# TESIS 1

# FIA PYT-2009-0259

UNIVERSIDAD CATOLICA DE LA SANTISIMA CONCEPCION

Facultad de Ingeniería

Ingeniería Civil Informática


# DESARROLLO DE UNA INTERFAZ DE MONITOREO Y PLANIFICACION DE RIEGO

**YASNA GUTIERREZ NEIRA**

INFORME DE PROYECTO DE TÍTULO PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INFORMÁTICO

**Profesor Guía**

Carlos Hernández

Concepción, Enero del 2012

## Resumen

El uso óptimo de los recursos es un objetivo importante en Ingeniería. En la Ingeniería Agrícola se necesitan métodos para que, entre otros recursos, se haga un uso eficiente del agua de riego. Esta necesidad ha llevado a introducir a la informática en este rubro.

En este contexto, un grupo de profesionales de la Universidad Católica de la Santísima Concepción está en proceso de crear un sistema para controlar la irrigación en plantaciones de arándanos, basado en técnicas de inteligencia artificial, a partir de las mediciones en el suelo, hechas por una red de sensores inalámbricos en la plantación.

El aporte de este proyecto es la creación de una interfaz de usuario para un sistema de monitoreo y planificación de riego. En este sistema los agricultores pueden acceder remotamente a observar y modificar planes de riego y fertilización, configurar los elementos del huerto y monitorear el nivel de humedad, integrando los distintos intereses y niveles de educación en materia agrónoma, como también distintos niveles de conocimientos del uso de las tecnologías de la informática de los usuarios finales.

Esta interfaz fue desarrollada buscando integrar atributos de calidad, entre ellos se consideró la **portabilidad**, dada la compatibilidad de la interfaz con distintos navegadores; facilidad de **mantención**, gracias a la tecnología de desarrollo (Tiles); **atractiva al usuario,** por el uso de una potente biblioteca gráfica (jQuery) y con **bajo acoplamiento** con respecto al modelo y controlador, gracias a la arquitectura MVC y al Framework Struts 2.

Como conclusiones se recomienda el uso del modelo de desarrollo evolutivo para refinar los requisitos funcionales a través de las diferentes versiones. Además se comprobó la importancia de incluir tempranamente a usuarios en el desarrollo, así los cambios pueden realizarse en una etapa temprana, teniendo menor repercusión en el resto de componentes del sistema.

## Abstract

Optimal use of resources is an important engineering's objective. Agricultural Engineering methods are needed to, among other resources, make efficient use of irrigation water. This need has led to introduce information technology in this area and deploy a set of information technologies in the fields.

In this context, a group of professionals from the *Católica de la Santisima Concepción's* University are in process of creating a system to efficiently control the cranberry plantation irrigation, based on artificial intelligence techniques, from measurements on the ground, the environment and plants, made by a wireless sensor network on the plantation.

The contribution of this project was creating an interface or view component of Model View Controller architecture for a system of monitoring and planning of irrigation. In this system, farmers can access remotely monitor and modify plans for irrigation and fertilization, configure the elements of the garden and monitor the moisture status, integrating the different interests, different levels of education in agronomy as well as different levels of knowledge use of information technologies for end users.

This interface was developed seeking to integrate quality attributes, including portability was considered, given the interface compatibility with different browsers, ease of maintenance, thanks to technology development (Tiles), appealing to the user, using a Powerful graphics library (jQuery) and loosely coupled to the model and controller, thanks to the MVC architecture and Struts 2 Framework.

In conclusion we recommend the use of evolutionary development model to refine the functional requirements through the different versions. In addition it was found early the importance of including users in the development and changes can be made at an early stage, have less impact on other system components.

# Agradecimientos

Porque ustedes, mi familia, me apoyaron desde el primer día que emprendí esta carrera, en las buenas y las malas, les entrego mis agradecimientos. Principalmente a usted mamá que siempre me anima diciéndome *"hija, usted siempre puede"* y a usted papá, gracias porque con el sudor de su frente me ha ayudado a llegar a esta meta. También gracias Rodrigo Lozano por tu gran apoyo emocional, por ser mi sostén en muchos momentos difíciles de esta etapa y amarme como yo te amo.

*"...Mira que te mando que te esfuerces y seas valiente: no temas ni desmayes, porque Jehová tu Dios estará contigo..."*
*Josué 1.9*

*Dedicada a mi creador, por hacerme de mi un testimonio de su existencia.*

# Contenido

# Índice de Figura

# Índice de Esquemas



# Índice de Tablas

## Nomenclatura y abreviaciones

AD= Análisis de datos

AP= Agricultura de Precisión

API= Application Programming Interface

ERS= Especificación de Requisitos del Sistema

GIS= Geographic Information System

*GPS = Global Positioning System*

IEEE= Institute of Electrical and Electronics Engineers

JSP= JavaServer Pages

MVC = Modelo Vista Controlador

RUT= Rol Único Tributario

SA= Software Architecture

SVG= Scalable Vector Graphics

ISO = International Organization for Standardization

W3C= World Wide Web Consortium

# Capítulo I: Introducción

## *1.1. Presentación del Tema*

Este proyecto es parte de un proyecto más grande. Este está compuesto por un equipo multidisciplinario, que pretende crear herramientas para el uso eficiente e informado del agua en plantaciones de arándanos mediante el uso de Tecnologías en las áreas de agricultura de precisión. Las herramientas (software) se basan en las técnicas de Inteligencia Artificial, donde el software de planificación produce planes de riego a partir de las mediciones relevantes en el suelo, el ambiente y las plantas, realizadas por una red de sensores inalámbricos (Ministerio de Agricultura, 2008).

Debido a la variabilidad espacial y temporal en la plantación, los planes generados inicialmente por el software, pueden no ser de total agrado del agricultor. Por tanto, a través de una interfaz es posible modificar el plan para adecuarlo a las condiciones hídricas que se desean. El plan de riego definitivo, se ejecutará automáticamente mediante el control de las válvulas. A su vez el agricultor tendrá la opción de monitorear las variables relevantes de la plantación, registrar y observar planes de fertilización y administrar los elementos del huerto.

En este contexto, este proyecto tiene como objetivo el construir las funcionalidades descritas anteriormente en una interfaz de usuario remota (a través de Internet), donde el agricultor interactúa con el sistema y puede observar y modificar el plan de riego, obtener estadísticas y ver el estado actual de las mediciones de los sensores.

En la interfaz interactúan distintos tipos de usuarios, con distintos niveles de conocimiento, por lo que una parte importante del trabajo realizado consistió en estudiar a los usuarios para entregar información con distintos niveles de procesamiento, acordes a sus capacidades de comprensión.

## 1.2. *Objetivo General*

Implementar una interfaz de monitoreo y planificación de riego remota, que permita desplegar información para los distintos niveles de conocimiento de usuarios.

## 1.3. *Objetivos Específicos*

- Estudiar los procesos productivos de la agricultura y la misión de la informática en ellos.
- Especificar requisitos (funcionales y no funcionales) del sistema de monitoreo y planificación de riego remota.
- Estudiar y seleccionar distintas herramientas informáticas (Open Source o Freeware) a utilizar en el desarrollo de la aplicación.
- Diseñar e implementar una Interfaz, organizada en niveles de conocimiento, desde las distintas perspectivas, roles y desempeños de los usuarios.
- Ejecutar un plan de pruebas y realizar las mejoras pertinentes.

## 1.4. *Justificación del problema*

En general, una buena organización de los datos garantiza disponer de la información precisa, en el instante preciso, permitiendo disminuir el grado de incertidumbre al momento de tomar decisiones. Particularmente en la agricultura, una buena organización de los datos permite a los agricultores tomar importantes decisiones, que influirán directamente en su cantidad y calidad de producción.

Por esto es fundamental que el agricultor interactúe adecuadamente con el sistema y que obtenga información que pueda comprender, lo que justifica el desarrollo una interfaz que apuntará a tres tipos de usuarios; con distintos roles, tareas y niveles de información. Un usuario con alto conocimiento de agricultura (usuario controlador), que no necesita que el sistema le interprete ciertos datos, pues le es fácil tomar una decisión de riego a partir de ellos y por otro lado se tiene un usuario con menor conocimiento (usuario monitor), quien perderá mucho tiempo tratando de interpretar datos, por lo cual le es cómodo que los datos tengan una alta interacción con el sistema y simplemente le muestre una acción a seguir. El tercer usuario tiene un rol de administrador de huerto (usuario administrador de huerto) y sus conocimientos son muy variados, por lo cual él necesita que ambos conceptos estén disponibles.

Por otro lado, los usuarios del sistema utilizan distintas plataformas de navegación, tanto navegadores de dispositivos móviles (celulares, Ipad, Pda, etcétera), como navegadores usados en computadores, por lo cual es importante desarrollar una interfaz web que priorice la compatibilidad con distintas plataformas, normalizada con recomendaciones internacionales de desarrollo web (W3C, IEEE).

Por último, la importancia de la creación de una interfaz de monitoreo y planificación remota radica en la reducción de los costos de producción que implica monitorear remotamente, dada la ruptura de barreras geográficas y a la disminución de trabajo.

## 1.5. *Delimitación del Problema*

El proyecto se limitó a desarrollar una interfaz de usuario para el monitoreo y planificación de riego. Todo el sistema de sensores y su conectividad se encuentra previamente desarrollado por un grupo de profesionales de la Universidad Católica de la Santísima Concepción.

En este proyecto se habla del desarrollo de una *"interfaz"*, puesto que el desarrollo del Sistema de Monitoreo y Planificación de riego se realizó bajo la arquitectura Modelo Vista Controlador (MVC), donde el desarrollo del Modelo y del Controlador son tareas de otro integrante del equipo, por lo cual el presente proyecto no se interiorizará en dicho desarrollo.

