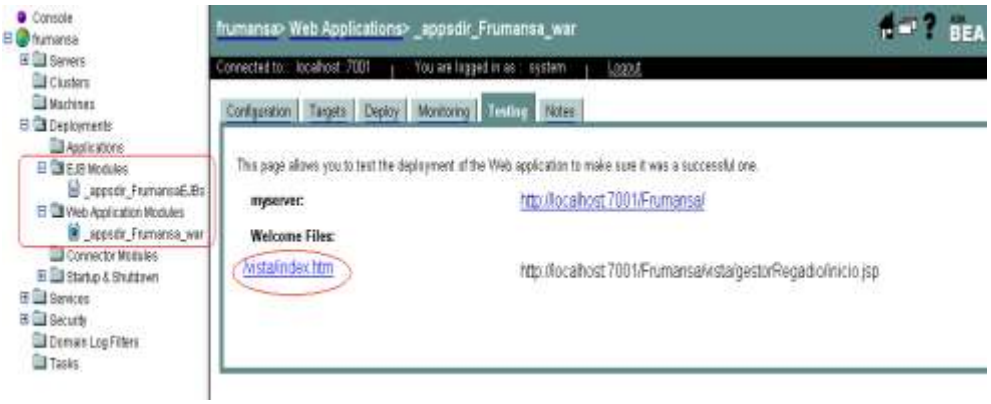

Figura 57 Aplicación corriendo dentro del servidor.

5.8.7. Y para finalizar desde el navegador Web escribe la siguiente dirección:

http://localhost:7001/Frumansa/vista/index.htm

Si aparece la siguiente figura, la instalación ha sido satisfactoria:

Figura 58 index.htm

Los datos guardados en la base de datos del usuario, para entrar a la zona privada son: login: **mansaneta**
Contraseña: **123456**
Una vez el administrador halla iniciado sesión tiene la opción de cambiar esta contraseña.

CAPÍTULO 6. CONCLUSIONES

6.1. Revisiones de objetivos y requisitos funcionales.

Prácticamente se ha conseguido todos los requisitos funcionales propuestos, ya que se tiene una gestión de regadío a través de una aplicación Web.
La aplicación consigue configurar todos los parámetros necesarios de la gestión de regadío. Antes de introducir los datos, siempre se le muestran al administrador los últimos datos almacenados en la base de datos y con la posibilidad de modificarlos y volverlos a guardar.

El único objetivo que no se ha podido llevar a cabo técnicamente, ha sido la detección de anomalías, no ha sido posible debido a que la empresa Progres no nos facilitó la tecnología para acceder al ordenador de riego Agronic, los encargados de esta detección de errores son unos sensores que están colocados en cada componente del cabezal de riego, cuando uno de estos falla, el sensor avisa al Agronic 4000 y este es el encargado de enviar estos errores.

Además el administrador puede iniciar y cerrar sesión cuando el lo desee y cambiar su propia contraseña, es más, si la contraseña se le olvida hay una funcionalidad que le envía un correo con su contraseña.

También se ha aprovechado esta aplicación para hacer una Web corporativa. Cualquier usuario de Internet puede tener información sobre la empresa, sus productos. Incluso puede solicitar información con un formulario de contacto.

6.2. Conclusiones

Al ser el primer proyecto que realizo, teniendo en cuenta su importancia y el tiempo que se le ha dado, se puede sacar muchas conclusiones al respecto: las conclusiones técnicas y las personales.

6.2.1. Conclusiones técnicas

El proyecto consta de varias partes. Entre ellas está la investigación, la documentación, el diseño, el desarrollo, la implementación y por último las pruebas realizadas.

Investigar y documentar supone mucho tiempo. Realizar bien esta tarea me ha facilitado mucho trabajo en las posteriores etapas del proyecto. Era muy necesario investigar sobre el Agronic 4000 y del cabezal de riego ya que debía conocer en detalle sus funcionalidades y componentes. Además, ha sido muy importante conocer bien las tecnologías utilizadas ya que se partía de cero en

la aplicación. Probablemente si pudiera haberle dedicado más tiempo, podría haber solucionado algunos problemas que han aparecido posteriormente con más facilidad.

Para comenzar a desarrollar debemos conocer bien los objetivos y requerimientos marcados por la necesidad del agricultor para la gestión de regadío.

Puedo concluir que ha sido complicado desarrollar al principio. Programar una aplicación Web J2EE conlleva mucho estudio ya que se ha empezado de cero.

Hay que tener en cuenta que la aplicación ha sido programada por capas y si en un futuro se deseara añadir alguna funcionalidad no llevaría mucho tiempo. Por otro lado los parámetros susceptibles de ser modificados para cada instalación han sido definidos en ficheros de configuración externos a la aplicación, minimizando así los cambios de código fuente para cada instalación. Por otro lado, durante la ejecución de la aplicación se utiliza un sistema de Log que traza en todo momento las acciones y errores realizados por la aplicación.

6.2.2. Conclusiones personales

El desarrollo de de este trabajo final de carrera me ha servido para formarme como ingeniera y aprender lecciones que no se dan en clase. El TFC es la conexión entre la carrera y el mundo laboral, se pueden aprender cosas con ello. Como planificar y estructurar un proyecto, plantear los objetivos e intentar conseguirlos, redactar una memoria… Son fundamentales estos conocimientos ya que pueden ser habituales en el mundo laboral.

Al ser un TFC de idea propia la motivación de la realización ha sido mayor que si hubiese sido un TFC propuesto por un departamento de la universidad o empresa.

Por último, he conseguido desarrollar una aplicación que cumple los objetivos propuesto inicialmente. Es una aplicación que posiblemente sirva de esqueleto para crear otras nuevas aplicaciones Web.

6.3. Líneas de trabajo futuro

El proyecto se ha marcado objetivos para diseñar un prototipo de aplicación. Para conseguir un buen funcionamiento del sistema de regadío mediante una arquitectura distribuida J2EE es necesario conectar con el ordenador de riego Agronic 4000, para ello existen varias líneas de trabajo que hay que seguir:

- Filtrado de los paquetes entre el Agronic PC y Agronic 4000 para averiguar que tipo de conexión y la información exacta que se envían.
- Seguidamente, investigar de cómo adaptar la aplicación para acceder al agronic 4000. (ingeniería inversa)

- Y el manejo de varios equipos de riego desde una misma aplicación.

Es de mi interés seguir involucrada en el desarrollo de esta aplicación. No descarto esta opción ya que mi línea de vida siempre ha estado en vuelta en temas de agricultura y mi línea de trabajo futuro será continuar con un postgrado de Comunicación, redes y gestión de contenidos.

BIBLIOGRAFÍA

[1] Agronic 4000
<http://www.progres.es/espanyol/agronic4000.html>

[2] J2EE v1.4 Documentation
<https://java.sun.com/j2ee/1.4/docs/>

[3] Javamail
<http://java.sun.com/products/javamail/>

[4] Enterprise JavaBeans Technology
<http://java.sun.com/products/ejb/>

[5] Log4J
<http://logging.apache.org/log4j/1.2/index.html >

[6] Gotapi
<http://www.gotapi.com>

[7] Javahispano documentation
<http://javahispano.org/documentacion/ >

[8] Tutoriales sobre RMI y EJB
<http://www.adictosaltrabajo.com/indexg.php?noimages=SI&pagina=listacat&id=16>

[9] Manuales
<http://www.programacion.com/>

[10] Manual de Bea Weblogic
<http://www.programacion.com/java/tutorial/beaintro/>

[11] Bea WebLogic
<http://www.bea.com/>

[12] MYSQL
<http://www.mysql.com/>

[13] Eclipse
<http://www.eclipse.org/>

[14] Manual J2EE
<http://xxito.files.wordpress.com/2008/05/manualj2ee.pdf>

[15] Eguíluz Pérez J., Introducción a JavaScript: Libro digital.
<http://www.librosweb.es/javascript/index.html>

[16] Eguíluz Pérez J., Introducción a CSS: Libro digital.
<http://www.librosweb.es/css/index.html>

[17] Eckel, Bruce, Piensa en Java. Prentice, 2007.

[18] Martin Sierra, A.J, Programador certificado java 2. Ra – MA, 2006.

[19] Ramos A., Ramos M., Montero F., Desarrollo de aplicaciones en entornos de 4ª generación y con herramientas case, Mc Graw Hill 2000.

[20] Programa de instalación del MySQL.
http://dev.mysql.com/downloads/mysql/5.0.HTML#win32

[21] Para la administración de la base de datos.
http://mysql-gui-tools.malavida.com/d1083-descargar-windows

ANEXOS

1. Escenario de la finca

Se muestran unas imágenes del cabezal de riego instalado en las fincas de Frumansa.

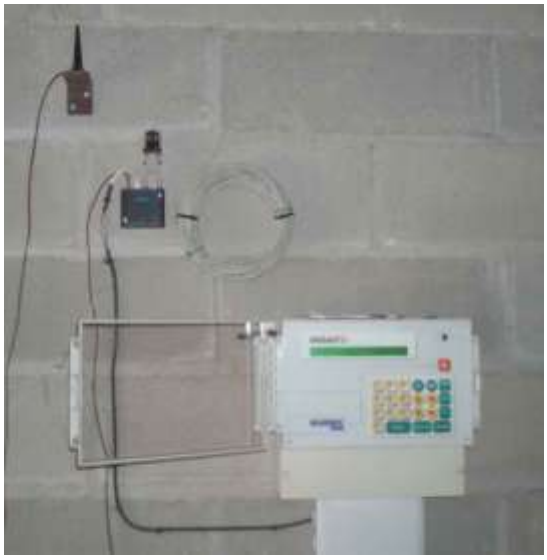

Figura 59 Agronic 4000

Figura 60 Filtro

Figura 61 Fertilizante y agitador

Figura 62 sectores

Figura 63 salida sectores

Figura 64 Relé

Figura 65 Gestión de riego

2. Tecnología J2EE

Java 2 Enterprise Edition se trata de una plataforma completa para construir aplicaciones Web basadas en el lenguaje Java. Se trata de una serie de tecnologías que permiten escribir aplicaciones en el lado del servidor para proporcionar servicios desde redes TCP/IP. Todas las tecnologías que componente J2EE fueron desarrolladas por Sun Microsystems y otros vendedores de software entre los que se incluye BEA Systems.
Un concepto muy importante que se ha de tener en cuenta, es que J2EE resuelve el problema del coste y la complejidad del desarrollo de servicios multi-capa, **proporcionando arquitecturas escalables, de alta disponibilidad, seguras y eficientes.**

Una arquitectura multi-capa consta de dos tipos de capas:

1) Las capas de **Componentes Software**, están compuestos por 3 capas internas:

   - La capa del **cliente** contiene los programas ejecutados por los usuarios, incluyendo navegadores Web y programas de aplicaciones de red. Estos programas se pueden escribir virtualmente en cualquier lenguaje de programación.
   - La capa **media** contiene el servidor que son direccionados directamente por los clientes, como servidores Web existentes o servidores Proxy.
   - La capa **backend** contiene los servicios que son accesibles a los clientes sólo a través del servidor. También incluyen bases de datos, sistemas de hojas de operación (planning) de recursos de la

empresa (ERP), aplicaciones mainframe, aplicaciones legales de la empresa, y monitores de transacciones

2) El segundo tipo, son las capas **Lógicas de Aplicación**:

- La capa de **presentación** incluye una lógica de interfaz de usuario y de visualización de aplicaciones. La mayoría de las aplicaciones J2EE utilizan un navegador Web en la máquina del cliente porque es mucho más fácil que programas de cliente que se despliegan en cada ordenador de usuario. En este caso, la lógica de la presentación es el contenedor Web del servidor WebLogic. Sin embargo, los programas del cliente escritos en cualquier lenguaje de programación, sin embargo, deben contener la lógica para representar el HTML o su propia lógica de la presentación.
- La capa de **control**: esta capa se encarga de gestionar las peticiones y repuestas entre la capa de presentación con la de lógica de negocio. La mayoría de las aplicaciones J2EE. Los servlets son los que proporcionan el control de la aplicación.
- Capa de **Lógica de Negocio.** Los JavaBeans Enterprise son componentes de la lógica de negocio para aplicaciones J2EE.

Estos dos tipos de multi-capas se basan en el patrón **modelo-vista-controlador (MVC):**

- **El Modelo: Encapsula** los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.
- **La Vista:** Muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.
- **El controlador (Controller)**: Reciben las entradas, usualmente como eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio ("**service requests**") para el modelo o la vista.

.

Figura 66 Patron MVC (modelo vista y controlador)

Como se ha dicho anteriormente, el lenguaje para construir aplicaciones Web es Java. A continuación, se muestra una descripción de las características más importantes del lenguaje:

- Es **simple:** Es un lenguaje sencillo de aprender. Su sintaxis es la de C++ “simplificada”. Los creadores de Java partieron de la sintaxis de C++ y trataron de eliminar de este todo lo que resultase complicado o fuente de errores en este lenguaje.

- Está **orientado a objetos:** Posiblemente sea el lenguaje más orientado a objetos de todos los existentes; en Java todo, a excepción de los tipos fundamentales de variables (int, char, long...) es un objeto.
- Es **distribuido:** Java está muy orientado al trabajo en red, soportando protocolos como TCP/IP, UDP, HTTP y FTP. Por otro lado el uso de estos protocolos es bastante sencillo comparando lo con otros lenguajes que los soportan.
- Es **robusto:** El compilador Java detecta muchos errores que otros compiladores solo detectarían en tiempo de ejecución o incluso nunca. (ej: if(a=b) then... el compilador Java no nos dejaría compilar este código.
- Es **seguro:** Sobre todo un tipo de desarrollo: los applets. Estos son programas diseñados para ser ejecutados en una página Web. Java garantiza que ningún applet puede escribir o leer de nuestro disco o mandar información del usuario que accede a la página a través de la red (como, por ejemplo, la dirección de correo electrónico). En general no permite realizar cualquier acción que pudiera dañar la máquina o violar la intimidad del que visita la página Web.
- Es **portable:** En Java no hay aspectos dependientes de la implementación, todas las implementaciones de Java siguen los mismos estándares en cuanto a tamaño y almacenamiento de los datos.
- Es una **arquitectura neutral:** El código generado por el compilador Java es independiente de la arquitectura: podría ejecutarse en un entorno UNIX, Mac o Windows. El motivo de esto es que el que realmente ejecuta el código generado por el compilador, no es el procesador del ordenador directamente, sino que este se ejecuta mediante una máquina virtual. Esto permite que los applets de una Web pueda ejecutarlos cualquier máquina que se conecte a ella independientemente de que sistema operativo emplee (siempre y cuando el ordenador en cuestión tenga instalada una máquina virtual de Java).
- Tiene un **rendimiento medio:** Actualmente la velocidad de procesado del código Java es semejante a la de C++, hay ciertas pruebas estándares de comparación (benchmarks) en las que Java gana a C++ y viceversa. Esto es así gracias al uso de compiladores *just in time*, compiladores que traduce los bytecodes de Java en código para una determinada CPU, que no precisa de la máquina virtual para ser ejecutado, y guardan el resultado de dicha conversión, volviéndolo a llamar en caso de volverlo a necesitar, con lo que se evita la sobrecarga de trabajo asociada a la interpretación del bytecode. No obstante por norma general el programa Java consume bastante más memoria que el programa C++, ya que no sólo ha de cargar en memoria los recursos necesario para la ejecución del programa, sino que además debe simular un sistema operativo y hardware virtuales (la máquina virtual). Por otro lado la programación gráfica empleando las librerías Swing es más lenta que el uso de componentes nativos en las interfaces de usuario. En general en Java se ha sacrificado el rendimiento para facilitar la programación y sobre todo para conseguir la característica de neutralidad arquitectural, si bien es cierto que los avances en las máquinas virtuales remedian cada vez más estas decisiones de diseño.
- **Multithread:** Soporta de modo nativo los threads, sin necesidad del uso de de librerías específicas (como es el caso de C++). Esto le permite

además que cada Thread de una aplicación java pueda correr en una CPU distinta, si la aplicación se ejecuta en una máquina que posee varias CPU. Las aplicaciones de C++ no son capaces de distribuir, de modo transparente para el programador, la carga entre varias CPU.