Dentro del desarrollo de la interfaz se incluyó una especificación de requisitos, realizada en conjunto con el desarrollador de Modelo y Controlador del sistema final, con quien a su vez, se analizaron las herramientas ya existentes de modo de tener una referencia de trabajo. También fue necesario identificar los roles de los stakeholder dentro del sistema, lo que se abordó mediante entrevistas, tanto a las personas representantes de cada rol, como a expertos en el tema. Así se logró definir cuál es información relevante para cada uno de ellos y cuáles serán las herramientas de análisis de los datos entregados por los sensores.

En la etapa del diseño del software uno de los puntos más importantes es la usabilidad, por lo cual se estudió en esta etapa algunas técnicas de usabilidad que ayudaron con el propósito. Para esto se recurrió al asesoramiento de un diseñador gráfico; también se trabajó en conjunto con los usuarios que conocen en profundidad el tema.

## *1.6. Metodología*

Para llevar a cabo la implementación del sistema se realizaron las siguientes etapas metodológicas, acordes con los objetivos planteados.

La **primera etapa** se centró en investigar la misión de la informática en la agricultura, de manera de entender el "estado del arte" y lo que esperaban los usuarios del sistema a implementar, usando como referencia otros sistemas similares ya existentes en el mercado.

La **segunda etapa** consistió en la especificación de requisitos, en donde se definió cada una de las funciones que conformarán el sistema. Para esta se realizaron entrevistas con distintos tipos de potenciales usuarios del sistema.

La **tercera etapa** se definió la arquitectura del sistema. Se seleccionaron distintas herramientas informáticas a utilizar en el desarrollo de la aplicación y de la interfaz en particular.

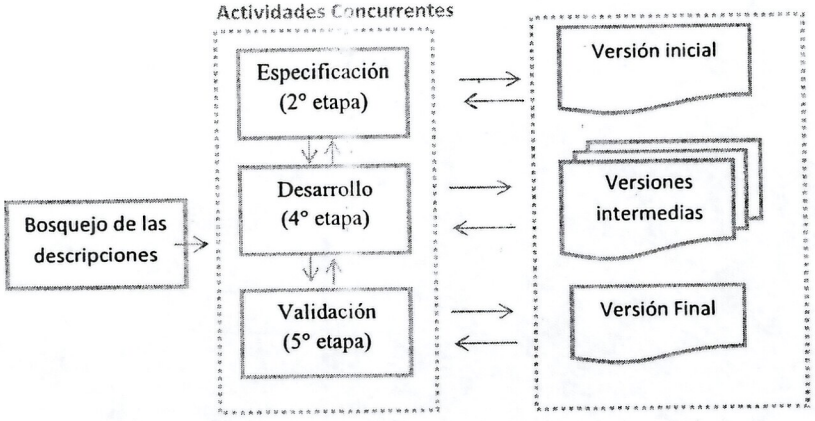
La **cuarta etapa** se dedicó al diseño de la interfaz y la implementación de lo definido en la especificación de requisitos, normalizando el desarrollo con recomendaciones internacionales (W3C, IEEE), lo que incluye la documentación asociada al sistema, como manuales de usuario.

La **quinta etapa** y final se evaluaron los resultados de la implementación y se realizaron las mejoras pertinentes.

La primera y tercera etapa fue realizada solo en la primera iteración de la ejecución del proyecto, mientras que las siguientes se realizaron concurrentemente, siguiendo un modelo de desarrollo evolutivo. Tal como se ve en el esquema 1, donde se muestra el comienzo con un bosquejo de las descripciones funcionales, para luego desarrollarlas en una versión inicial, las que en la siguiente etapa se validaron en conjunto con los usuarios. De esta interacción se logra una mayor comprensión de los requisitos, tanto para los desarrolladores, como para los usuarios finales, así de los comentarios obtenidos, se

vuelven a realizar las actividades de especificaciones funcionales, desarrollo y validación, generando versiones intermedias del sistema. Las actividades se repetirán hasta llegar una versión final aceptada por el usuario, de acuerdo a los requisitos inicialmente pactados. Este modelo de desarrollo se seleccionó como el adecuado para este sistema dado que el proyecto global está en una etapa de investigación, por lo que algunos requisitos no estaban del todo claros y podían cambiar durante el desarrollo del sistema; así se pudo *"desarrollar una implementación inicial, exponiéndola a los comentarios de usuario y refinándola a través de las diferentes versiones hasta que se desarrolla un sistema adecuado"* (Sommerville, 2005). Particularmente se usó el modelo evolutivo exploratorio, ya que se comenzaron a desarrollar las partes que se comprendían mejor. Así se logró obtener un sistema de producción flexible y expandible.

**Comentario [YG1]:** Explicación esquema 1.




Esquema 1: Modelo de desarrollo de software evolutivo.

Fuente: (Sommerville, 2005)

# Capítulo II: Estado del Arte

## *2.1. Agricultura tradicional*

Los huertos que son objeto de estudio dentro del proyecto y quienes representan a los usuarios finales del sistema gestionan sus predios agrícolas de la siguiente forma.

Los métodos de riego del arándano se diferencian dos etapas. Durante los meses de invierno, por la presencia de lluvias, no se ejecutan planes de regadío. El resto del año se realiza un plan de riego prácticamente ininterrumpido, esto a raíz de que el riego es un factor importantísimo en el adecuado desarrollo de los arándanos ya que sus raíces son superficiales y la carencia de pelos en ellas limitan la absorción de agua. De ahí nace la importancia de suministrar riego eficiente y oportuno para obtener un buen desarrollo vegetativo y crecimientos de los frutos. Una pequeña falta de este recurso producirá que el fruto se deteriore y generará una pérdida total de la producción (Clima Frutal, 2010).

En la actualidad los huertos no tienen un sistema de monitoreo que les permita conocer la humedad del suelo con un mínimo margen de error. Dados los riesgos que produce la falta de agua, los productores prefieren no correr el peligro de someter sus plantaciones a la falta del recurso, por lo que no reparan en gastos de agua ni energía eléctrica, aumentando de manera considerable los costos de producción y disminuyendo sus utilidades.

El sistema de riego utilizado es el riego por goteo, por lo que en general no están tan expuestos a suministrar un exceso de agua a los arbustos. Sin embargo, si esto llegase a ocurrir, en la actualidad los agricultores no cuentan con método alguno para detectarlo, ni menos a distancia, siendo que el exceso de agua también disminuye considerablemente la calidad de la producción.

## 2.2. *Análisis de Software Similares:*

### 2.2.1. Data Trac

Data Trac es un software existente en el mercado de la Agricultura de Precisión, desarrollado por la empresa Decagón. Se eligió este software ya que tiene funcionalidades muy similares a las que se pretende implementar.

Data trac está diseñado para recoger y realizar gráficos con los datos procedentes de dataloggers (dispositivo electrónico que registra mediciones provenientes de diferentes sensores). Permite conocer los datos del datalogger de interés, tanto en forma gráfica como en un cuadro de datos. Además, permite realizar la programación y el manejo de riegos. (Decagon, 2010)

La visualización gráfica (Ver Figura 1) permite seleccionar el período de visualización eligiendo un intervalo predeterminado. Permite aumentar o disminuir el tamaño del gráfico, ampliando o reduciendo el espacio temporal de los datos representados.

**Comentario [YG2]:** Explicación figura 1.

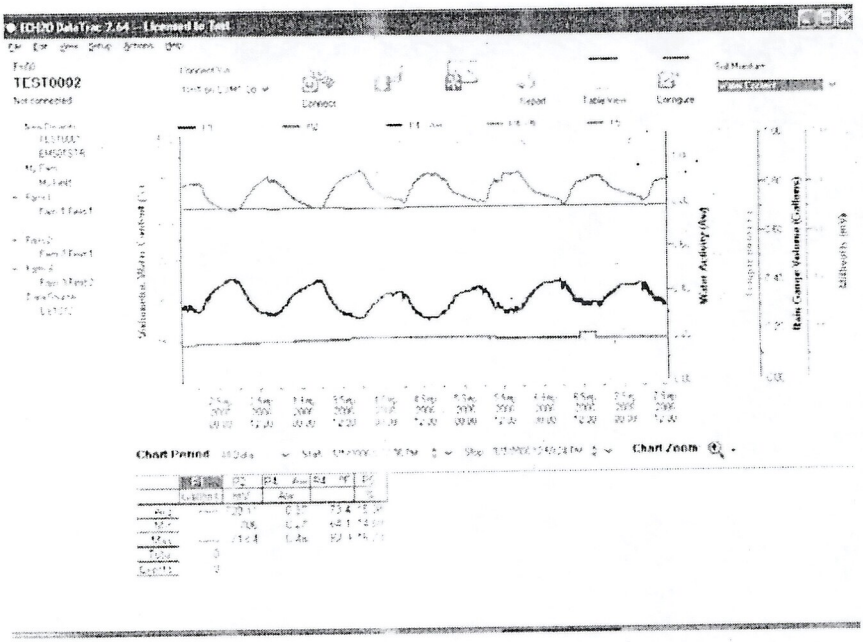



Figura 1: Vista de gráficos Data Trac.

Fuente: (Decagon, 2010).

Permite hacer lecturas instantáneas de todos los sensores un huerto, de dos formas diferenciadas por la cantidad de datos a descargar: Nuevos datos o todos los datos almacenados actualmente en el dispositivo.