Una vez vistas las características de java, hay que saber que las APIs están en el paquete **javax**. Las fundamentales APIs se encuentran en el son las siguientes:

- **Servlets**.
- **JSP**.
- **EJB** (*Enterprise Java Beans*).
- **JDBC.**

Para ejecutar aplicaciones J2EE se necesitan servidores de aplicaciones compatibles, por ejemplo:

- Bea WebLogic.
- IBM WebSphere ApplicationServer.
- Oracle Application Server

De forma gratuita y de código abierto, *open source* (implementaciones parciales):

- Tomcat.
- JBoss.
- Evidan JOnAS.

A continuación se va entrar en detalle sobre las APIs mencionadas anteriormente.
La primera API son los **SERVLETS,** La tecnología **Servlet** proporciona las mismas ventajas del lenguaje Java en cuanto a **portabilidad** (“write once, run anywhere”) y **seguridad**, ya que un servlet las características del lenguaje. Esto es algo de lo que carecen los ***programas CGI***, ya que hay que compilarlos para el sistema operativo del servidor.

Se pueden destacar las siguientes **características** sobre los **servlets:**

- Son independientes del servidor utilizado y de su sistema operativo, lo que quiere decir que a pesar de estar escritos en ***Java***, el servidor puede estar escrito en cualquier lenguaje de programación, obteniéndose exactamente el mismo resultado que si lo estuviera en ***Java***.
- Los ***servlets*** pueden llamar a otros ***servlets***, e incluso a métodos concretos de otros ***servlets***. De esta forma se puede distribuir de forma más eficiente el trabajo a realizar. Por ejemplo, se podría tener un ***servlet*** encargado de la interacción con los clientes y que llamara a otro ***servlet*** para que a su vez se encargara de la comunicación con una base de datos. De igual forma, los ***servlets*** permiten *redireccionar* peticiones de servicios a otros ***servlets*** (en la misma máquina o en una máquina remota).
- Los ***servlets*** pueden obtener fácilmente información acerca del ***cliente*** (la permitida por el protocolo ***HTTP***), tal como su dirección ***IP***, el ***puerto*** que se utiliza en la llamada, el método utilizado (***GET***, ***POST*****,** ...), etc.

- Permiten además la utilización de ***cookies*** y ***sesiones***, de forma que se puede guardar información específica acerca de un usuario determinado, personalizando de esta forma la interacción cliente-servidor. Una clara aplicación es ***mantener la sesión*** con un cliente.
- Los ***servlets*** pueden actuar como enlace entre el cliente y una o varias ***bases de datos*** en arquitecturas *cliente-servidor de 3 capas* (si la base de datos está en un servidor distinto).
- Asimismo, pueden realizar tareas de ***proxy*** para un ***applet***. Debido a las restricciones de seguridad, un ***applet*** no puede acceder directamente por ejemplo a un servidor de datos localizado en cualquier máquina remota, pero el ***servlet*** sí puede hacerlo de su parte.
- Los ***servlets*** permiten la generación dinámica de código ***HTML*** dentro de una propia página ***HTML***. Así, pueden emplearse ***servlets*** para la creación de páginas dinámicas.

En definitiva, se pueden resaltar las siguientes **ventajas** de los servlets ante los CGI:

- **Mejora del rendimiento**. Los Servlets usan la misma aplicación y para cada petición lanzan un nuevo hilo (al estilo de los Sockets).
- **Simplicidad**. Quizá la clave de su éxito. El cliente sólo necesita un navegador HTTP. El resto lo hace el servidor.
- **Control de sesiones**. Se pueden almacenar datos sobre las sesiones del usuario (una de las taras más importantes de http).
- **Acceso a la tecnología Java**. Lógicamente esta tecnología abre sus puertas a todas las posibilidades de Java (JDBC, Threads, etc.).

A lo que respecta al patrón MVC, un servlet se implementa dentro de la capa controlador, interactuando de intermediario entre la vista y el modelo.

La segunda API a comentar son las JSPs. Las **JavaServer Pages** (JSP) permiten separar la parte dinámica de nuestras páginas Web del HTML estático. El código de programación es el mismo del HTML regular de la forma normal, usando cualquier herramienta de construcción de paginas Web que usemos normal. Con la diferencia de que se encierra el código de las partes dinámicas en unas etiquetas especiales, la mayoría de las cuales empiezan con "<%" y terminan con "%>"
Hay tres tipos de construcciones JSP que embeberemos en una página: elementos de script, directivas y acciones. Los elementos de script permiten especificar código Java que se convertirá en parte del servlet resultante, las directivas permiten controlar la estructura general del servlet, y las acciones permiten especificar componentes que deberían ser usados, y de otro modo controlar el comportamiento del motor JSP.

Las **ventajas** de las JSPs que se pueden encontrar son:

- **Contra Active Server Pages** (ASP). ASP es una tecnología similar de Microsoft. Las ventajas de JSP están duplicadas. Primero, la parte dinámica está escrita en Java, no en Visual Basic. Segundo, es portable a otros sistemas operativos y servidores Web.

- **Contra los Servlets**. JSP no nos da nada que no pudiéramos en principio hacer con un servlet. Pero es mucho más conveniente escribir (y modificar!) HTML normal que tener que hacer un billón de sentencias println que generen HTML. Además, separando el formato del contenido podemos poner diferentes personas en diferentes tareas: los expertos en diseño de páginas Web pueden construir el HTML, dejando espacio para que los programadores de servlets inserten el contenido dinámico.
- **Contra Server-Side Includes (SSI)**. SSI es una tecnología ampliamente soportada que incluye piezas definidas externamente dentro de una página Web estática. JSP es mejor porque permite usar servlets en vez de un programa separado para generar las partes dinámicas. Además, SSI, realmente está diseñado para inclusiones sencillas, no para programas "reales" que usen formularios de datos, hagan conexiones a bases de datos, etc.
- **Contra JavaScript**. JavaScript puede general HTML dinámicamente en el cliente. Esto una capacidad útil, pero sólo maneja situaciones donde la información dinámica está basada en el entorno del cliente. Con la excepción de las cookies, el HTTP y el envió de formularios no están disponibles con JavaScript. Y, como se ejecuta en el cliente, JavaScript no puede acceder a los recursos en el lado del servidor, como bases de datos, catálogos, información de precios, etc.

En lo que respecta al patrón MVC, una jsp se implementa dentro de la capa vista, que interactúa con el usuario presentándole los datos y realizando todas las peticiones deseadas a la capa controlador.

El Elb API **EJB (Enterprise Java Beans)** es una arquitectura para desarrollar y desplegar aplicaciones distribuidas basadas en componentes.
Las aplicaciones basadas en EJB son escalables, transaccionales y con seguridad multiusuario.

Las características más relevantes de un EJB son las siguientes:

- Permiten el desarrollo de aplicaciones débilmente acopladas.
- Su comportamiento está especificado por **interfaces**.
- Su funcionamiento se puede configurar en tiempo de despliegue de forma declarativa (descriptor de despliegue).
- Los componentes EJB son **multiplataforma** (WODE: Write Once Deploy Everywhere).
- Como es un desarrollo basado en **componentes**, proporciona:
  - Reusabilidad
  - Modularidad
  - Interoperabilidad entre aplicaciones distribuidas

Hay tres tipos diferentes de EJBs, sus características son:

- **Beans de sesión** (*Session Beans):* Un bean de sesión reperesenta un único cliente y no se comparte entre clientes. Un cliente llama a los métodos del bean de sesión, que son dirigidos a través del contenedor EJB al bean enterprise.

Proporcionan servicios (capa de negocio). Hay dos tipos de Beans de sesión:

- **Beans de sesión sin estado**: Un bean de sesión sin estado no mantiene un estado conversacional para cada cliente individual. Cada llamada a un bean de sesión sin estado debería considerarse como uns petición a un ejemplar totalmente nuevo, ya que cualquier estado de las variables de ejemplar se perderá entre dos llamadas.
- **Beans de sesión con estado**: un bean de sesión con estado mantiene un estado conversacional con un cliente durante la duración de una sesión. Esto implica que el bean de sesión con estado puede mantener ejemplares de variables a través de varias llamadas desde un cliente durante una única sesión. Cuando el cliente termina de interactuar con el bean enterprise y el contenedor EJB lo elimina, termina la sesión del bean y se borran todos los datos de estado del bean.

- **Beans de entidad** (*Entity Beans):* están pensado para representar la lógica de negocio de una entidad existente en un almacenamiento persistente, proporcionan acceso a datos y relaciones entre ellos (capa de persistencia).
  - **Con persistencia gestionada por el bean (BMP**): el bean de entidad contiene código de acceso a la base de datos (normalmente JDBC) y es responsable de leer y escribir en la base de datos su propio estado. Las entidades BMP tienen mucha ayuda ya que el contenedor alertará al bean siempre que sea necesario hacer una actualización o leer su estado desde la base de datos. El contenedor también puede manejar cualquier bloqueo o transación, para que la base de datos se mantenga íntegra.
  - **Con persistencia gestionada por el contenedor** *(CMP):* el contenedor maneja la persistencia del bean de entidad. Las herramientas de los vendedores se usan para mapear los campos de entidad a la base de datos y no se escribe ningún código de acceso a las bases de datos en la clase del bean.
- Beans dirigidos por **mensajes** (Message Driven Beans): son los beans enterprise que manejan los mensajes asíncronos recibidos de colas de mensaje JMS. JMS encamina mensajes a un bean dirigido por mensaje, que selecciona un ejemplar de un almacén para procesar el mensaje.

La arquitectura de un EJB se muestra en la siguiente figura:

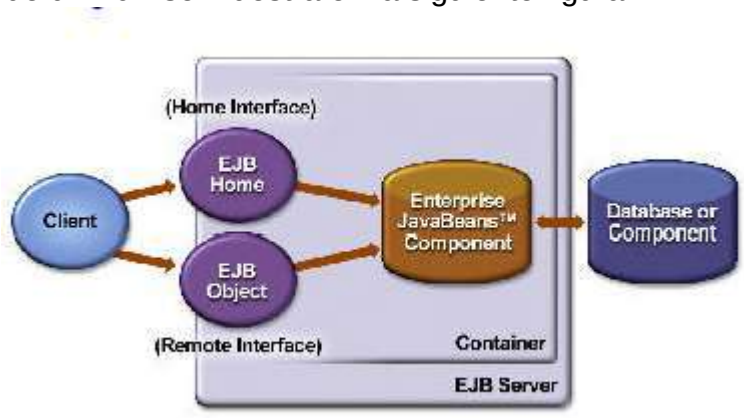

Figura 67 Arquitectura EJBs.

Se puede ver como un EJB necesita de la implementación de tres ficheros:

- Remote Interface
- Home Interface

- Enterprise Bean Class

A continuación, en la siguiente figura se muestran los EJBs en J2EE

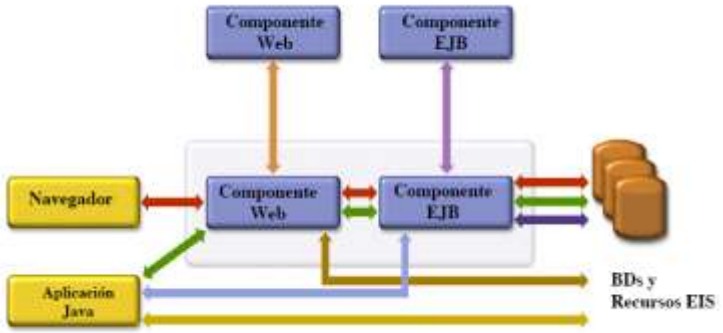

Figura 68 EJBs en J2EE.

Las **ventajas** que ofrece la arquitectura Enterprise JavaBeans a un desarrollador de aplicaciones se listan a continuación:

- **Simplicidad**. Debido a que el contenedor de aplicaciones libera al programador de realizar las tareas del nivel del sistema, la escritura de un enterprise bean es casi tan sencilla como la escritura de una clase Java. El desarrollador no tiene que preocuparse de temas de nivel de sistema como la seguridad, transacciones, multi-threading o la programación distribuida. Como resultado, el desarrollador de aplicaciones se concentra en la lógica de negocio y en el dominio específico de la aplicación.
- **Portabilidad de la aplicación**. Una aplicación EJB puede ser desplegada en cualquier servidor de aplicaciones que soporte J2EE.
- **Reusabilidad de componentes**. Una aplicación EJB está formada por componentes enterprise beans. Cada enterprise bean es un bloque de construcción reusable. Hay dos formas esenciales de reusar un enterprise bean a nivel de desarrollo y a nivel de aplicación cliente. Un bean desarrollado puede desplegarse en distintas aplicaciones, adaptando sus características a las necesidades de las mismas. También un bean desplegado puede ser usado por múltiples aplicaciones cliente.
- **Posibilidad de construcción de aplicaciones complejas**. La arquitectura EJB simplifica la construcción de aplicaciones complejas. Al estar basada en componentes y en un conjunto claro y bien establecido de interfaces, se facilita el desarrollo en equipo de la aplicación.
- **Separación de la lógica de presentación de la lógica de negocio**. Un enterprise bean encapsula típicamente un proceso o una entidad de negocio. (un objeto que representa datos del negocio), haciéndolo independiente de la lógica de presentación. El programador de gestión no necesita de preocuparse de cómo formatear la salida; será el programador que desarrolle la página Web el que se ocupe de ello usando los datos de salida que proporcionará el bean. Esta separación

hace posible desarrollar distintas lógicas de presentación para la misma lógica de negocio o cambiar la lógica de presentación sin modificar el código que implementa el proceso de negocio.

- **Despliegue en muchos entornos operativos**. Entendemos por entornos operativos el conjunto de aplicaciones y sistemas (bases de datos, sistemas operativos, aplicaciones ya en marcha, etc.) que están instaladas en una empresa. Al detallarse claramente todas las posibilidades de despliegue de las aplicaciones, se facilita el desarrollo de herramientas que asistan y automaticen este proceso. La arquitectura permite que los beans de entidad se conecten a distintos tipos de sistemas de bases de datos.
- **Despliegue distribuido**. La arquitectura EJB hace posible que las aplicaciones se desplieguen de forma distribuida entre distintos servidores de una red. El desarrollador de beans no necesita considerar la topología del despliegue. Escribe el mismo código independientemente de si el bean se va a desplegar en una máquina o en otra (cuidado: con la especificación 2.0 esto se modifica ligeramente, al introducirse la posibilidad de los interfaces locales).
- **Interoperabilidad entre aplicaciones**. La arquitectura EJB hace más fácil la integración de múltiples aplicaciones de diferentes vendedores. El interfaz del enterprise bean con el cliente sirve como un punto bien definido de integración entre aplicaciones.
- **Integración con sistemas no-Java**. Las APIs relacionadas, como las especificaciones Connector y Java Message Service (JMS), así como los beans manejados por mensajes, hacen posible la integración de los enterprise beans con sistemas no Java, como sistemas ERP o aplicaciones mainframes.
- **Recursos educativos y herramientas de desarrollo**. El hecho de que la especificación EJB sea un estándar hace que exista una creciente oferta de herramientas y formación que facilita el trabajo del desarrollador de aplicaciones EJB.

La base del funcionamiento de los componentes de los EJB es el protocolo **RMI**, haciendo posible el acceso a los objetos remotos. RMI (Remote Method Invocation) define la forma de comunicación remota entre objetos Java situados en máquinas virtuales distintas. Supongamos que un objeto cliente quiere hacer una petición a un objeto remoto situado en otra JVM (Máquina Virtual Java, Java Virtual Machine). RMI pretende hacer transparente la presencia de la red, de forma que cuando se escriba el código de los objetos clientes y remotos no tengas que tratar con la complicación de gestionar la comunicación física por la red.

Para ello, RMI proporciona al cliente un objeto proxy (llamado stub) que recibe la petición del objeto cliente y la transforma en algo que se puede enviar por la red hasta el objeto remoto. Este stub se hace cargo de todos los aspectos de bajo nivel (streams y sockets). En el lado del servidor, un objeto similar (llamado skeleton) recibe la comunicación, la desempaqueta y lanza el método correspondiente del objeto remoto al que sirve. Al igual que la petición, se deben empaquetar los argumentos de la llamada. El programador sólo debe

definir el código del método en el objeto remoto. Los objetos stub y skeleton los construye el compilador de RMI de forma automática.

Por último, **JDBC** es un API (incluida tanto en las diferentes versiones de J2SE y J2EE) que proporciona conectividad con gestores de bases de datos.
La idea de Sun era desarrollar una sola API (*application programming* interfaces, interfaz de programación de aplicaciones) para el acceso a bases de datos. Los requisitos deJDBC son:

- JDBC es una API a nivel SQL (el lenguaje SQL es el que realizaría la conexión con las bases de datos), independiente por tanto de la plataforma.
- JDBC debe ser similar al funcionamiento de las API para acceso a bases de datos existentes (en especial a la ya entonces famosa ODBC de Microsoft).
- JDBC es sencilla de manejar

En la figura siguiente se puede apreciar como la idea es que las aplicaciones sólo se tengan que comunicar con el interfaz JDBC. Éste es el encargada de comunicarse con los sistemas de base de datos.

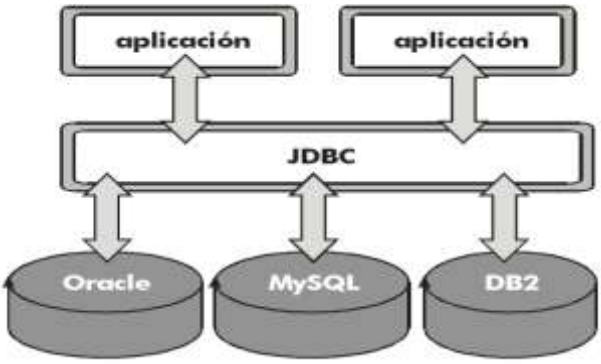

Figura 69 Comunicación JDBC.

Para finalizar, en la siguiente figura se muestra la evolución de J2EE que ha tenido a largo de los años hasta la actualidad:

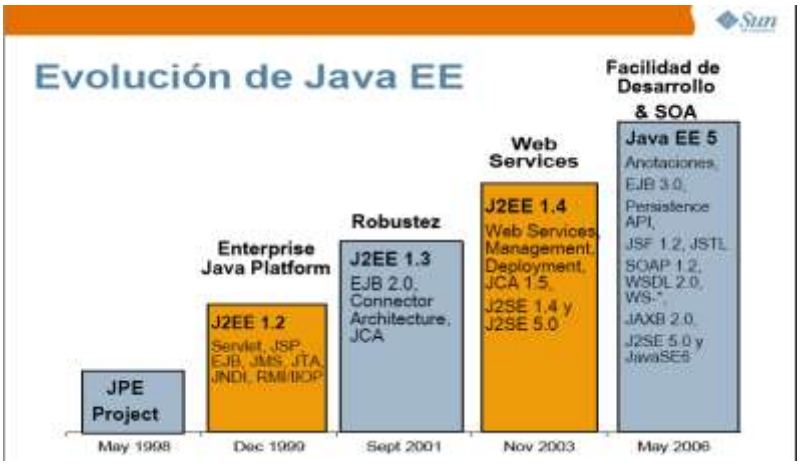

Figura 70 Evolución J2EE

3. Herramientas y software base.

3.1. BEA WebLogic 8.1

Ofrece diferentes servicios como:

- Lógica de presentación: está basada en la presentación de los datos al usuario utilizando un navegador Web. Representando por documentos html las jsp.
- Lógica de negocio: Los EJB son los principales componentes de este servicio.
- **Capa** de **aplicación**: Los clientes basados en Web se conecta con el servidor usando el protocolo http siendo protocolo petición-respuesta. El servidor BEA responde las peticiones http ejecutando un Servlet, que devuelve resultados al cliente. La jsp busca un objeto Servlet en un servidor BEA WebLogic mediante JNDI (Java Naming and Directory Interface), se explica más adelante. Los Servlets se conectan con el protocolo RMI a un EJB, es el recurso para aplicaciones distribuidas.
- **Datos y servicios de Acceso**: **JNDI** es un API estándar de java que permite a las aplicaciones buscar un objeto por su **nombre**, Servlets, EJB, colas JMS y JDBC DataSources. La conectividad de la base de datos JDBC java proporciona acceso a los recursos de la base de datos utilizando el driver JDBC, que es un interface especificada para cada proveedor de la base de datos. La ventaja de Bea WebLogic que ofrece almacenes de conexiones JDBC, cada conexión se guarda con un JNDI.
- **Tecnología de mensajería**: permite aplicaciones comunicarse con otras intercambiando mensajes. Incluyen APIs **Java Message Service** (JMS) y **JavaMail**.

A continuación se definen las ventajas del servidor de aplicaciones:

- BEA es la única compañía que provee de una plataforma unificada a aquellas organizaciones que quieran construir, extender, integrar, utilizar y gestionar aplicaciones y procesos empresariales.
- La Plataforma BEA WebLogic proporciona una sólida y expandible infraestructura.

La **desventaja** de utilizar BEA es:

- No es un software libre.

3.2. Eclipse 3.1.

El software utilizado para el desarrollo del código Java es Eclipse 3.1. Es una plataforma de desarrollos open source basada en Java. Es un desarrollo de IBM cuyo código fuente fue puesto a disposición de los usuarios. En sí mismo Eclipse es un marco y un conjunto de servicios para construir un entorno de desarrollo a partir de componentes conectados (plug-in). Hay plug-ins para el desarrollo de Java (JDT Java Development Tools) así como para el desarrollo en C/C++, COBOL, etc. los plug-ins están disponibles en la Web.Solo se

cargan los necesarios en el momento de ser utilizados con el objetivo de minimizar el tiempo de arranque del eclipse y de los recursos.
La base para el Eclipse es la plataforma de cliente enriquecido. Dispone de editor de texto, resaltado de sintaxis, compilación de tiempo real, pruebas unitarias con JUnit, controlador de versiones con CVS, integración de ANT, asistentes para creación de proyectos, clases, test, etc

3.3. Together 6.1

Para el desarrollo de los diagramas UML se ha utilizado en **Together 6.1**, Together es una herramienta utilizada para el análisis, diseño e implementación de aplicaciones orientadas a objetos. Para el modelado de la aplicación, utiliza el lenguaje estándar **UML** (Unified Modeling Language) el cual proporciona un vocabulario uniforme, una notación textual y gráfica, y un conjunto de reglas para conseguir un modelado correcto. Adicionalmente a los diagramas, también soporta los conceptos de estereotipos, restricciones, anotaciones y extensibilidad.

Algunas de las **ventajas** de utilizar Together para realizar el modelado del sistema son:

- Together permite tener en cualquier momento **sincronizado** todo el modelo UML con el código que implementa este modelo; es decir; cambios en el modelo se verán reflejados directamente en cambios en el código y viceversa.
- Existe la posibilidad de asignar distintos colores a distintos elementos según los roles de estos en los diagramas, proporcionando así de una forma gráfica una visión más clara del modelo.
- Permite definir estereotipos propios y personalizar las propiedades de los diagramas y de los elementos que componen estos diagramas.
- La **flexibilidad** de Together permite definir un proceso propio de modelado sin tener que seguir una normativa concreta. Esto permite que se pueda definir desde un proceso muy restrictivo a un proceso muy flexible.

3.4. MySQL-5.0.41

La base de datos ha sido diseñada con el **MySQL-5.0.41** y **MySQL-GUI-Tools para 5.0** que es la unificación de las siguientes aplicaciones administrativas:

- MySQL Administrator.
- MySQL query Browser.
- MySQL migration toolkit.
- MySQL Workbench.

MYSQL es un sistemas de gestión de base de datos es un software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y la aplicación la utilizan.

Las características principales de MySQL son:

- **Es un gestor de base de datos**. Una base de datos es un conjunto de datos y un gestor de base de datos es una aplicación capaz de manejar este conjunto de datos de manera eficiente y cómoda.
- **Es una base de datos relacional**. Una base de datos relacional es un conjunto de datos que están almacenados en tablas entre las cuales se establecen unas relaciones para manejar los datos de una forma eficiente y segura. Para usar y gestionar una base de datos relacional se usa el lenguaje estándar de programación SQL.
- **Es Open Source**. El código fuente de MySQL se puede descargar y está accesible a cualquiera, por otra parte, usa la licencia GPL para aplicaciones no comerciales.
- **Es una base de datos muy rápida**, segura y fácil de usar. Gracias a la colaboración de muchos usuarios, la base de datos se ha ido mejorando optimizándose en velocidad. Por eso es una de las bases de datos más usadas en Internet.
- Existe una gran cantidad de software que la usa.

4. Diseño de la aplicación

4.1. Diagrama de clase

En este apartado se incluye el diagrama de clases del paquete BBDD.Bean que no son imprescindibles para entender el diseño de la aplicación.
En la figura 71 se muestra el diagrama de clases del paquete BBDD.Bean.