La visualización de datos sin procesar se despliega en tablas (Ver Figura 2). Permite las copias de las celdas seleccionadas al portapapeles, que se puede pegar en otros programas para su uso en informes o páginas web.

**Comentario [YG3]:** Explicación figura 2

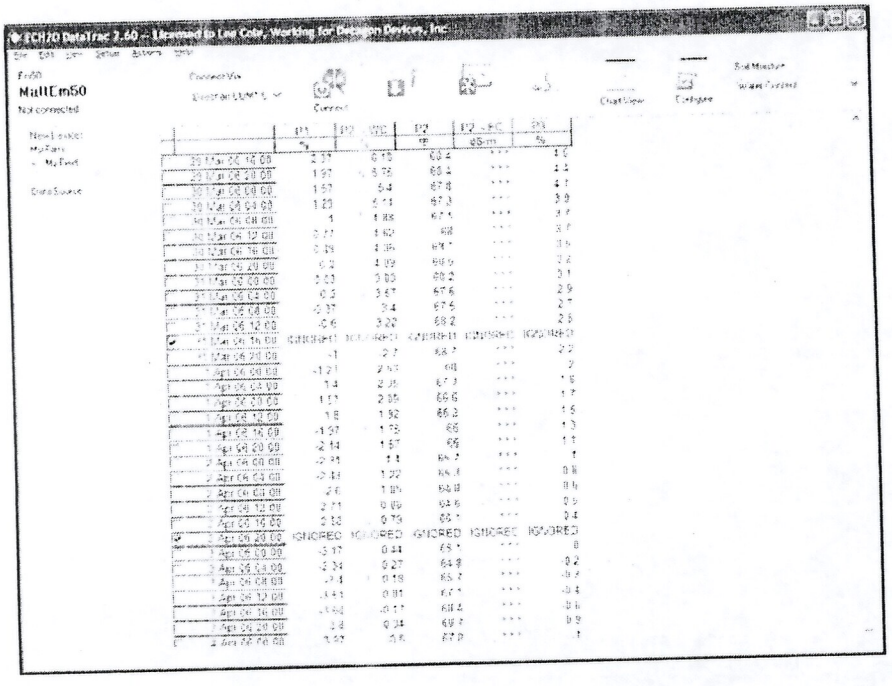


Figura 2: Vista de tablas Data Trac.

**Fuente: (Decagon, 2010).**

De Data Trac podemos observar la falencia de herramientas gráficas. Los gráficos presentados son una valiosa fuente de información, pero solo pueden ser comprendidos por personas con alto conocimiento agrónomo.

### 2.2.2. Vineyard Manager

Vineyard Manager es un sistema, que al igual que Datatrac permite el manejo de un predio agrícola, a través del uso de la Agricultura de Precisión. (Burrell, J., Brooke, T., & Beckwith, R. 2004). La diferencia radical entre este software y Datratac se observa claramente en el uso de elementos gráficos implementado en él. Por ejemplo, si el administrador desea saber cuándo y dónde el huerto fue rociado con pesticidas, se arrastra el ícono que representa los pesticidas, para ver el riesgo de un brote infeccioso con puntos rojos sobre las figuras que representan el predio (Ver Figura 3). De igual manera permite visualizar las distintas cualidades de un predio agrícola (humedad, temperatura, riesgos de intervención externa, actividades, entre otras), con posibilidad de manejar el periodo de tiempo a observar.

**Comentario [YG4]:** Explicación figura 3

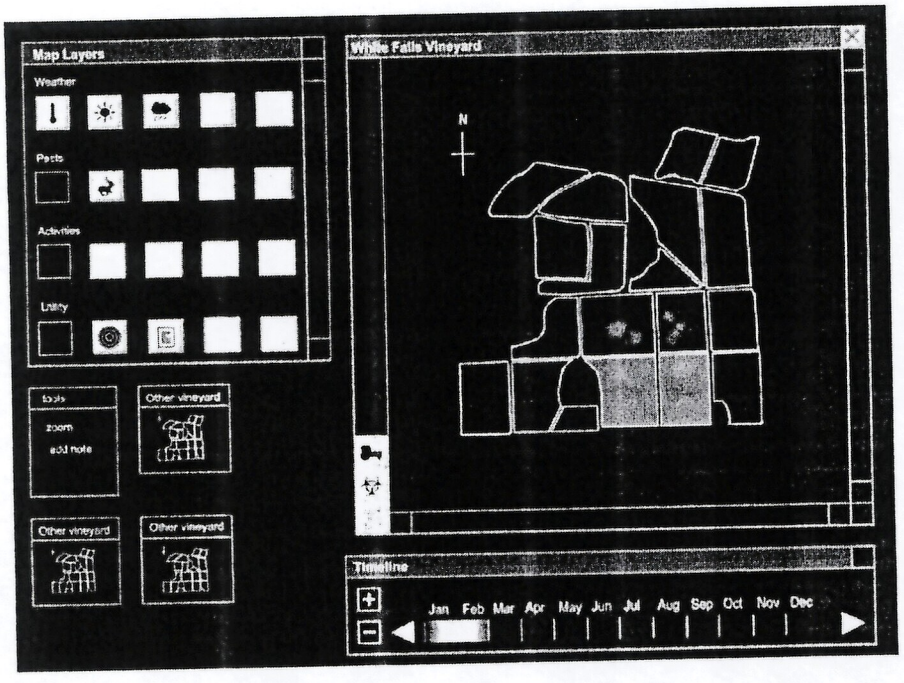



Figura 3: Vista de Vineyard Manager.

**Fuente: (Burrell, J., Brooke, T., & Beckwith, R. 2004).**

Vineyard Manager permite una interacción altamente personalizada gráficamente del huerto que se está monitoreando, sin embargo este proyecto buscaba generar un sistema genérico, independiente del predio agrícola. Por otro lado, se observa que la interfaz destina poco más de la mitad de la pantalla al área donde se despliega la información, lo que dificulta la visualización de ésta. De este ejemplo se concluyó que es mejor desplegar el menú de las funciones disponibles en la parte superior y no en la central, dejando así una mayor área al despliegue de información.

# Capítulo III: Marco Teórico

## *3.1. Agricultura de Precisión (AP)*

La AP es la aplicación de un conjunto de técnicas, apoyadas por equipamiento de alta tecnología, para el manejo de la producción agrícola. El Instituto de Investigaciones Agropecuaria define la AP como un "concepto agronómico de gestión de parcelas agrícolas, basado en la existencia de variabilidad en los campos" (Instituto de Investigaciones Agropecuarias, 2008). La variedad se refiere al supuesto de que los distintos sectores de un mismo campo responden de manera diferente a una misma técnica de cultivo (planes de regadío, fertilización, fumigación, entre otros), aplicada de manera uniforme en todo el terreno. En esos casos es posible aumentar la productividad ajustando la técnica de cultivo a cada sub-zona en particular, donde cada sub-zona puede ser identificada gracias al uso de distintos equipos tecnológicos (GPS, sensores de humedad de suelo, sensores de flujo de savia, entre otros), diferenciándose por las medidas obtenidas, como: déficit hídrico, humedad, temperatura de suelo, crecimiento vegetativo y productividad. Estas mediciones luego se pueden cruzar para establecer relaciones a índices de crecimiento y/o productividad que permiten diseñar estrategias de manejo diferenciado.

En consecuencia, la AP permite la disminución de las dimensiones de la unidad mínima de análisis, ayudando a la toma de decisiones y generando un mejor rendimiento. Por tanto, la recuperación de la inversión se logra por el ahorro por el uso informado de los insumos y por una mejor valorización de las cosechas. A su vez, el aportar la dosis correcta en el lugar idóneo y en el momento óptimo, puede beneficiar al cultivo, al suelo y a las capas de agua subterráneas (Wikipedia, 2010).

## 3.2. *Usabilidad*

Usabilidad se define como *"La medida en la que un producto se puede usar por determinados usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso especificado* "(ISO 9241-11, 1998).

Muchas veces se asume que la usabilidad es una cualidad exclusiva de la presentación de la información. Por consecuencia los test de usabilidad son aplicados solo a la interfaz y los problemas tan sólo se resuelven modificando dicha presentación. Sin embargo, esta metodología ha fallado a menudo pues en la práctica se ha observado que los grandes problemas de usabilidad se solucionan generando cambios profundos en la funcionalidad de una aplicación (Berry, 2000). De este análisis nace la *"Analogía del Iceberg de la usabilidad"*, que explica que los aspectos relacionados con la presentación, es decir, lo que normalmente se entiende como interfaz, sólo afectan en un 40% a la usabilidad, tal como se ilustra en la Figura 4. El 60% restante está influenciado por lo que en la figura se representa como "modelo del usuario", que está constituido por los objetivos que el usuario quiere alcanzar con sus tareas dentro del sistema (Berry, 2000).