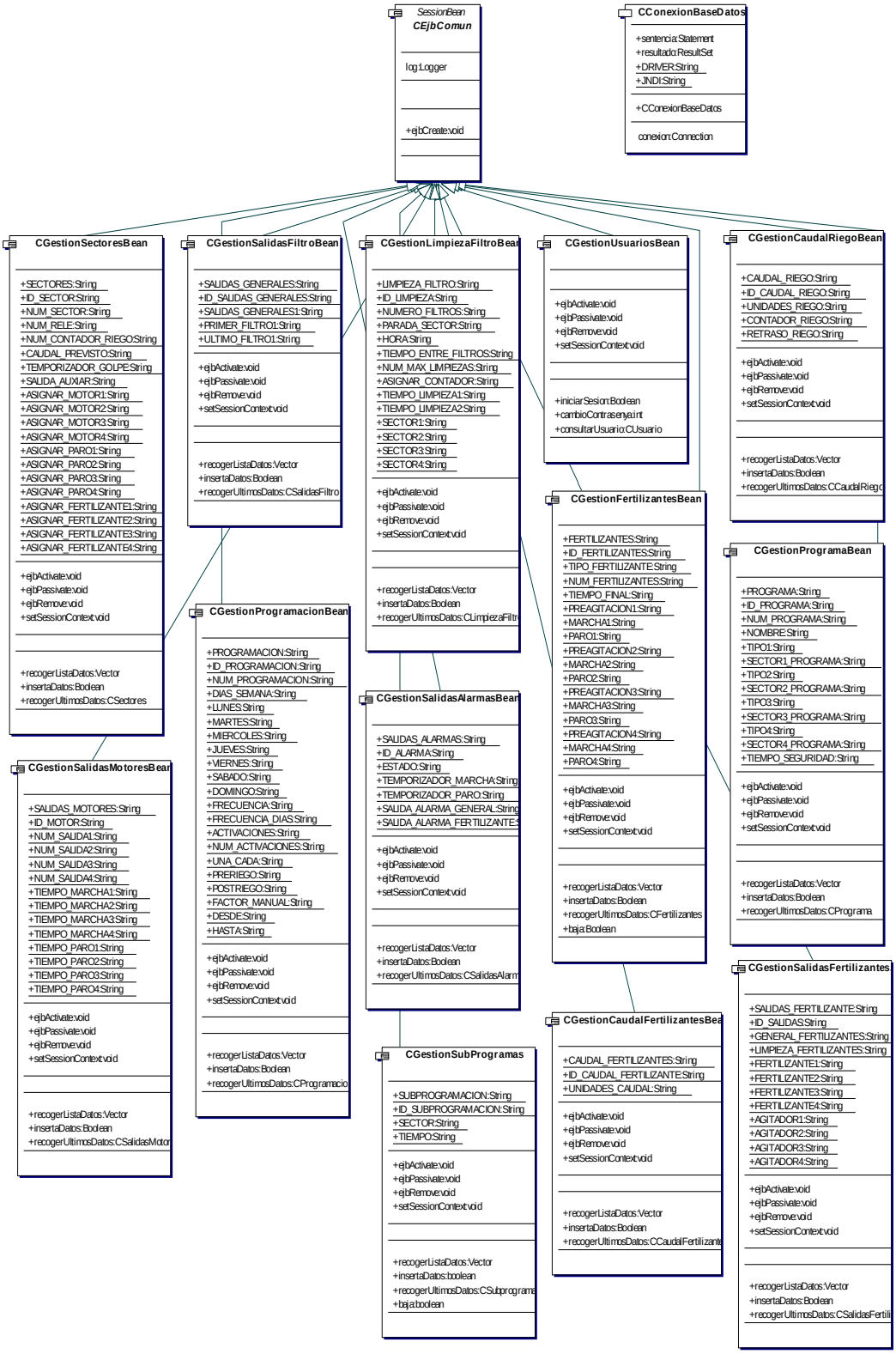

Figura 71 Paquete bbdd.Bean

4.2. Diagrama de secuencia

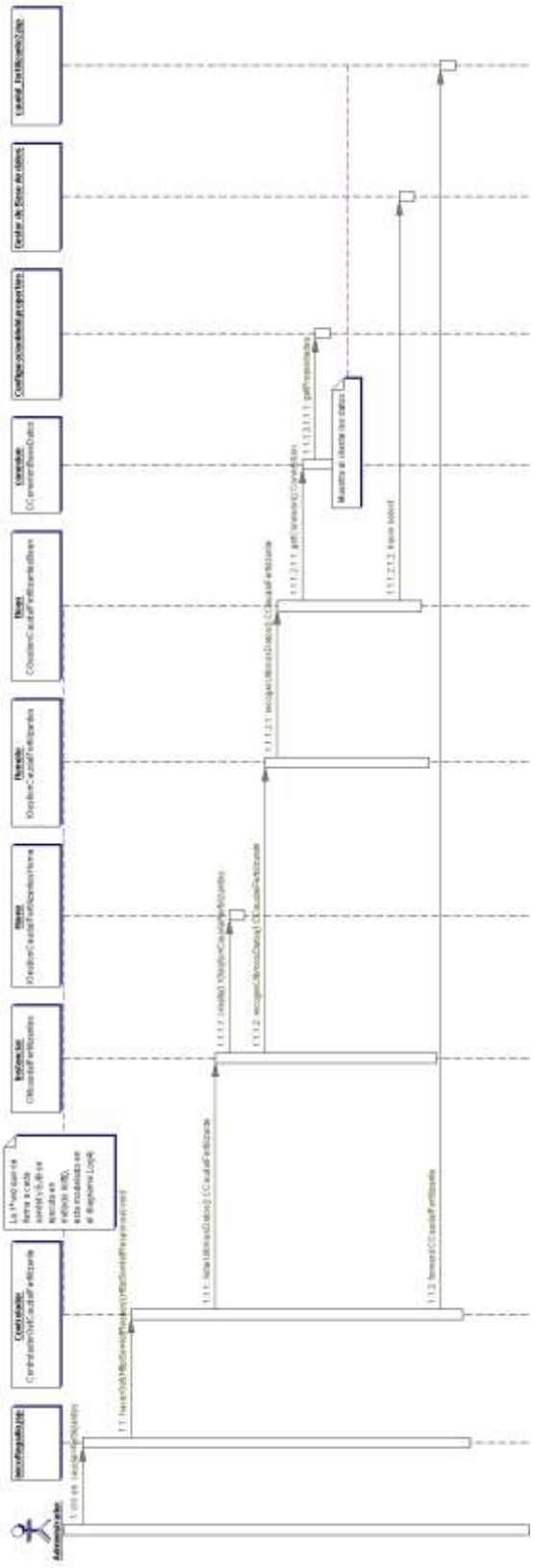

Figura 72 Diagrama de secuencia GetCaudalFertilizante

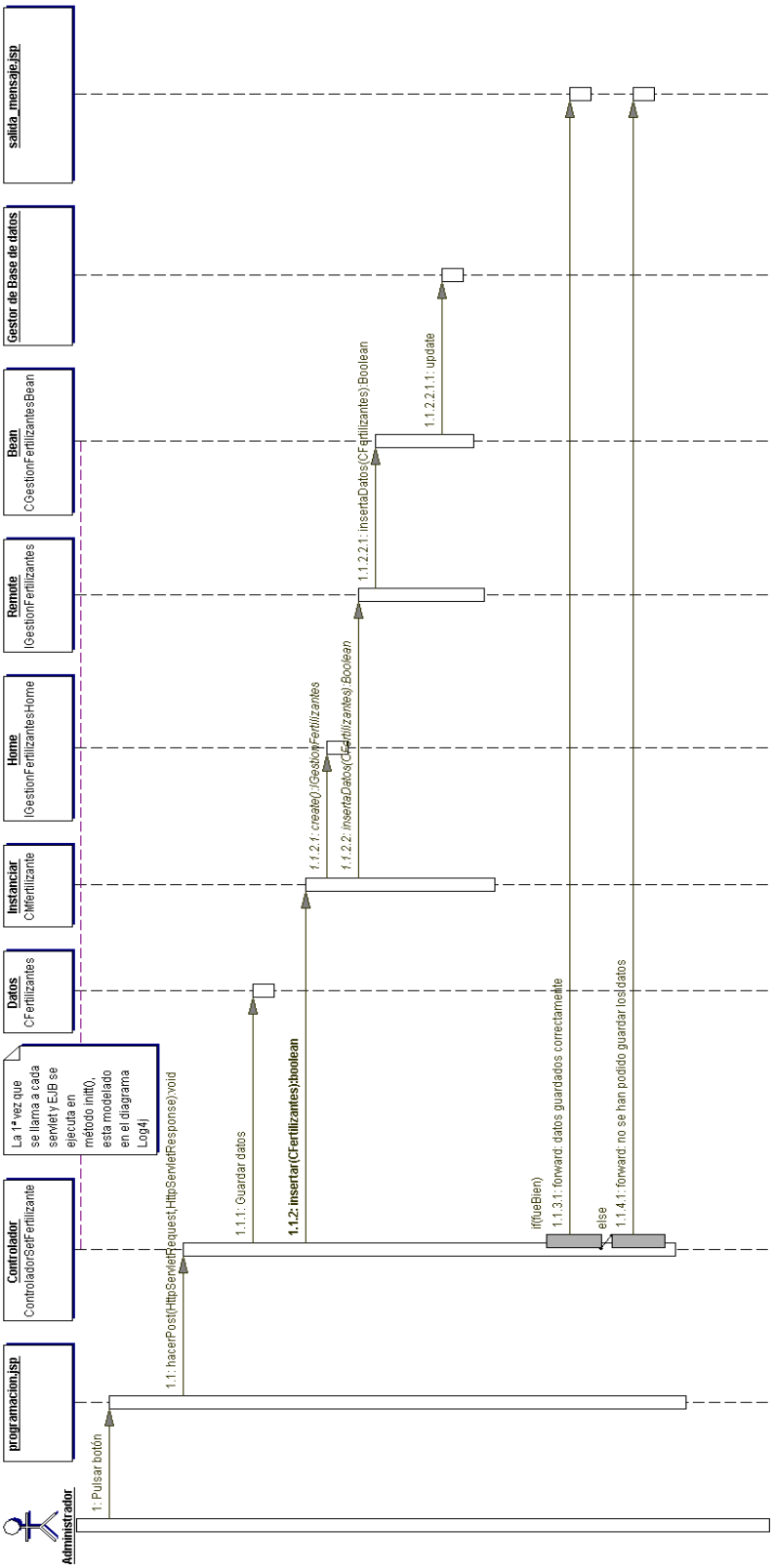

Figura 73 Diagrama de secuencia SetFertilizante

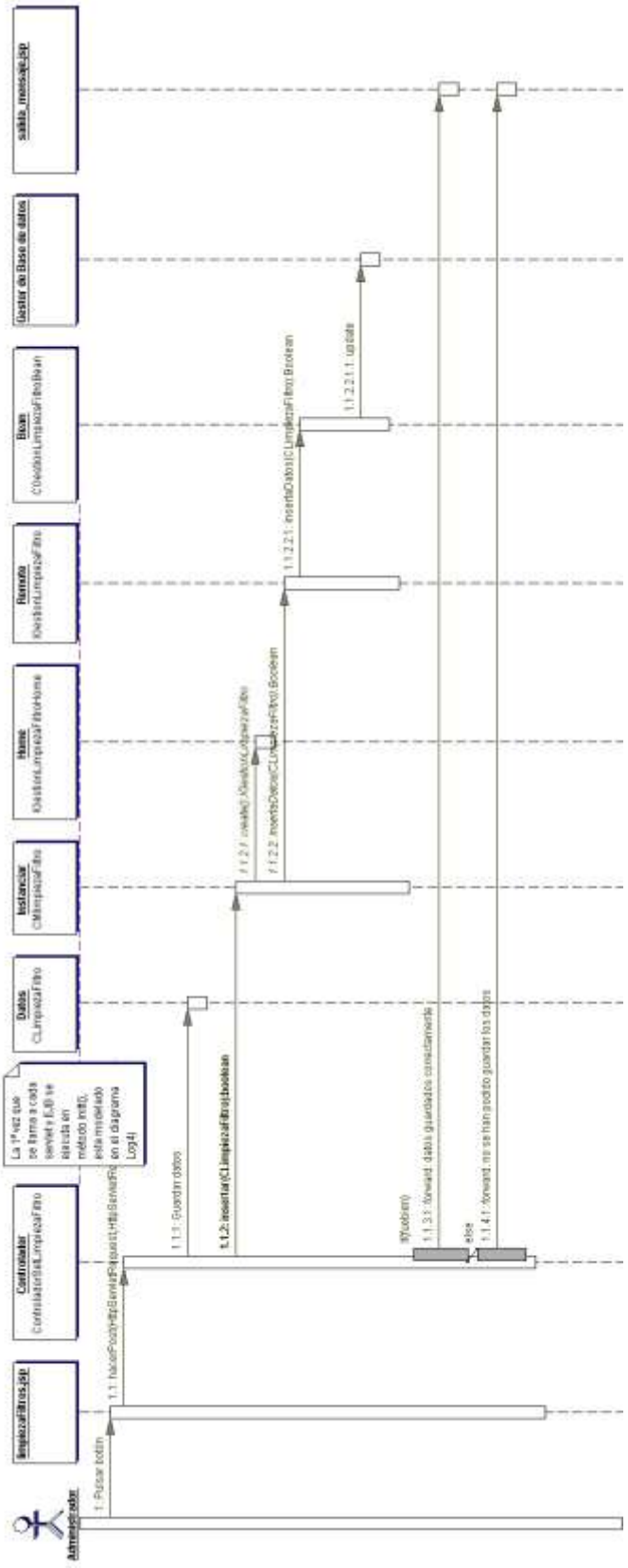

Figura 74 Diagrama de secuencia SetLimpiezaFiltro

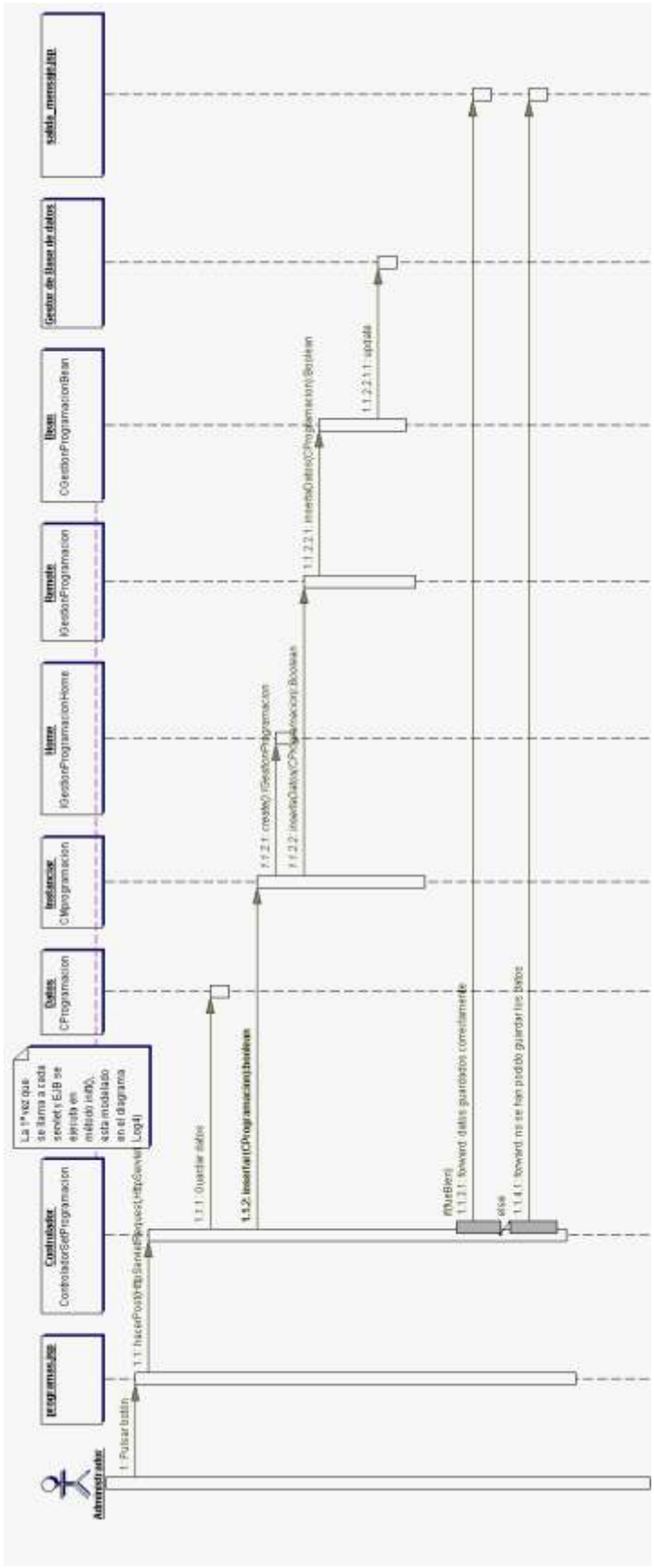

Figura 75 Diagrama de secuencia SetProgramacion

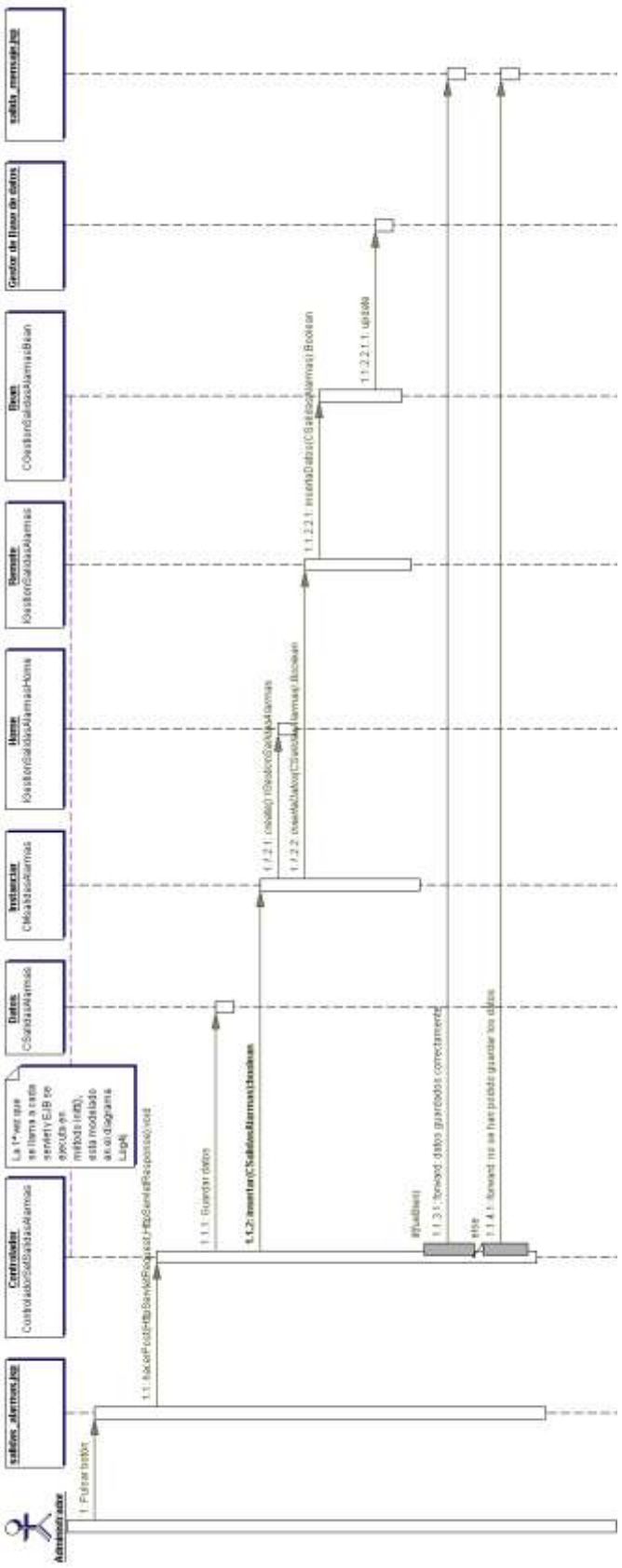

Figura 76 Diagrama de secuencia SetSalidaAlarmas

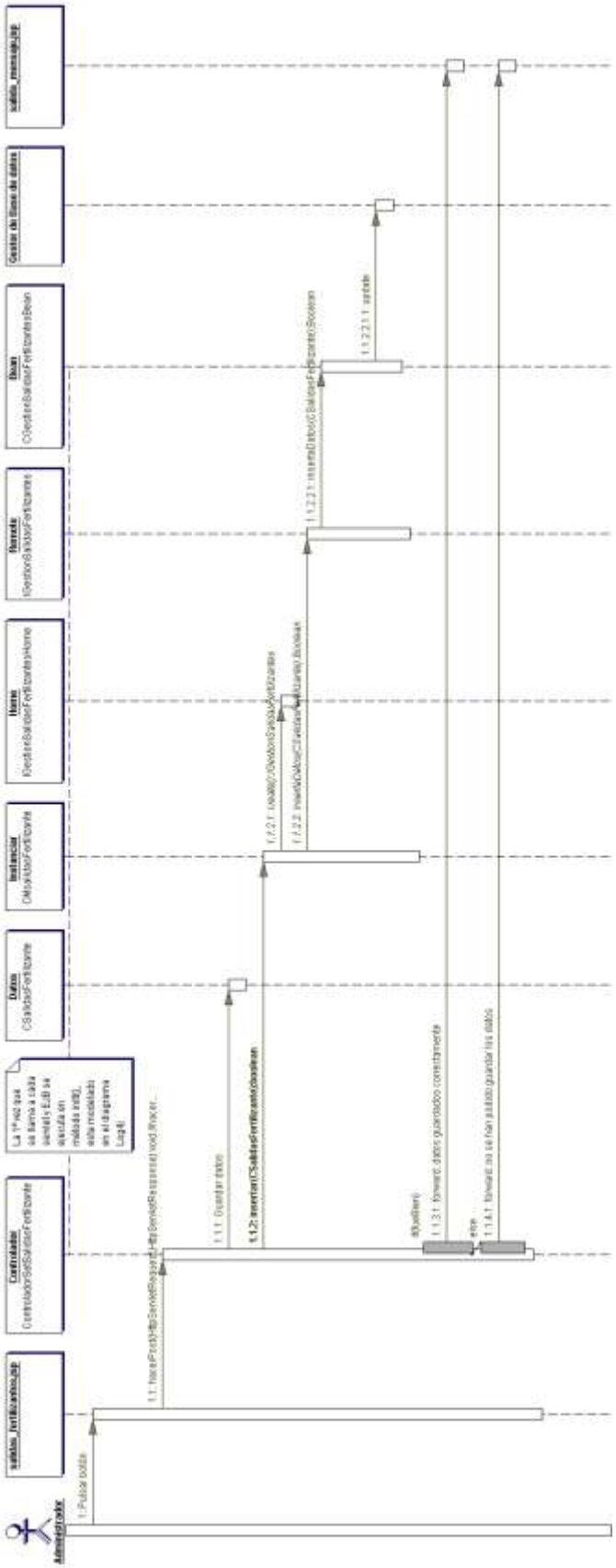

Figura 77 Diagrama de secuencia SetSalidasfertilizantes

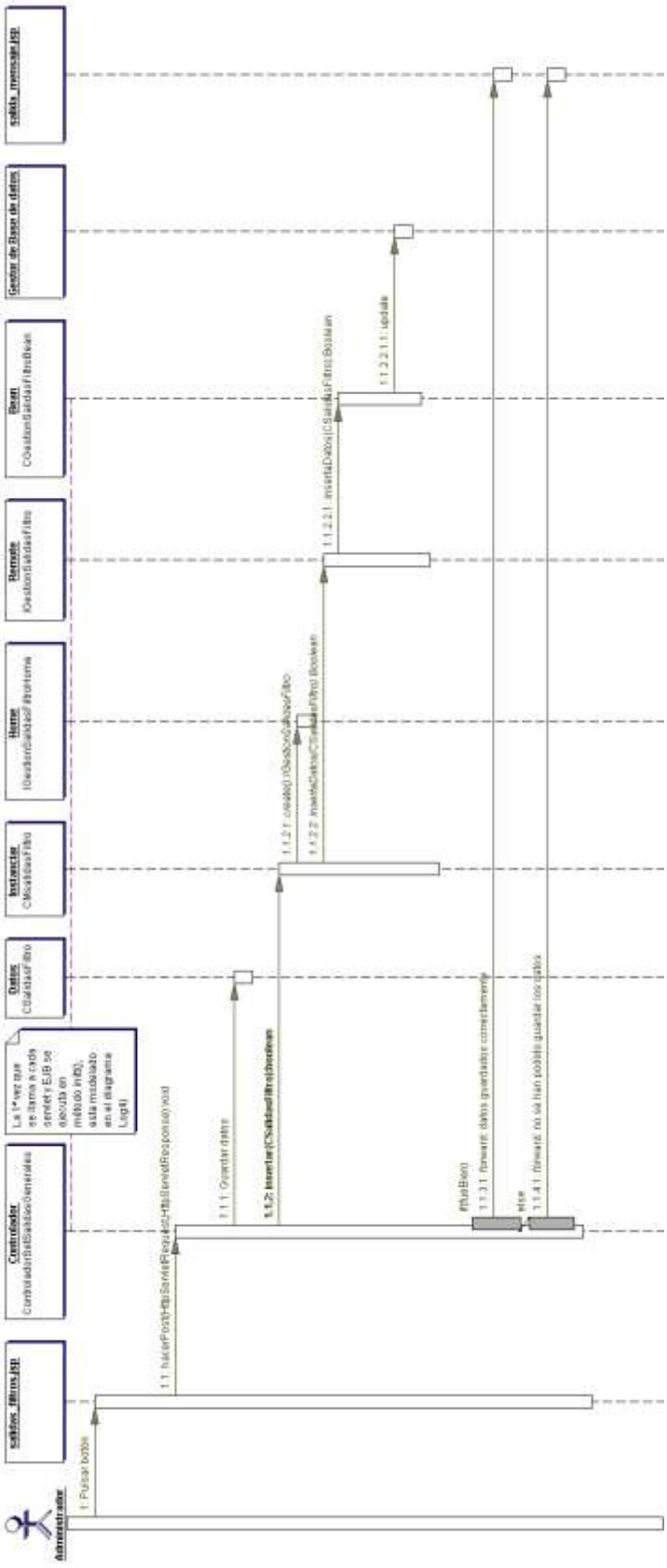

Figura 78 Diagrama de secuencia SetSalidasGenerales

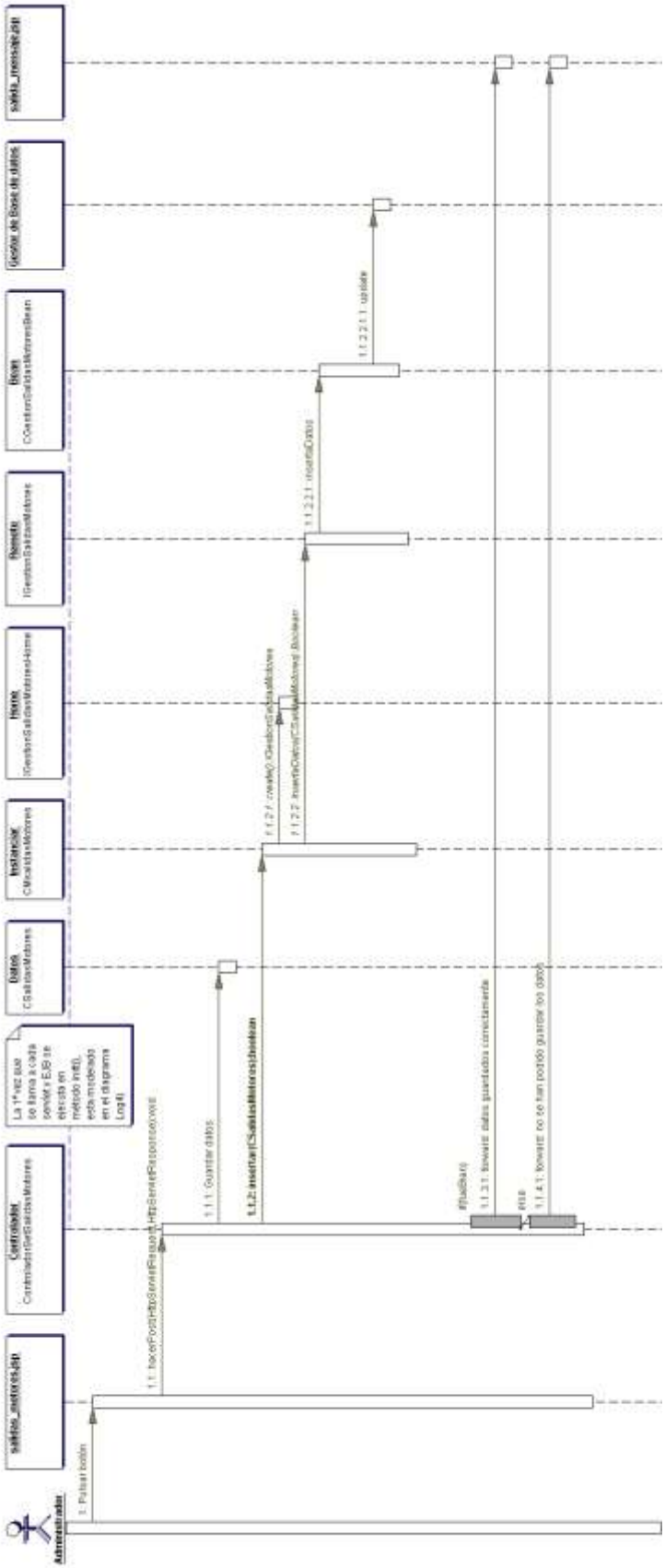

Figura 79 Diagrama de secuencia SetSalidaMotores

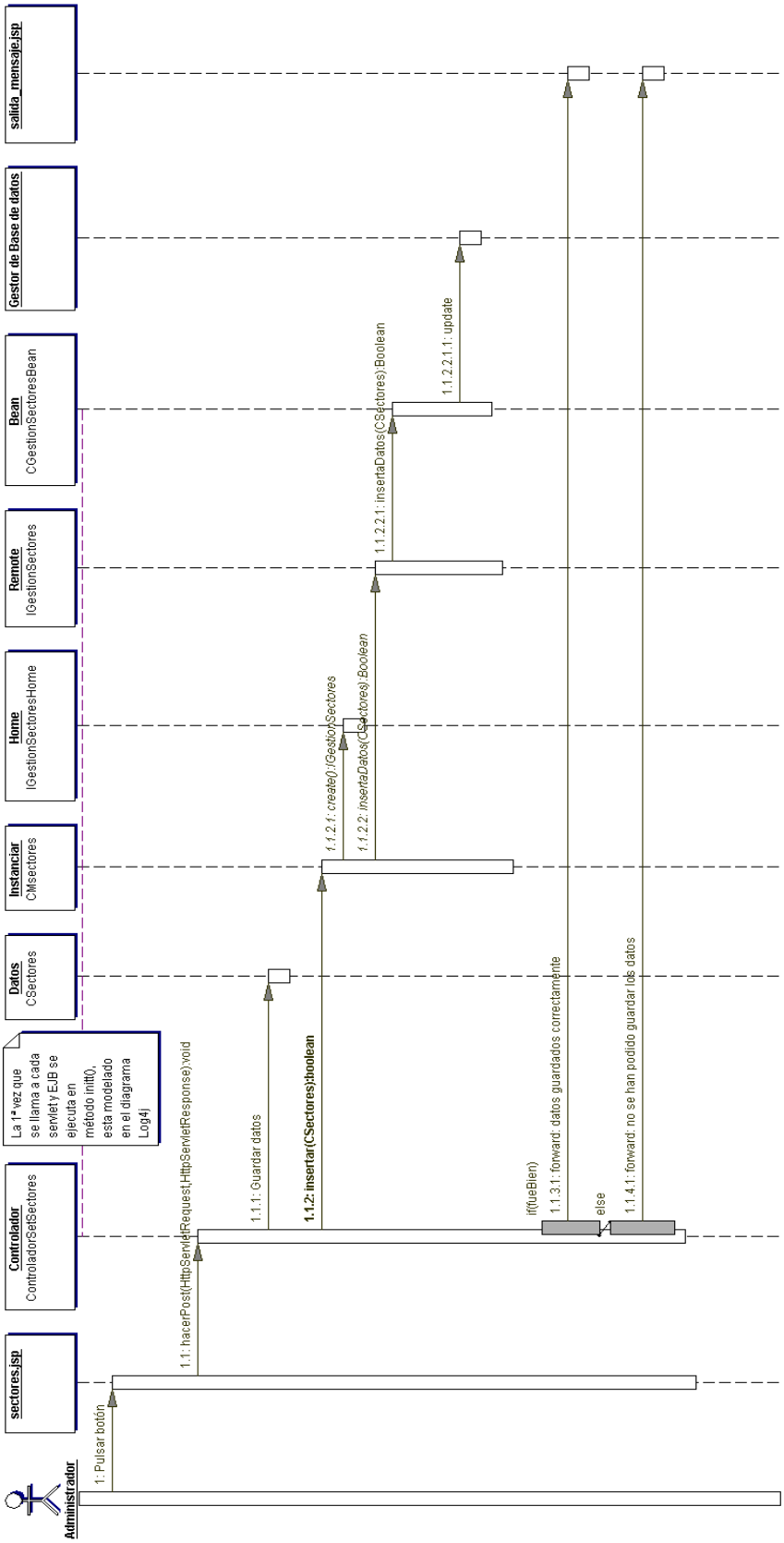

Figura 80 Diagrama de secuencia SetSectores

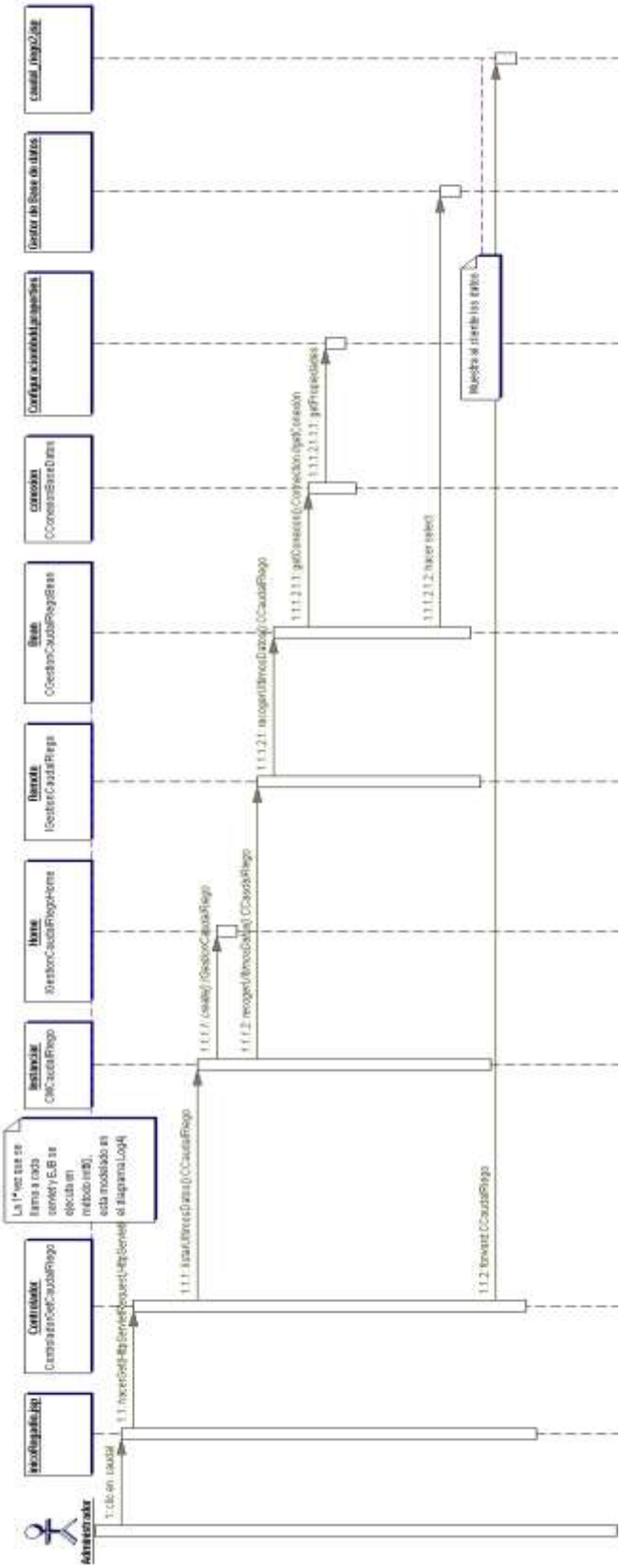

Figura 81 Diagrama de secuencia GetCaudaRiego

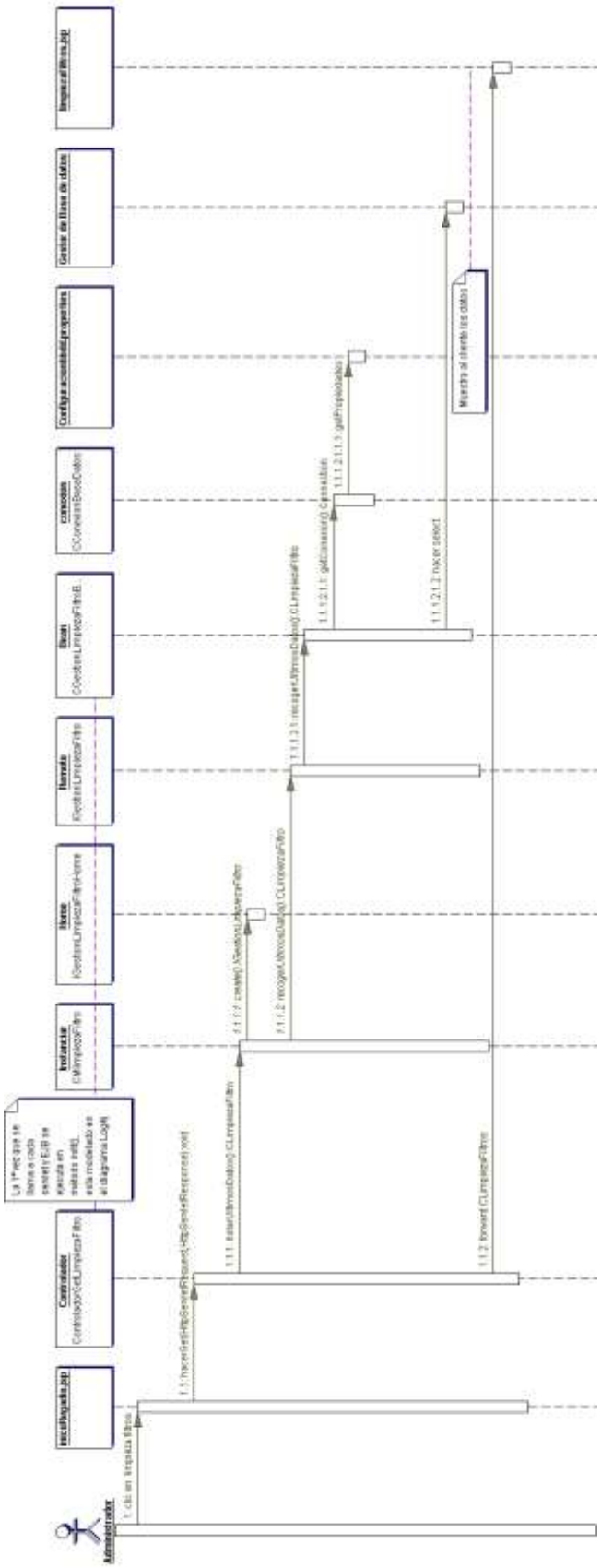

Figura 82 Diagrama de secuencia GetLimpiezaFiltro

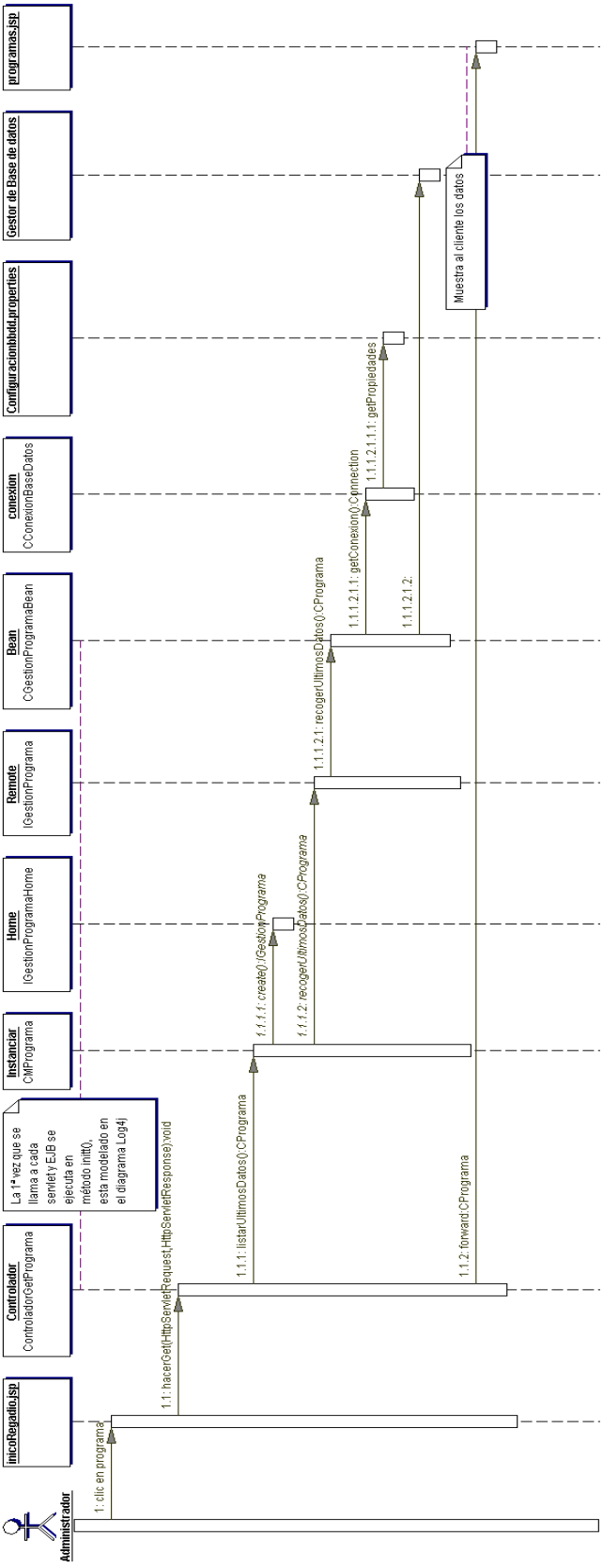

Figura 83 Diagrama de secuencia getPrograma

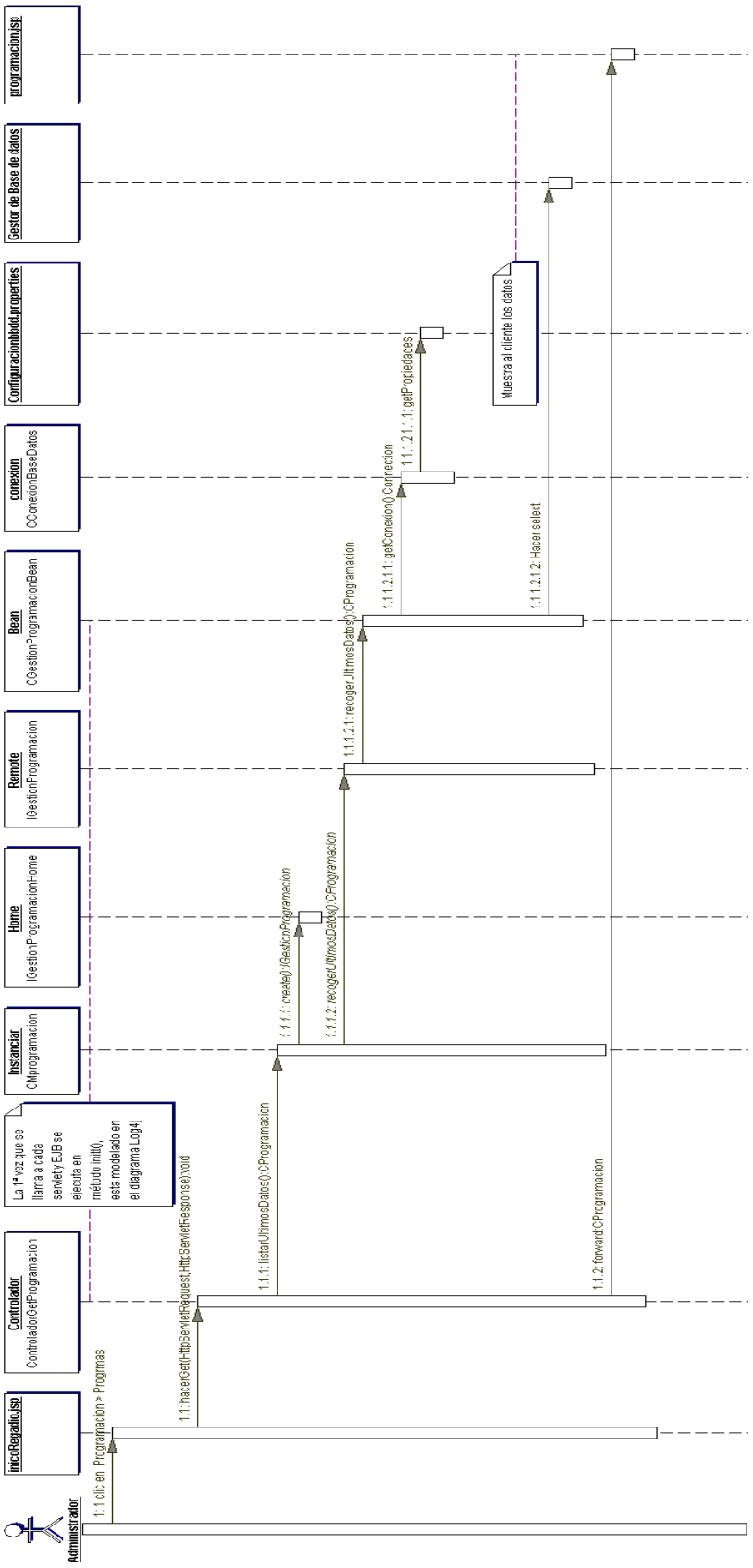

Figura 84 Diagrama de secuencia GetProgramacion