**Comentario [YG5]:** Explicación figura 4

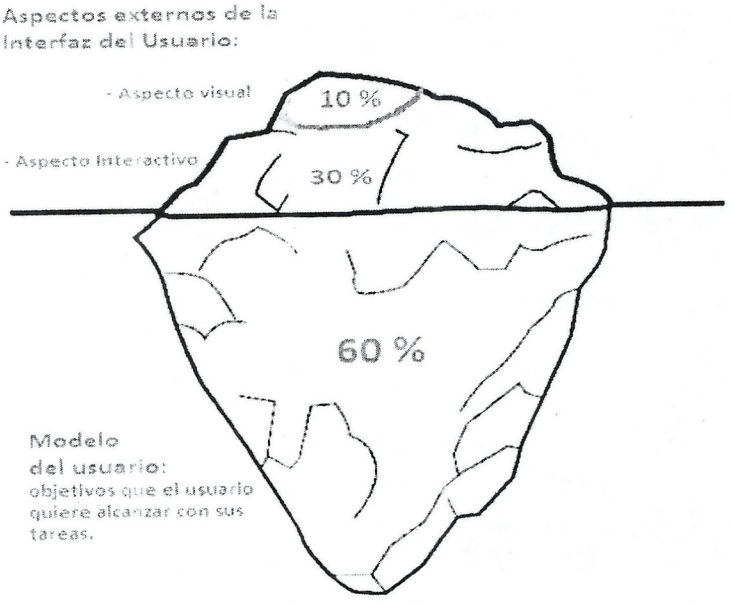



Figura 4: Iceberg de la Usabilidad

**Fuente: (Berry, 2000)**

Por lo tanto no solamente el diseño de la interfaz contribuirá a que el sistema sea atractivo al usuario, sino también la completitud y correctitud en la implementación de los requisitos que el usuario explicitó para el sistema. Para validar la usabilidad con respecto al modelo del usuario (60%) se aplicó un plan de pruebas que evalúa este ítem (capítulo VI), mientras que para asegurarnos que el 40% restante sea atractivo al usuario se realizó el siguiente análisis a considerar en el diseño de la interfaz.

### 3.2.1. Aspectos Visuales

Son distintos los factores presentes en los humanos a estudiar al momento de realizar un sistema visualmente atractivo al usuario, tales como las sensaciones de los canales de entrada (sistemas visuales, auditivos, etc.), distintos factores de percepción, distintas capacidades de memoria o necesitan distintas representaciones del conocimiento (Toni Granollers i Saltiveri, 2005). En éste caso la investigación se centrará en dos aspectos aplicables al presente proyecto:

- ✓ **La organización perceptual y la tarea del usuario**

La organización de los elementos puede facilitar o entorpecer el trabajo de un usuario. Una idea principal para un buen diseño es que la organización perceptual de la información debe estar extrapolada a cómo el usuario lleve a cabo la tarea en la actualidad, sin el uso del sistema.

- ✓ **Percepción y acceso al conocimiento**

Cuando una persona lee un texto, accede al conocimiento que tiene almacenado en la memoria semántica por medio de la trasformación fonológica de las palabras. Este proceso lleva un tiempo determinado y puede verse interferido por numerosos factores, como confundir la lectura de una letra por otra (leer mesa en vez de meta) y por supuesto si el conocimiento asociado a la información observada es deficiente, confuso o peor aún si este conocimiento asociado no tiene registros en la memoria del usuario (Toni Granollers i Saltiveri, 2005). Ante esto, una solución puede ser el "conocimiento por medio de imágenes", las que permiten entregar la información por los medios adecuados de manera que pueda ser asociado fácilmente en la memoria del usuario.

- ➢ Conocimiento por medio de imágenes: Las personas también pueden acceder a la información almacenada por representaciones anexas, como lo son las imágenes. Por ejemplo en el presente proyecto se presentó información a los

usuarios de menor nivel de conocimiento agrónomo y tecnológico, por medio de imágenes animadas, con los colores de un semáforo, que representen el estado y necesidades del predio agrícola.

### 3.2.2. Aspectos Interactivos

La usabilidad también se representa por "*la rapidez y facilidad con que las personas llevan a cabo sus tareas propias mediante el uso del producto que están trabajando* (Granollers i Saltiveri, Lorés Vidal, & Cañas Delgado) ". Para lograr una fluidez al momento de interactuar con el sistema, es necesario que estos elementos sean desarrollados bajo los siguientes aspectos (Granollers i Saltiveri, Lorés Vidal, & Cañas Delgado):

i. Aproximarse al usuario final: Se debe conocer, entender y trabajar con las personas que representan los usuarios actuales y potenciales del producto, así será posible conocer sus capacidades de comprensión.

ii. Conocer el contexto de uso: Las personas utilizan productos para incrementar su propia productividad, por lo tanto para producir sistemas usables, hay que entender y conocer los trabajos y tareas del usuario que el sistema modifica.

iii. El producto debe adaptarse a sus modelos mentales: Los usuarios normalmente tienen un esquema o una rutina de cómo realizan sus actividades diariamente, por lo tanto un nuevo sistema se debe adecuar a este proceso, de manera de no disminuir su productividad, sino por el contrario contribuir a una realización de su trabajo con un menor esfuerzo. Por lo tanto se va a relacionar usabilidad con productividad y calidad.

iv. Son los usuarios, y no los diseñadores o desarrolladores, los que determinan cuando cuándo un producto es fácil de usar: Por esto es muy importante incluir a los usuarios dentro de todas las etapas de desarrollo de software y principalmente en las etapas de prueba.

## 3.3. *Web 2.0*

La Web 2.0 es una evolución de las Web estáticas tradicionales, donde los usuarios eran solo receptores; hacia aplicaciones web dinámicas, enfocadas al usuario final y a su interacción con el entorno. Principalmente la Web 2.0 supone dos puntos (O'rreilly, 2010):

- Los usuarios pasan de tener una actividad pasiva a activa, de ser un receptor de información a participar en su construcción y elaboración, creando contenidos web gracias a la utilización de los servicios de las páginas. Por lo tanto, los sitios Web dejan de tener sentido sin usuarios que exploten los servicios que éste ofrece, donde pueden participar aportando contenidos o recursos a la aplicación web.

- La Web pasa a ser una plataforma, donde se puede acceder a múltiples herramientas para el desarrollo de tareas, sin necesidad de instalar el software en el computador, lo que marca un cambio sustancial en la definición inicial y el uso de internet.

### 3.3.1. Tecnologías basadas en Web 2.0

*"Usted puede visualizar Web 2.0 como un sistema de principios y prácticas que conforman un verdadero sistema solar de sitios que muestran algunos o todos esos principios, a una distancia variable de ese núcleo"* afirma Tim O'reilly, uno de los autores del concepto Web 2.0 (O'rreilly, 2010 ).

La Web 2.0 no busca imponer la utilización de ciertas herramientas, sino más bien crear una tendencia al uso de éstas. En la actualidad existen varias tecnologías que están usándose, en busca de seguir evolucionando la Web. Algunas tecnologías y directrices que dan vida a un proyecto Web 2.0 y que aplican al siguiente proyecto son (O'rreilly, 2010 ):

- Transformar software de escritorio hacia la plataforma Web.
- Respeto a los estándares del W3C.
- Separación del contenido del diseño con uso de hojas de estilo CSS.

- Uso de AJAX.
- Dar control total a los usuarios en el manejo de su información.
- Proveer APIS o XML para que las aplicaciones puedan ser manipuladas por otros.
- Facilitar el posicionamiento con URL´s sencillos.

## 3.4. *Definición de la Arquitectura de Software (SA)*

Aunque para la mayoría de los usuarios "la interfaz es la aplicación puesto que es la parte que ven y a través de la cual interactúan" (Hartson, 1998), debemos entender que la usabilidad de la aplicación depende no sólo del diseño del interfaz, sino también de su arquitectura, estructura, funcionalidad y organización, en otras palabras, del componente no visible del diseño, representado en el 60% del ya conocido "Iceberg de la Usabilidad", de ahí la importancia de elegir una adecuada arquitectura de desarrollo de software.

Como definición oficial de Arquitectura del Software tenemos: *"La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución"* (IEEE Std 1471-2000 , 2007).

En la actualidad existen distintos tipos de arquitectura, donde el uso o selección de una de ellas se diferenciará principalmente por tipo de sistema que se busque desarrollar, por ejemplo, entre los más usados tenemos:

- Arquitectura Orientada a Servicios: Se basa en la utilización de servicios desacoplados para dar soporte a las necesidades de los usuarios. Esta arquitectura aborda a aquellos negocios en que los servicios son requeridos en varios procesos a la vez, por lo que otorga una alta reusabilidad (Vázquez, 2010).

Esta arquitectura no es atractiva para este proyecto en particular, dado que se observan pocos servicios que puedan ser reutilizados en los distintos procesos, a lo más unas diez validaciones y no servicios de la lógica del negocio.

- Arquitectura de Cuatro Capas: En esta arquitectura se definen un conjunto de capas, donde cada capa representa: la presentación, la aplicación, el dominio de la aplicación y el repositorio (Galli Granada, 2001). Es óptima para aplicaciones complejas, que necesiten este nivel de desacoplamiento. En este caso la separación entre el dominio de la aplicación (lógica del negocio) y la aplicación en sí es bastante compleja y el equipo no cuenta con los conocimientos para ejecutarlo, por lo que el estudio de una herramienta sería bastante extenso, motivo por el cual esta arquitectura también es descartada.

- Modelo Vista Controlador: Esta arquitectura separa la lógica de negocio (el modelo), la presentación (la vista) y las peticiones (controlador), por lo que es útil para aplicaciones de rápida evolución, pues la separación facilita la mantención (Gómez García, 2008).

### 3.4.1. Modelo Vista Controlador (MVC)

Luego de investigar algunas arquitecturas se seleccionó MVC, pues el proyecto no tenía requisitos muy claros, por lo que preveía cambios en ciertos elementos, cambios que al usar MVC solo afectarán al componente a modificar y no al resto (Gómez García, 2008). Además se seleccionó dado que potencia aquellas aplicaciones que requieran de una gran interactividad con los usuarios, como es el caso de aplicaciones Web. Por último, y quizás el motivo más importante que llevó a seleccionar esta arquitectura, es porque el equipo de trabajo ya contaba con conocimiento en el desarrollo de aplicaciones MVC a través del framework Struts2, lo que permitía disminuir la alta curva de aprendizaje y consecuentemente se dedicara menos tiempo a dicho aprendizaje y más al desarrollo.

Las desventajas que se aborda con esta arquitectura es que tiene una alta curva de aprendizaje para los nuevos desarrolladores; sin embargo como existe conocimiento previo, esta desventaja será fácil de superar. Por otro lado la distribución de componentes obliga a crear y mantener un mayor número de archivos, por lo cual el desarrollo tuvo que ser muy cuidadoso en este punto y se mantuvo una organización jerarquizada de carpetas con los distintos archivos que correspondían a un mismo componente, lo que facilitará la comprensión del sistema en la etapa de mantención. (Sule, 2008)

Esta arquitectura organiza la aplicación en tres partes bien diferenciadas. Por un lado tenemos el Modelo, el cual representa los datos de la aplicación y sus reglas de negocio, por otro la Vista, compuesta de interfaces que representan los formularios de entrada y salida de datos, y finalmente, el Controlador, encargado de procesar las peticiones entrantes del usuario para generar una respuesta y controlar el flujo de ejecución del sistema, tal como se muestra en la Figura 5.

**Comentario [YG6]:** Explicación figura 5.

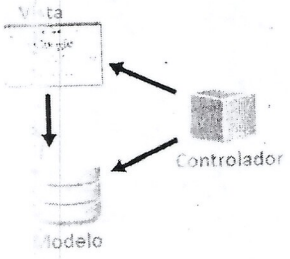



Figura 5: Arquitectura MVC

Fuente: **(Gómez García, 2008)**

A continuación se detalla la descripción de los elementos de la arquitectura MVC.

i. **Modelo:** Es el encargado de guardar los datos en un medio persistente (ya sea una base de datos, un archivo de texto, registro, etc.). En el modelo es donde se genera la lógica del negocio y a su vez es el proveedor de los recursos. Es muy típico que los modelos incorporen otro patrón de diseño, ya que así es más fácil mantener la comunicación con la base de datos.

ii. **Vistas:** Estas presentan el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario. Es la encargada de mostrar las respuestas que deben ser enviadas al cliente. Esta respuesta normalmente incluirá datos generados por el controlador. De esta forma el contenido de la página no será estático sino que será generado de forma dinámica.

iii. **Controlador:** Es el que escucha los cambios en las vistas y se los envía al modelo, el cual le regresa los datos a la vista. Este ciclo se repite cada vez que el usuario genere una acción. El Controlador en cierta forma debe tener un registro de la relación entre ordenes que le pueden llegar y la lógica de negocio que le corresponde (es como una operadora de teléfono que recibe una petición y une dos líneas).

# Capítulo IV: Especificación de Requisitos del Sistema

## *4.1. Introducción*

La siguiente especificación de requisitos define las funcionalidades para el sistema de monitoreo y planificación de riego, sistema que será una herramienta para que los agricultores puedan acceder remotamente a observar y gestionar sus huertos, integrando distintos niveles de educación en materia agrónoma, como también distintos niveles de conocimientos del uso de las tecnologías de la información de los usuarios finales.

### 4.1.1. Propósito

El propósito de generar una ERS es describir detalladamente los requerimientos funcionales y no funcionales del sistema de monitoreo y planificación de riego. Para esto se conceptualizará lo que los agricultores y dueños de fundo esperan del sistema, para que estas ideas sean comprendidas sin ambigüedades por los desarrolladores. También será útil en la etapa de verificación y pruebas finales, pues será una guía que contendrá los resultados que se esperan del sistema.

### 4.1.2. Alcance

El sistema a desarrollar pretende cubrir las necesidades de observar y modificar el plan de riego y fertilización, administrar los elementos del huerto y monitorear la humedad del huerto, permitiendo acceder a esta información de forma remota, clasificándola y organizándola según el nivel de relevancia para el tipo de usuario que esta interactuando con el sistema. Este sistema se conectará con una red de sensores inalámbricos, que serán manipulados en un aplicativo en C. Sin embargo,

la configuración de esta red no será explicitada en esta ERS, pues es tarea de otro equipo del proyecto global.

### 4.1.3. Personal involucrado

| | |
|---|---|
| **Nombre** | Joel Jil Garay |
| **Rol** | Desarrollador |
| **Responsabilidades** | Encargado de desarrollar el **modelo y controlador** del sistema, lo que implica modelar la base de datos y la lógica del negocio. |

| | |
|---|---|
| **Nombre** | Yasna Gutiérrez Neira |
| **Rol** | Desarrollador |
| **Responsabilidades** | Encargado de desarrollar y diseñar la **interfaz** del sistema (vista), para cada uno de los usuarios que interactuarán en él. |

## 4.2. *Descripción general*

### 4.2.1. Perspectiva del producto

Se espera obtener un sistema web de monitoreo y planificación de riego, mediante el cual los agricultores puedan gestionar sus huertos. Este sistema se complementa con una red de sensores, encargados de capturar las medidas del huerto. Sin embargo, la configuración de éstos es tarea de otro equipo, por lo que no se especificarán en este documento.

### 4.2.2. Funcionalidad del producto

Los requisitos funcionales se organizarán por módulos, de la siguiente manera.

Módulo Monitoreo (M)

M1. Monitorear por nodo

M2. Mostrar el estado de riego actual

M3. Mostrar los eventos de riego

M4. Mostrar la cantidad de agua usada en un periodo de tiempo

M5. Mostrar la cantidad de energía eléctrica usada en un periodo de tiempo

M6. Mostrar el estado de las baterías de los nodos

M7. Mostrar alertas de humedad

Módulo Fertilización (F)

F1. Ingresar plan de fertilización

F2. Editar plan de fertilización

F3. Eliminar plan de fertilización

F4. Mostrar fertilizaciones activas

F5. Mostrar fertilizaciones pasadas

Módulo Riego (R)

R1. Crear Método de riego

R2. Mostrar históricos de Métodos de Riego

R3. Mostrar Método de Riego Actual

Módulo Administración de huerto (H)

H1 Administrar Bombas

H1.1 Ingresar bomba

H1.2 Eliminar bomba

H1.3 Editar bomba

H2 Administrar Válvulas

H2.1 Ingresar válvula

H2.2 Eliminar válvula

H2.3 Editar válvula

H3 Administrar Nodos

H3.1 Ingresar nodo

H3.2 Eliminar nodo

H3.3 Editar nodo

H4 Administrar Canales

H4.1 Ingresar Canal

H4.2 Editar canal

H5. Administrar usuarios

H5.1 Ingresar usuarios

H5.2 Modificar usuarios

H5.3 Eliminar usuarios

H5.4 Listar usuarios del huerto

H6. Mostrar configuración actual del huerto

Administración de Cuentas (C)

C.1 Autentificar cuenta de usuario

C.2 Editar datos de usuario

### 4.2.3. Características de los usuarios

| Tipo de usuario | **Usuario monitor** |
|---|---|
| Formación | Tiene bajo conocimiento en el uso de las tecnologías de la información y bajo conocimiento técnico de la agricultura. |
| Habilidades | Es quien más interactúa con el huerto, pues es él quien normalmente (de forma manual, sin el sistema implementado) monitorea visualmente el huerto. Sin embargo, dado su falencia de conocimientos técnicos es necesario que al momento de presentar los datos éstos sean fáciles de interpretar, es decir, mostrarle si las lecturas de los sensores implican algún riesgo o no para el huerto (de manera visual) junto con una acción a seguir. |

| | |
|---|---|
| Actividades | En el sistema él podrá observar todo lo referente al módulo monitoreo del huerto, como también a visualizar los planes de fertilización y la configuración actual del huerto. Dado que estas funcionalidades no generan modificaciones en el huerto, ni en la configuración del sistema, se le llama usuario "Monitor". (Para mayor detalle ver diagramas de casos de uso. Anexos 1, 2, 3 y 4). |

**Comentario [YG7]:** Me había equivocado, era diagrama de casos de uso

| Tipo de usuario | **Usuario controlador** |
|---|---|
| Formación | Tiene manejo cotidiano del uso de las tecnologías de la información y alto conocimiento técnico de la agricultura. |
| Habilidades | Dado su alto conocimiento agrícola, es capaz de interpretar datos complejos obtenidos del huerto, por lo tanto no necesita que el sistema le interprete las lecturas de los sensores, pues esta información ya es valiosa para él. |
| Actividades | En el sistema, él podrá observar todo lo referente al módulo monitoreo del huerto, como también al módulo fertilización, ejecutar planes de regadío y ver la configuración actual del huerto (para mayor detalle, ver diagramas de casos de uso. Anexos 1, 2, 3 y 4). Dado su rol es llamado usuario "Controlador". |

**Comentario [YG8]:** Me había equivocado, era diagrama de casos de uso

| Tipo de usuario | **Usuario administrador de huerto** |
|---|---|
| Formación | La formación de este usuario es bastante diversa, puede o no tener conocimientos agrícolas y sus conocimientos en el uso de las tecnologías es suficiente para el uso de un sistema web. |
| Habilidades | Su habilidad principal es la capacidad de tomar decisiones radicales en la configuración de su huerto y de sus usuarios asociados. |
| Actividades | En el sistema, él podrá observar todo lo referente al módulo monitoreo del huerto, como también al módulo fertilización y ejecutar planes de regadío.<br>Además, tendrá acceso a modificar la configuración de su huerto y a agregar, editar y eliminar usuarios de su huerto (para mayor detalle, ver diagramas de casos de uso. Anexos 1, 2, 3 y 4). |

**Comentario [YG9]:** Me había equivocado, era diagrama de casos de uso

Para visualizar mejor que funciones estarán disponibles para un usuario en particular, ver en anexos 1, 2, 3 y 4 los diagramas de casos de usos.