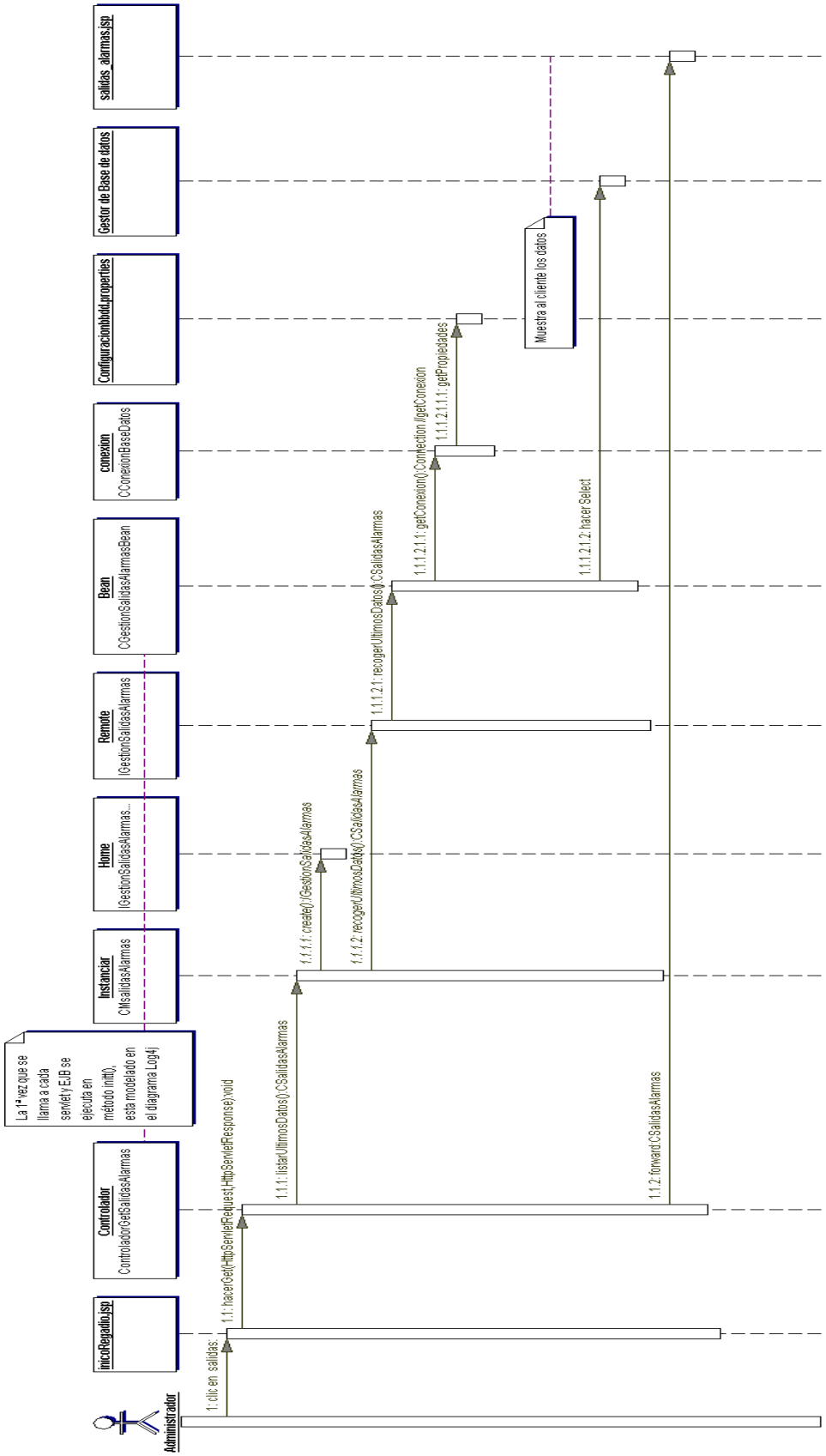

Figura 85 Diagrama de secuencia GetSalidaAlarmas

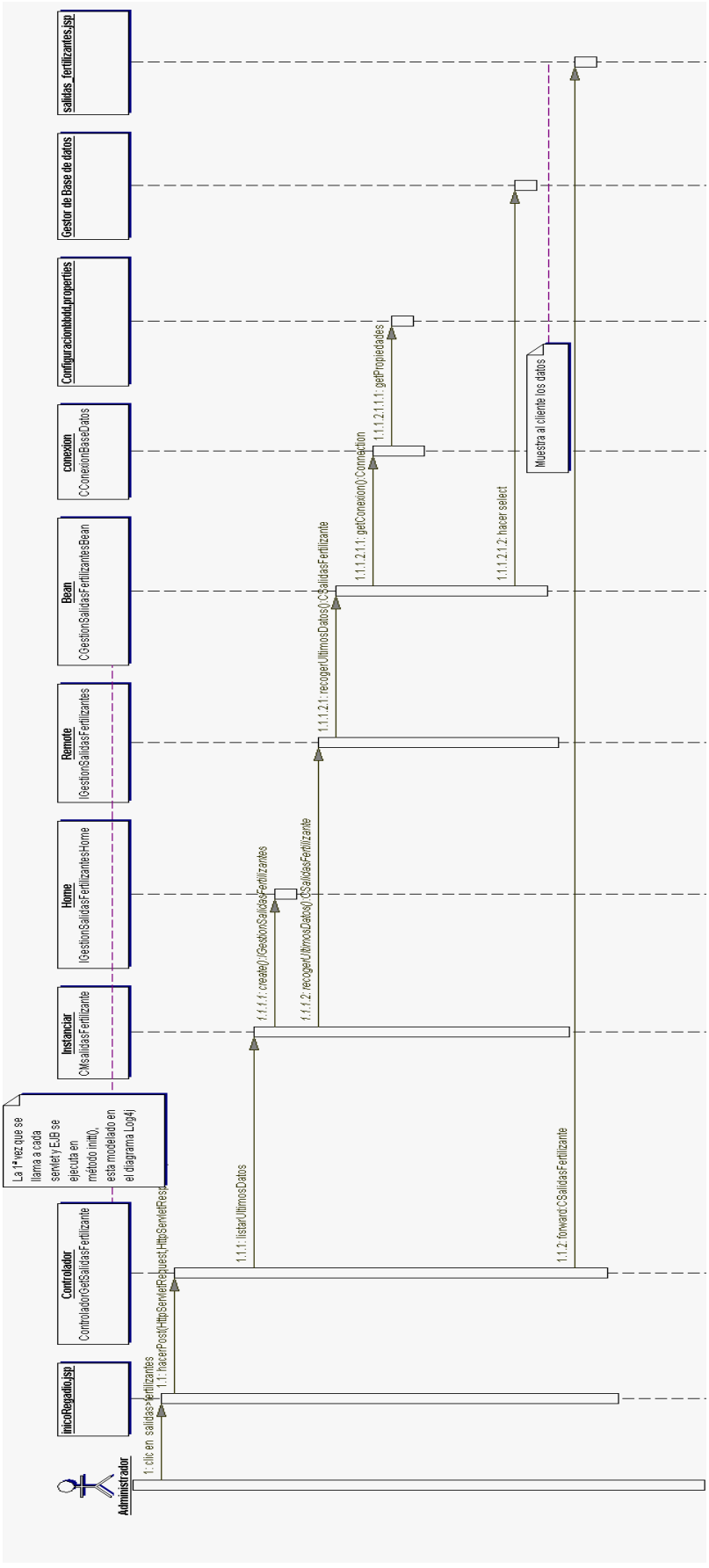

Figura 86 Diagrama de secuencia GetSalidasFertilizante

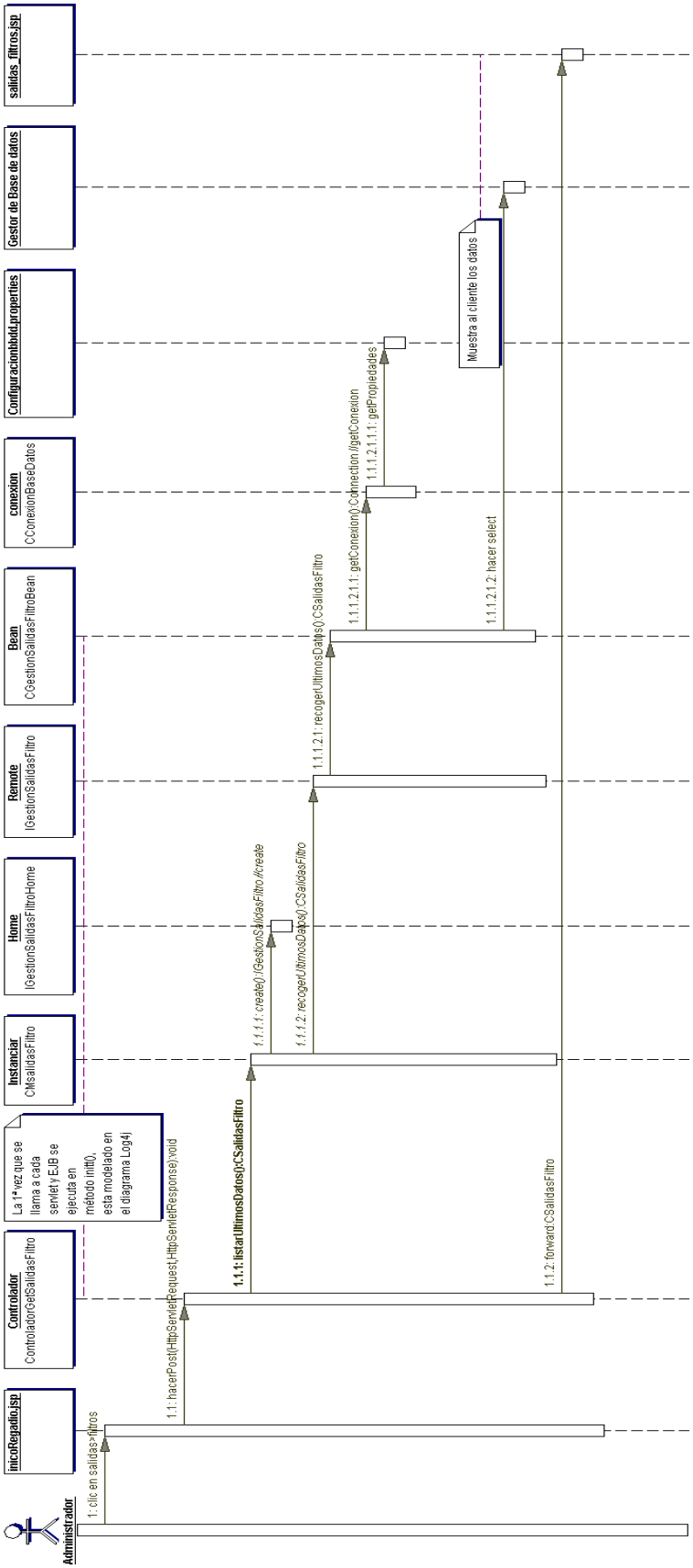

Figura 87 Diagrama de secuencia GetSalidaFiltro

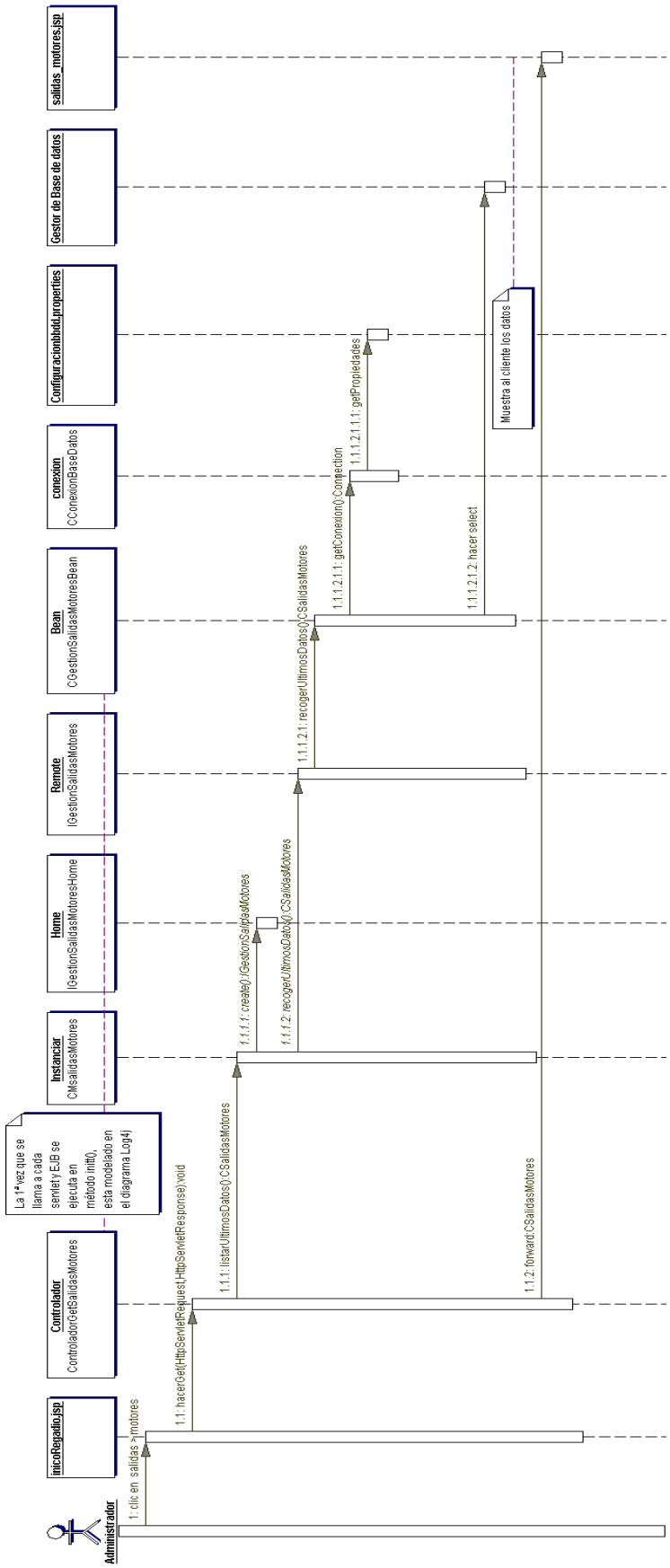

Figura 88 Diagrama de secuencia GetSalidaMotores

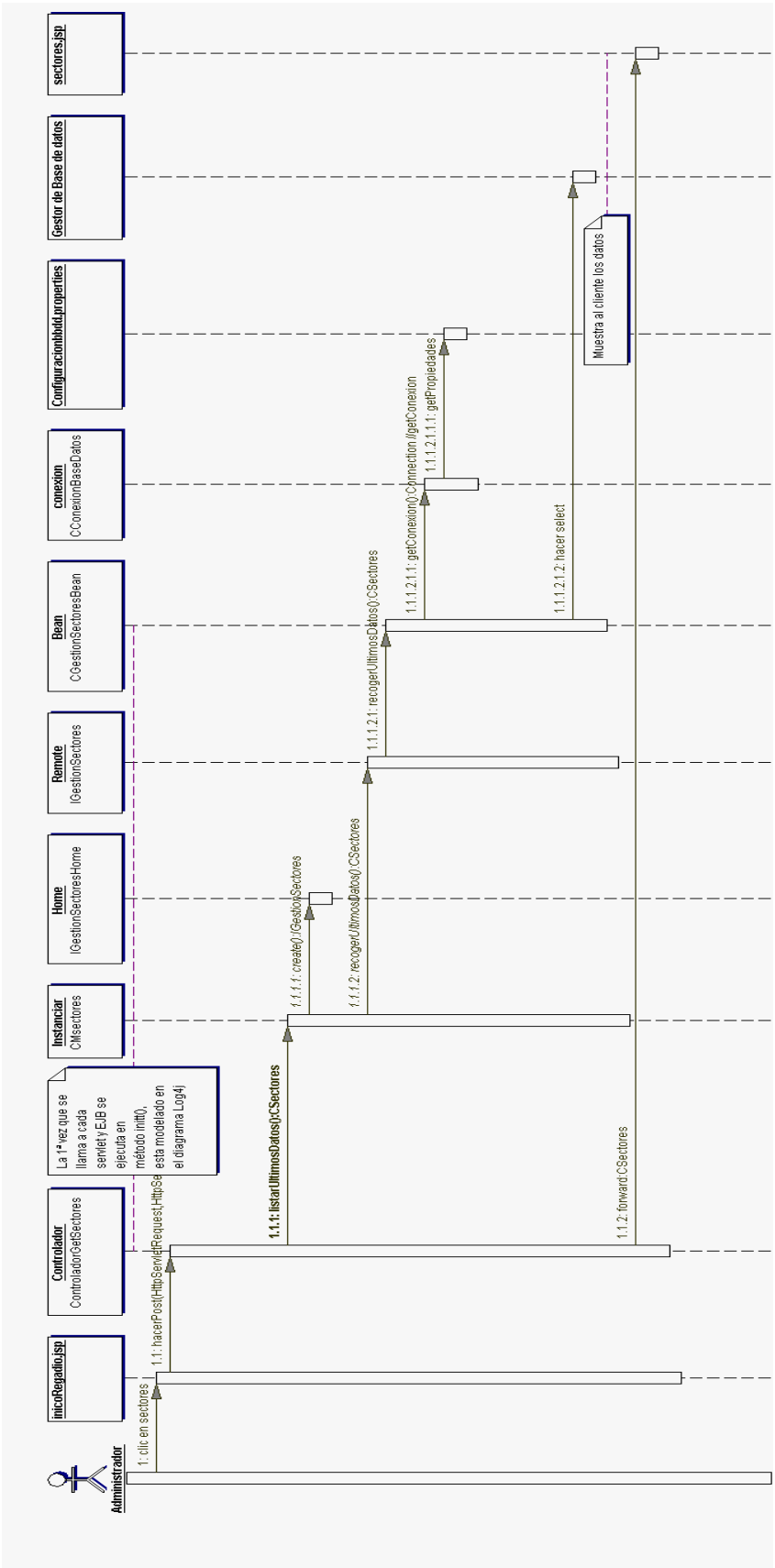

Figura 89 Diagrama de secuencia GetSectores

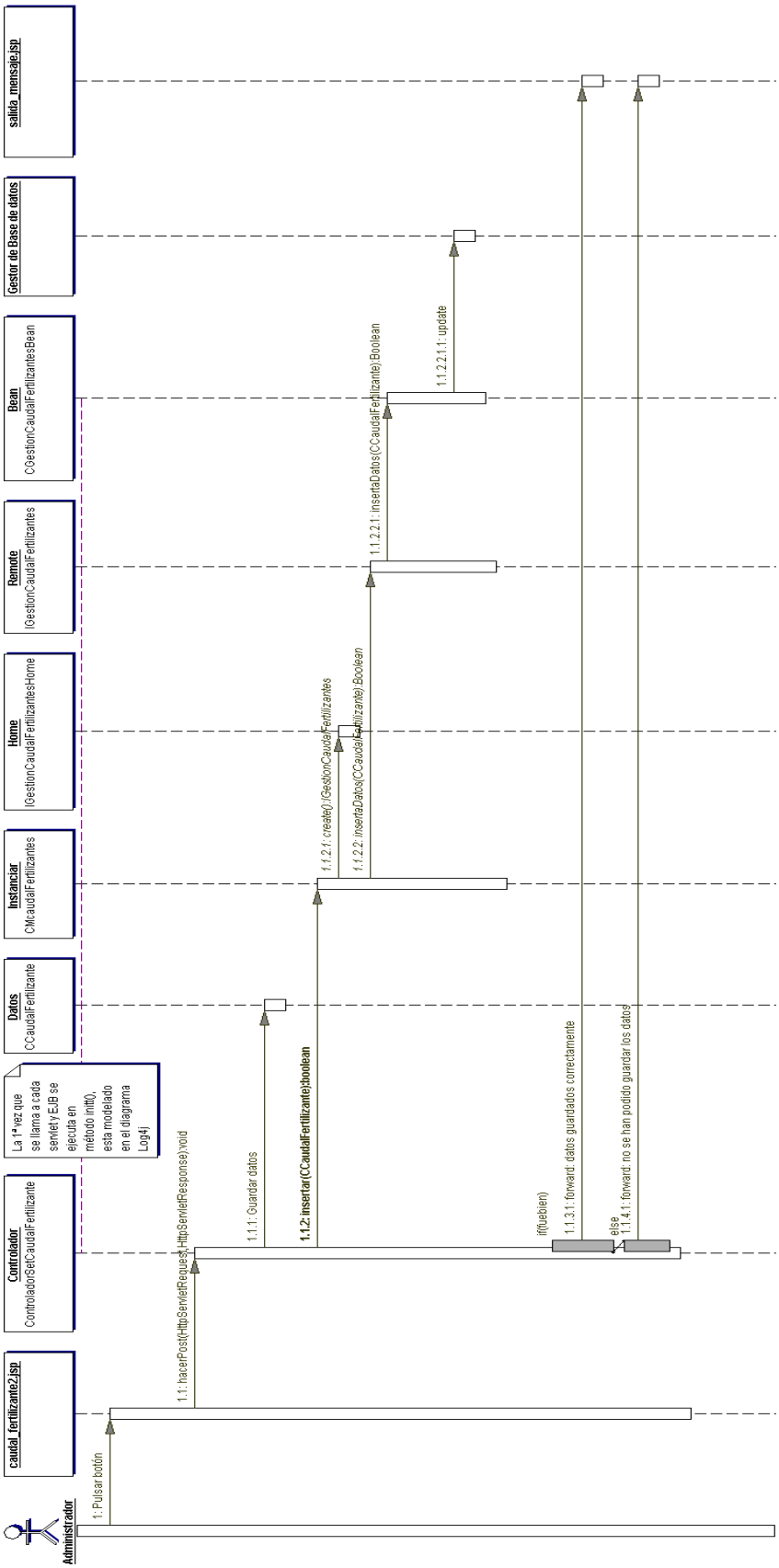

Figura 90 Diagrama de clases SetCaudalRegadío

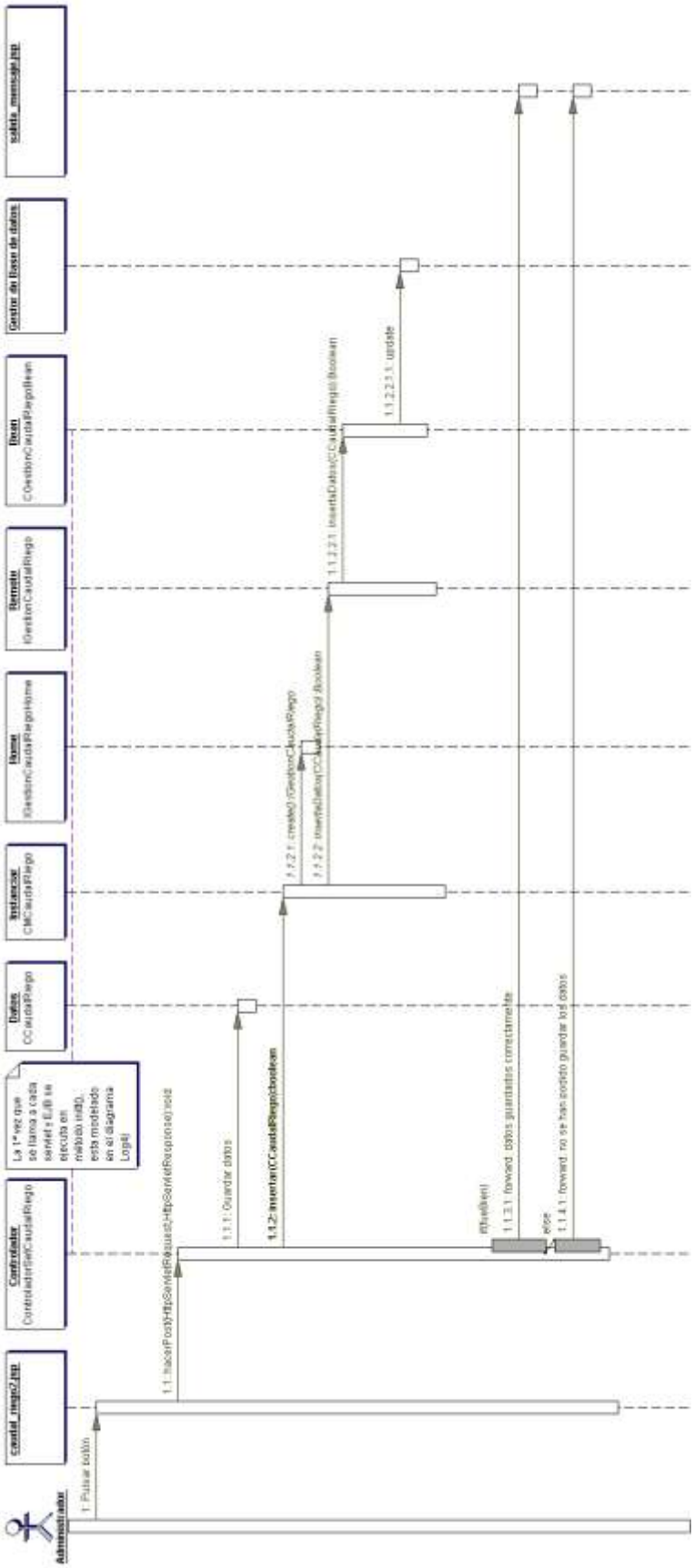

Figura 91 Diagrama de clases SetCaudalRegadío

5. Manual de usuario

En este apartado se describe un breve manual de usuario de la aplicación, algunas imágenes aparecen en el apartado 4.1 implementación visual, pero no se muestra al completo la aplicación Web.

La primera página es la figura Index.htm, una vez que el flash se ha cargado, se puede pulsar a “entrar”, si no se quiere esperar se puede pulsar sobre “saltar intro”

Figura 92 index.htm

Seguidamente se muestra la 73, el contenido de esta es la información sobre la empresa. Otra forma de acceder a ella es pulsando sobre el botón quienes somos.

Figura 93 Quienes.htm