### 4.2.4. Restricciones

Dado que el sistema es parte de un proyecto que aún está en una etapa de investigación, es necesario que éste sea flexible a los cambios y escalable futuras expansiones. Por lo cual se usará en su desarrollo la arquitectura de software MVC (ver justificación en ítem 3.4.1).

El sistema se desarrollará bajo la plataforma Java EE usando el framework "Struts2" como marco principal de desarrollo (ver justificación en ítem 5.2.2).

En la capa "Vista" (interfaz de usuario) se utiliza el framework "Tiles", encargado de presentar la información (ver justificación en ítem 5.2.3.2). Como servidor web se usa Apache Tomcat y como servidor de base de datos MySQL server. Lo anteriormente descrito se resume en la Figura 6, junto a la interacción de las distintas herramientas.

**Comentario [YG10]:** Justificación figura 6.

Para el acceso a los datos en la capa de persistencia se utiliza "Hibernate" como framework encargado de realizar la comunicación entre la base de datos y el sistema. Esta base de datos se sincronizará con una base de datos central, que junto a un aplicativo desarrollado en C, serán los encargados de recibir los datos de los sensores inalámbricos. Este modelo es el que en un futuro se pretende replicar para n huertos, tal como muestra la Figura 6. Sin embargo esta especificación de requisitos no se interiorizará en explicar las funciones del aplicativo en C, ni en la conectividad con los sensores, tal como se mencionó en el punto 4.1.2.

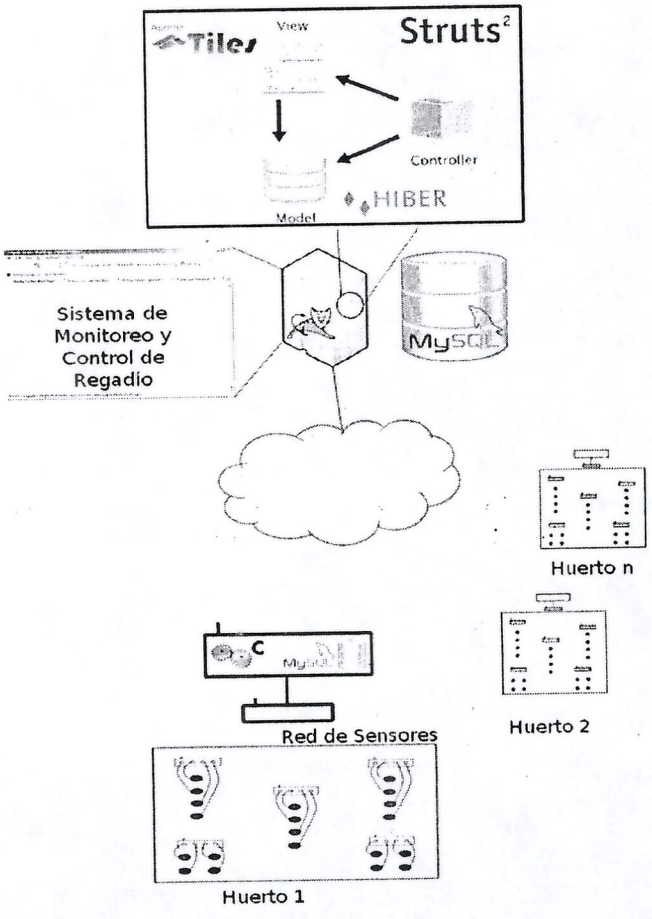



Figura 6: Componentes del Sistema de Monitoreo y planificación de regadío.

Fuente: Propia

Todos los componentes del sistemas (Frameworks, Librerías, etc.), entornos de desarrollos usados, servidor web y sistema operativo usados en el desarrollo son libres, lo cual implica que no existirá conflictos de licencias. Los desarrolladores están en libertad de agregar otras bibliotecas, siempre y cuando sus licencias también sean libres y permitan uso comercial.

### 4.2.5. Suposiciones y dependencias

Este sistema está diseñado para ser implementado en un huerto que use el goteo como método de regadío. Además la base de datos está diseñada para que una bomba pueda tener asociada varias válvulas (reguladora del paso de agua de riego), pero una válvula debe pertenecer sólo a una bomba. A su vez una válvula tiene asociado un nodo de canales (conjunto de sensores) y un nodo de canales está asociado a una sola válvula. Este nodo de sensores está compuesto por varios canales, que serán quienes capturarán los valores de humedad del lugar donde están posicionados. De este nodo de canales se seleccionará un canal de interés, que es un canal que entrega la lectura más fidedigna del nodo. Si este modelo de regadío cambia, puede afectar a los requisitos y se debe rediseñar el sistema.
Por otro lado, se supondrá dentro de todo el documento que un canal es lo mismo que un sensor, como también que un nodo corresponde a un conjunto de canales.

### 4.2.6. Evolución previsible del sistema

En esta versión del Sistema de Monitoreo y Planificación de riego no incorporó un módulo de "ubicador de sensores". El ubicador de sensores es un algoritmo para determinar una posición geográfica óptima para los sensores de humedad, así obtener mayor fidelidad de los datos obtenidos del huerto.

El diseño del sistema es para un huerto, lo que en una siguiente versión de multiplicará para N huertos. Esto tiene mayor impacto en el diseño y modelo relacional de la base de datos del sistema.

## *4.3. Requisitos específicos*

### 4.3.1. Requisitos funcionales

#### 4.3.1.1. Módulo Monitoreo (M)

**M1. Monitorear por nodo**

**Descripción:** Esta función desplegará los datos capturados por los sensores asociados a un nodo, en un periodo de tiempo determinado.

**Entradas:** *Id_nodo* (entero, hasta 10 caracteres, obligatorio), *fecha_inicial* (formato dd/mm/aa hh:mm, obligatorio) y *fecha_final* (formato dd/mm/aa hh:mm, obligatorio).

**Proceso:** Una vez que el usuario haya seleccionado un nodo, se listarán los sensores asociados a él, dando la posibilidad de seleccionar los sensores a monitorear (uno o más). Finalmente se seleccionará el periodo a monitorear, indicando fecha y hora del periodo inicial y final, para desplegar la información recopilada; humedad capturada por los sensores seleccionados, versus fecha y hora de la lectura.

Se debe validar que todos los datos de entrada obligatorios hayan sido ingresados y en el formato especificado, si alguno falta o fue ingresado en formato erróneo, indicar cuál es y permitir volver a intentarlo.

**Salidas:** Las salidas dependerán del resultado del proceso.

**Comentario [YG11]:** Este proceso de validación lo había ingresado en las salidas, ahora lo corregí y lo integré como parte del proceso. Aplica para todo el documento de ERS.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de no haber ingresado los datos obligatorios; mostrar mensaje por pantalla "Datos requeridos".

**S3.** En caso de haber ingresado las entradas en un formato que no corresponda al especificado; mostrar mensaje "Datos incorrectos".

**S4:** En caso de éxito, por no haber ocurrido S1, S2 ni S3, desplegar *datos* (entero, hasta 5 caracteres), *humedad* (entero, hasta 5 caracteres), *fecha_lectura* (formato fecha dd/mm/aa) y *hora_lectura* (formato hora hh:mm).

**M2. Mostrar el estado de riego actual**

**Descripción:** Esta función permitirá ver el estado de humedad en el instante y compararlo con el umbral de riego definido.

**Entradas:** *Id_canal_interes* (entero, hasta 10 caracteres).

**Proceso:** Se realizará una lectura de todos los canales de interés de las válvulas del huerto, desplegando ese último dato en un gráfico de barras, donde cada barra representara a cada una de las válvulas del huerto y sobre cada barra de humedad se graficará una barra que represente el umbral de humedad de riego. Para las válvulas que no registren un umbral de regadío, por no tener un método de riego reactivo, solo se desplegará la última lectura del sensor. Los datos se deben organizar en un gráfico de líneas cuyo eje *"x"* contenga *nombre_valvula*, el eje *"y"* representará la *humedad*, por lo que se asociará con los *datos* de los sensores y *umbral*. El umbral se mostrará sólo para las válvulas con método de riego reactivo.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de éxito (que no ocurra S1) desplegar *nombre_valvula* (alfanumérico, hasta 30 caracteres), *humedad* (flotante, 2 decimales, hasta 5 caracteres), *datos* (entero, hasta 5 caracteres) y *umbral* (entero, hasta 5 caracteres).

**M3. Mostrar los eventos de riego**

**Descripción:** Esta función mostrará los eventos de regadíos de todo el huerto, para el último método de regadío implementado, es decir los momentos en que se lanzó el regadío. Estos se organizarán en una tabla.

**Entradas:** *Fecha_implementación_regadío* (formato dd/mm/aa).

**Proceso:** Se realizará una búsqueda del último método de regadío ejecutado, comparando las fechas en que éstos se implementaron. Luego se listará toda la información asociada al evento riego organizada en una tabla.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de no encontrar métodos de regadío guardados; mostrar mensaje por pantalla "No se han guardado métodos de regadío".

**S3:** En caso de haber encontrado el último método de regadío implementado, desplegar: *Id_evento* (entero, hasta 10 caracteres), *Id_riego* (entero, hasta 10 caracteres), *Id_válvula* (entero, hasta 10 caracteres), *fecha_lanzamiento* (formato fecha dd/mm/aa) y *duración* (formato hora hh:mm).

**M4. Mostrar la cantidad de agua usada en un periodo de tiempo**

**Descripción:** Mostrará la cantidad de agua utilizada en un periodo determinado, resultado de los regadío ejecutados.

**Entradas:** *Fecha_inicial* (formato fecha dd/mm/aa, obligatorio), *fecha_final* (formato fecha dd/mm/aa , obligatorio).

**Proceso:** Una vez ingresado correctamente el periodo de monitoreo se realizará una búsqueda de todos los eventos de regadío ejecutados dentro de ese periodo. Luego para todas las válvulas asociadas a una misma bomba se sumarán los tiempos de ejecución ("*duración*" formato hora) de regadío, que no hayan sido ejecutados de forma paralela, sino en distinta fecha y hora. Así se obtendrá el tiempo de ejecución total de la bomba. El resultado de esta sumatoria se multiplicará por el caudal de la bomba ("*caudal*" entero hasta 5 caracteres). Este proceso se repetirá para todas las bombas del huerto, resultados que se sumarán entre sí y generarán la cantidad de agua utilizada en el huerto.

La fórmula descrita anteriormente se resume de la siguiente forma:

$$Agua\_total = \sum_{i=0}^{n} (caudal_{bomba\_i} * \sum_{j=0}^{m} (duracion_{valvula\ j\ bomba\_i}))$$

Donde:

n=número total de bombas del huerto

m= número total de válvulas del huerto

Se debe validar que todos los datos de entrada obligatorios hayan sido ingresados y en el formato especificado, si alguno falta o fue ingresado en formato erróneo, indicar cuál es y permitir volver a intentarlo.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2.** En caso de no haber ingresado los datos de entrada obligatorios; mostrar mensaje por pantalla "Ingrese datos obligatorios".

**S3.** En caso de no haber ingresado los datos en el formato especificado; mostrar mensaje por pantalla "Datos incorrectos".

**S4:** En caso éxito (no ocurrencia de S1, S2 ni S3); desplegar el resultado *Agua_total* (Flotante, hasta 20 caracteres y 1 decimal).

**M5. Mostrar la cantidad de energía eléctrica usada en un periodo de tiempo**

**Descripción:** Mostrará la cantidad de energía eléctrica utilizada en un periodo determinado, resultado de los regadío ejecutados.

**Entradas:** *Fecha_inicial* (formato fecha dd/mm/aa, obligatorio), *fecha_final* (formato fecha dd/mm/aa, obligatorio).

**Proceso:** Una vez ingresado correctamente el periodo de monitoreo se realizará una búsqueda de todos los eventos de regadío ejecutados dentro de ese periodo. Luego, para todas las válvulas asociadas a una misma bomba, se sumarán los tiempos de ejecución de regadío ("*duración*" formato hora), que no hayan sido ejecutados de forma paralela, sino en distinta fecha y hora. Así se obtendrá el tiempo de ejecución total de la bomba. El resultado de esta sumatoria se multiplicará por el consumo de energía eléctrica de la bomba ("*consumo_electrico*" flotante, hasta 5 caracteres y 1 decimal). Este proceso se repetirá para todas las bombas del huerto, resultados que se sumarán entre sí y generarán la cantidad de electricidad utilizada en el huerto.

La fórmula descrita anteriormente se resume de la siguiente forma:

$$energia\_total = \sum_{i=0}^{n}(consumo\_electrico_{bomba\_i} * \sum_{j=0}^{m}(duracion_{valvula\_j\ bomba\_i}))$$

Donde:

n=número total de bombas del huerto.

m= número total de válvulas del huerto.

Se debe validar que todos los datos de entrada obligatorios hayan sido ingresados y en el formato especificado, si alguno falta o fue ingresado en formato erróneo, indicar cuál es y permitir volver a intentarlo.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2.** En caso de no haber ingresado los datos de entrada obligatorios; mostrar mensaje por pantalla "Ingrese datos obligatorios".

**S3.** En caso de no haber ingresado los datos en el formato especificado; mostrar mensaje por pantalla "Datos incorrectos".

**S4:** En caso éxito (no ocurrencia de S1, S2 ni S3); desplegar el resultado *energia_total* (flotante, hasta 20 caracteres y 1 decimal).

**M6. Mostrar el estado de las baterías de los nodos**

**Descripción:** Esta función permitirá ver el estado de las baterías que alimenta a los nodos.

**Entradas:** *nivel_medio_bateria* (entero, hasta 5 caracteres), *nivel_bajo_bateria* (entero, hasta 5 caracteres).

**Proceso:** Se realizará una lectura de los canales que hayan sido guardados como baterías de cada nodo y se comparará su lectura con las entradas de este proceso. Si la lectura es menor que *nivel_medio_bateria*, se dará alerta informando que pronto necesitará cambiar la baterías, mientras si es menor que *nivel_bajo_bateria*, se dará alerta informando que la batería se ha acabado.

**Comentario [YG12]:** Esto estaba en las salidas y era parte de los procesos, por lo que fue corregido.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de éxito (que no ocurra S1) y si el proceso arroja que la lectura es menor que *nivel_medio_bateria*; mostrar por pantalla "compre otra batería, pronto tendrá que cambiarla".

**S3:** En caso de éxito (que no ocurra S1) y si el proceso arroja que la lectura es menor que *nivel_bajo_bateria*, mostrar por pantalla "cambie la batería".

**M7. Mostrar alertas de humedad**

**Descripción:** Esta función permitirá ver si el huerto tiene déficit o exceso de humedad en alguna de sus válvulas y a su vez una acción a seguir frente a un determinado suceso.

**Entradas:** *Minima_humedad* (entero, hasta 5 caracteres), *maxima_humedad* (entero, hasta 5 caracteres).

**Proceso:** Se realizará una lectura de los canales de interés de cada válvula, y los comparará con las entradas ya descritas. Si la lectura del canal es menor que la entrada *minima_humedad*; mostrar alerta de humedad baja y como acción a seguir mostrar "Aumente la frecuencia de riego o disminuya el umbral de regadío".

**Comentario [YG13]:** Esto estaba en las salidas y era parte de los procesos, por lo que fue corregido.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de éxito (que no ocurra S1) y si el proceso arroja que la lectura del canal es menor que la entrada *minima_humedad*; mostrar alerta de humedad baja y como acción a seguir mostrar "Aumente la frecuencia de riego o disminuya el umbral de regadío".

**S3:** En caso de éxito (que no ocurra S1) y si el proceso arroja que la lectura del canal es mayor que la entrada *maxima_humedad*; mostrar alerta de humedad alta y como acción a seguir mostrar "Disminuya la frecuencia o el tiempo de regadío".

#### 4.3.1.2. Módulo Fertilización (F)

**F1. Ingresar plan de fertilización**

**Descripción:** Permitirá registrar un plan de fertilización para el huerto.

**Entradas:** *Fecha_implementacion* (formato fecha dd/mm/aa, obligatorio), *hora_implementacion* (formato hora hh:mm, obligatorio), *nombre_fertilizacion* (alfanumérico, hasta 30 caracteres, obligatorio), *id_fertilizacion* (entero, hasta 10 caracteres, obligatorio) y *plan_fertilizacion* (alfanumérico, hasta 1000 caracteres, obligatorio).

**Proceso:** Una vez ingresados los datos de entrada, se validarán según su formato definido y si son correctos, se ingresarán a la base de datos.

Se debe validar que todos los datos de entrada obligatorios hayan sido ingresados y en el formato especificado, si alguno falta o fue ingresado en formato erróneo, indicar cuál es y permitir volver a intentarlo.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de que falten los datos obligatorios; mostrar mensaje por pantalla "Datos requeridos".

**S3.** En caso de haber ingresado datos en un formato distinto al especificado; mostrar mensaje por pantalla "Datos incorrectos".

**S4.** En caso de éxito, es decir que no ocurra S1, S2 ni S3; mostrar mensaje por pantalla "Plan de fertilización guardada".

**F2. Editar plan de fertilización**

**Descripción:** Permitirá editar la información asociada a un plan de fertilización registrado.

**Entradas:** *Nueva_fecha_implementacion* (formato fecha dd/mm/aa, obligatorio), *nueva_hora_implementacion* (formato hora hh:mm, obligatorio), *nuevo_nombre_fertilizacion* (alfanumérico, hasta 30 caracteres, obligatorio), *id_fertilizacion* (entero, hasta 10 caracteres, obligatorio) y *nuevo_plan_fertilizacion* (alfanumérico, hasta 1000 caracteres, obligatorio).

**Proceso:** Una vez que el usuario haya seleccionado un plan de fertilización a editar, se desplegará su información actual, con posibilidad de editarla. Luego se validarán los datos de entrada y se guardarán en la base de datos.

Se debe validar que todos los datos de entrada obligatorios hayan sido ingresados y en el formato especificado, si alguno falta o fue ingresado en formato erróneo, indicar cuál es y permitir volver a intentarlo.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos, mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de que falten los datos obligatorios; mostrar mensaje por pantalla "Datos requeridos".

**S3.** En caso de haber ingresado datos en un formato distinto al especificado; mostrar mensaje por pantalla "Datos incorrectos".

**S4.** En caso de éxito, es decir que no ocurra S1, S2 ni S3; mostrar mensaje por pantalla "Plan de fertilización editado correctamente".

**F3. Eliminar plan de fertilización**

**Descripción:** Permitirá eliminar una plan de fertilización ya guardado para el huerto. Sólo se dará la posibilidad de eliminación a aquellos planes cuya fecha de implementación sea mayor a la actual.

**Entradas:** *Id_fertilizacion* (entero, hasta 10 caracteres).

**Proceso:** Una vez que el usuario seleccione un determinado plan de fertilización a eliminar, se eliminaran sus registros de la base de datos.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2.** En caso de éxito, es decir que no ocurra S1; mostrar mensaje por pantalla "Plan de fertilización eliminado".