Una vez se muestra la figura anterior se puede acceder a cualquier parte del menú, las páginas se recargan dentro del div inferior por lo que la url que siempre se muestra es /Frumansa/vitsa/quienes.htm. a otra parte de la aplicación que se puede acceder desde quienes somos es a la galería de fotos, en la figura 74 se muestra esta galería.

Figura 94 /Frumansa/vista/galeriadefotos/index.htm

Si se pulsa sobre cualquier imagen se puede ver la foto con mayor tamaño, como en la figura 75, si te fijas dependiendo de la posición de la imagen en la galería de fotos, aparecen unas flechitas en la parte superior, es para volver a la galería y las flechitas de la derecha e izquierda para pasar las fotos de la galería. Si el usuario desea volver a la pagina de Frumansa sólo tiene que pulsar en volver al espacio Web de Frumansa.

Figura 95 Fotos

A continuación, se muestra una imagen de cuando el usuario pulsa en Nuestros Productos del menú.

Figura 96 Productos

En la figura 77 se muestra la página “donde estamos”, en ella hay un plano y una breve descripción de las fincas.

Figura 97 Donde estamos

Cuando el usuario pulsa el botón “Contactar” se muestra la figura 78. Es un formulario para contactar con el administrador. La aplicación se encarga de enviar un correo al administrador con los datos introducidos.

Figura 98 Contacta.

Cuando se pulsa sobre el botón “Login” aparece la página de la figura 79, en este caso el administrador es quien debe de rellenar este formulario con usuario y contraseña para entrar a la zona de gestión de regadío.

Figura 99 Login.

Si el administrador no recuerda la contraseña, pulsando sobre el botón “No puedo acceder a mi cuenta” se muestra la pagina de la figura 80. Introduciendo el usuario y pulsando enviar, se envía un correo electrónico al administrador con la contraseña.

Figura 100 Recordar contraseña.

Si el administrador ha introducido bien el usuario y la contraseña, ha entrado a la zona privada, al gestor de regadío (figura 81). En esta zona siempre está el menú superior de “Gestor de Regadío”, “Cambiar contraseña” y “Cerrar sesión”. El menú que hay en izquierda de la figura 81 “Parámetros”, es para la gestión de regadío.

Figura 101 Gestor de Regadío.

Cuando se pulsa sobre “Fertilizantes” se muestra los parámetros a configurar (figura 82), se muestran los que se introdujeron la última vez que se configuraron, con la posibilidad de modificarlos.

Figura 102 Gestión Fertilizantes

Para la gestión de limpieza de filtros es la figura 83.

Figura 103 Gestión de Filtros

En el menú de parámetros, si se pulsa sobre salidas, el menú se despliega 4 submenus sobre las salidas generales. Salidas de alarmas se puede ver en la figura 84. Salidas de fertilizantes en la figura 85, salidas de filtros en la figura 86 y salida de motores en la figura 87.

Figura 104 Salida de alarmas.

Figura 105 Salida de Fertilizantes.

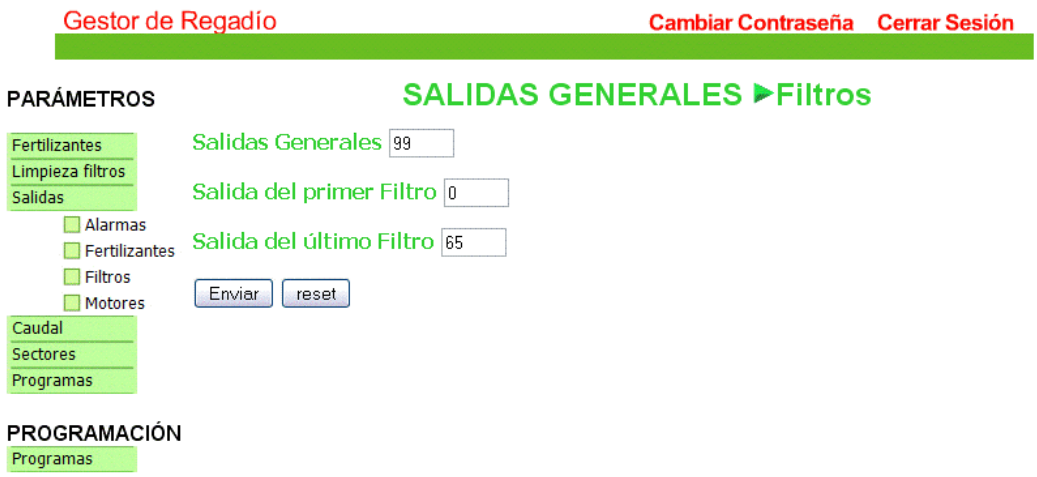

Figura 106 Salida de Filtros.

Figura 107 Salida de motores.

Pulsando sobre “Caudal” del menú de la derecha, también se despliega dos submenús, gestión de caudal de riego (figura 88) y de caudal de fertilizantes (figura 89)

Figura 108 Caudal de Riego.

Figura 109 Caudal de Fertilizantes.

Para la gestión de sectores es la figura 90.

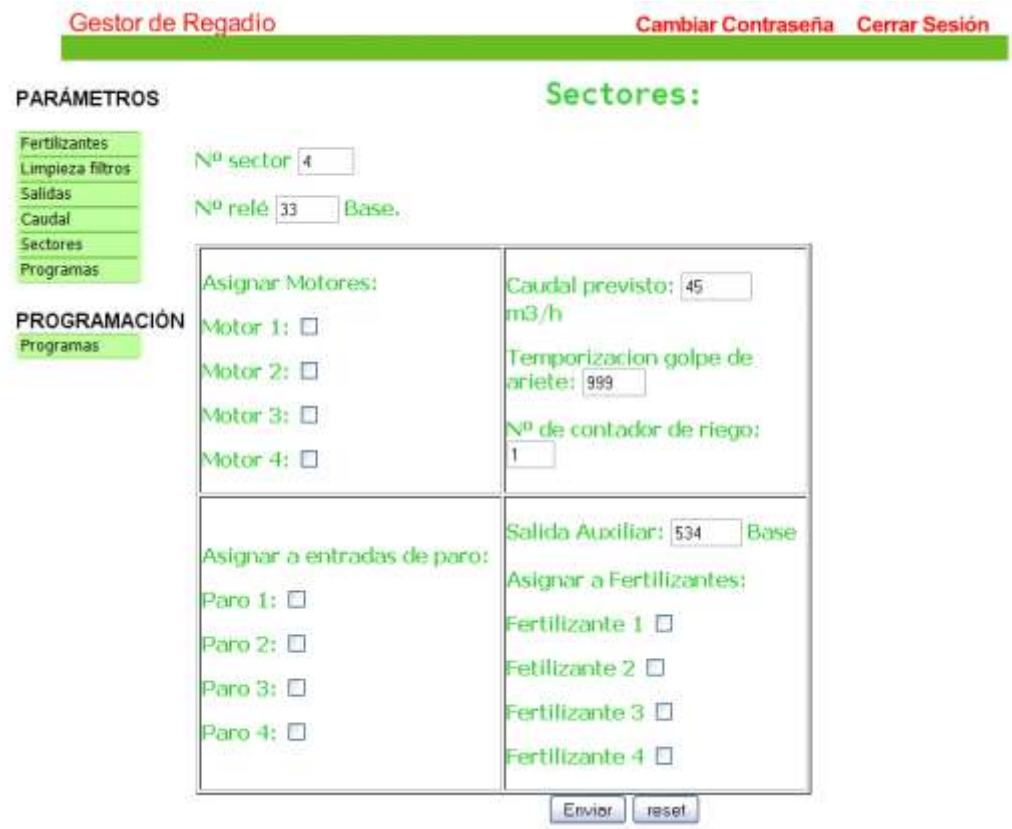

Figura 110 Gestión de Sectores.

Si pulsas en el menú parámetros “programa” se puede configurar el programa con todos los parámetros guardados anterior mente modificados. En la figura 91 se muestra esta configuración.

Figura 111 Parámetros programas.

Si pulsas sobre el botón programas de programación del menú de la izquierda, aparece la pantalla de la figura 92.

Figura 112 Programación

Por último, el administrador puede cambiar la contraseña, pulsando donde pone “cambiar contraseña” que aparece en el menú de la parte superior, la pantalla que se muestra es la figura siguiente:

Figura 113 Cambio de contraseña