**F4. Mostrar fertilizaciones activas**

**Descripción:** Permitirá mostrar por pantalla una plan de fertilización activo guardado para el huerto. Una fertilización activa será aquella cuya fecha y hora de implementación sea mayor a la actual.

**Entradas:** *fecha_actual* (formato fecha dd/mm/aa), *hora_actual* (formato hora hh:mm).

**Proceso:** Se realizará una búsqueda de todos los planes de fertilización cuya fecha y hora de implementación sea mayor a la actual. Este resultado se desplegará por pantalla en orden ascendente, con respecto a su fecha y hora de implementación.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2.** En caso de éxito, es decir que no ocurra S1; mostrar por pantalla *fecha_implementacion* (formato fecha dd/mm/aa), *hora_implementacion* (formato hora hh:mm), *nombre_fertilizacion* (alfanumérico, hasta 30 caracteres) y *plan_fertilizacion* (alfanumérico, hasta 1000 caracteres).

**F5. Mostrar fertilizaciones pasadas**

**Descripción:** Permitirá mostrar por pantalla una plan de fertilización ya realizadas en el huerto. Una fertilización pasada será aquella cuya fecha y hora de implementación sea menor a la actual.

**Entradas:** *fecha_actual* (formato fecha dd/mm/aa), *hora_actual* (formato hora hh:mm).

**Proceso:** Se realizará una búsqueda de todos los planes de fertilización cuya fecha y hora de implementación sea menor a la actual. Este resultado se desplegará por pantalla en orden descendente, con respecto a su fecha y hora de implementación.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2.** En caso de éxito, es decir que no ocurra S1; mostrar por pantalla *fecha_implementacion* (formato fecha dd/mm/aa), *hora_implementacion* (formato hora hh:mm), *nombre_fertilizacion* (alfanumérico, hasta 30 caracteres) y *plan_fertilizacion* (alfanumérico, hasta 1000 caracteres).

### 4.3.1.3. Módulo Riego (R)

**R1. Crear método de riego**

**Descripción:** Permitirá crear un nuevo método de riego, para ser implementado cierta fecha en el huerto. Este método puede ser del tipo **Plan**, que tendrá una frecuencia, un tiempo de regadío y una fecha de inicio, o bien **Reactivo**, que tendrá un tiempo de regadío y un umbral de humedad mínimo de ante el cual se activará. A su vez puede ser de modo **Manual**, es decir que el usuario ingresará los parámetros del reacción del método de

riego (*tiempo*, *frecuencia* y *fecha_inicio*; o *tiempo* y *umbral*), o bien automático, donde el sistema los generará[1].

Observación: Un método de riego se implementará para todo el huerto, sin embargo los valores: *tiempo*, *frecuencia* y *fecha_inicio*; o *tiempo* y *umbral*, (según corresponda al tipo de regadío) serán ingresados separadamente para cada válvula del huerto y a su vez se podrá excluir una válvula, en la que no se ingresarán dichos valores y ésta no ejecutará los eventos de regadío.

**Entradas:** *Id_metodoriego* (entero, hasta 10 caracteres, obligatorio), *nombre_riego* (alfanumérico, hasta 30 caracteres, obligatorio), *fecha_implementacion* (formato fecha dd/mm/aa, obligatorio), *observaciones* (alfanumérico, hasta 500 caracteres, no obligatorio) e *id_usuario* (entero, hasta 10 caracteres, obligatorio).

Para cada válvula en la que se implementará el nuevo método de riego se ingresará *tipo_riego* (Plan o reactivo) y *modo_riego* (manual o automático). Si es de modo manual se ingresarán los siguientes datos:

- Si es riego tipo plan: *tiempo* (formato hora hh:mm, obligatorio), *frecuencia* (entero, hasta 5 caracteres, obligatorio), *fecha_inicio* (formato fecha dd/mm/aa, obligatorio) e *id_valvula* (entero, hasta 10 caracteres, obligatorio).
- Si es riego tipo reactivo: *tiempo* (formato hora hh:mm, obligatorio), *umbral* (entero, hasta 5 caracteres, obligatorio) e *id_válvula* (entero, hasta 10 caracteres, obligatorio).

**Proceso:** Una vez ingresado de entrada, se validarán y si son correctos, se ingresarán a la base de datos.

---

[1] En una posterior versión de este sistema se implementará un software de aprendizaje, que a partir de los regadíos ya implementados generará los parámetros de regadío que hicieron reaccionar mejor al huerto, por ahora y por estar aun en etapa de investigación, el sistema ingresará los siguientes valores: umbral(700), tiempo (00:40:00) para un plan reactivo; frecuencia(02:00:00), tiempo (00:40:00) y fecha_inicio (fecha actual) para un regadío tipo plan.

Se debe validar que todos los datos de entrada obligatorios hayan sido ingresados y en el formato especificado, si alguno falta o fue ingresado en formato erróneo, indicar cuál es y permitir volver a intentarlo.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de que falten los datos obligatorios especificados en la entrada; mostrar mensaje por pantalla "Datos requeridos".

**S3.** En caso de haber ingresado datos en un formato distinto al especificado; mostrar mensaje por pantalla "Datos incorrectos".

**S4.** En caso de éxito, es decir que no ocurra S1, S2 ni S3; mostrar mensaje por pantalla "Plan de regadío ingresado correctamente".

**R2. Mostrar históricos método de riego**

**Descripción:** Permitirá observar todos los métodos de riegos que han sido ejecutados en el huerto, ordenados en una tabla.

**Entradas:** *Id_metodo_riego* (entero, hasta 10 caracteres).

**Proceso:** Una vez realizada la petición de este requisito por parte del usuario, se listarán todos los métodos de regadíos registrados. Si el método de regadío es de tipo plan, se desplegará la información ingresada para el plan, si es de tipo reactivo, se mostrarán los datos asociados al plan reactivo.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2.** En caso de éxito, es decir que no ocurra S1, se mostrará: *nombre_ riego* (alfanumérico, hasta 30 caracteres), *fecha_implementacion* (formato fecha dd/mm/aa), *observaciones* (alfanumérico, hasta 500 caracteres), *id_usuario*

que lo creó (entero, hasta 10 caracteres), *tipo* (plan o reactivo) y *modo* (manual o automático).

Si es de tipo manual, para cada válvula del huerto se mostrará:

- Si es riego tipo plan: *tiempo* (formato hora hh:mm, obligatorio), *frecuencia* (entero, hasta 5 caracteres, obligatorio), *fecha_inicio* (formato fecha dd/mm/aa) e *id_válvula* (entero, hasta 10 caracteres, obligatorio).
- Si es riego tipo reactivo: *tiempo* (formato hora hh:mm, obligatorio), *umbral* (entero, hasta 5 caracteres, obligatorio) e *id_valvula* (entero, hasta 10 caracteres, obligatorio).

**R3. Mostrar método de riego actual**

**Descripción:** Permitirá observar el ultimo método de riego que haya sido ingresado.

**Entradas:** F*echa_creacion* (formato fecha dd/mm/aa).

**Proceso:** Una vez realizada la petición de este requisito por parte del usuario, se generará una búsqueda de la *fecha_creacion* más actual y se listarán sus datos asociados.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2.** En caso de éxito, es decir que no ocurra S1, se mostrará: *Id_metodo_riego* (entero, hasta 10 caracteres), *nombre_ riego* (alfanumérico, hasta 30 caracteres), *fecha_implementacion* (formato fecha dd/mm/aa), *observaciones* (alfanumérico, hasta 500 caracteres), *id_usuario* que lo creó (entero, hasta 10 caracteres), *tipo* (plan o reactivo) y *modo* (manual o automático).

Si es de tipo manual, para cada válvula del huerto se mostrará:

- Si es riego tipo plan: *tiempo* (formato hora hh:mm, obligatorio), *frecuencia* (entero, hasta 5 caracteres, obligatorio), *fecha_inicio* (formato fecha dd/mm/aa) e *id_válvula* (entero, hasta 10 caracteres, obligatorio).
- Si es riego tipo reactivo: *tiempo* (formato hora hh:mm, obligatorio), *umbral* (entero, hasta 5 caracteres, obligatorio) e *id_válvula* (entero, hasta 10 caracteres, obligatorio).

#### 4.3.1.4. Módulo Administración de Huerto (H)

**H1. Administrar Bombas**

**H1.1 Ingresar bomba**

**Descripción:** Permitirá ingresar una nueva bomba al huerto, un huerto puede tener varias bombas y a su vez una bomba puede tener varias válvulas asociadas. Todos los datos de entrada son obligatorios.

**Entradas:** *Id_bomba* (entero, hasta 10 caracteres, obligatorio), *nombre_bomba* (alfanumérico, hasta 30 caracteres, obligatorio), *caudal* (flotante, hasta 5 caracteres con 1 decimal, obligatorio) y *consumo_electrico* (flotante, hasta 5 caracteres con 1 decimal, obligatorio).

**Proceso:** Una vez ingresado de entrada, se validará su formato y se ingresarán a la base de datos.

Se debe validar que todos los datos de entrada obligatorios hayan sido ingresados y en el formato especificado, si alguno falta o fue ingresado en formato erróneo, indicar cuál es y permitir volver a intentarlo.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión". Permitir volver a intentarlo.

**S2:** En caso de que falten los datos especificados en la entrada; mostrar mensaje por pantalla "Datos requeridos".

**S3.** En caso de haber ingresado datos en un formato distinto al especificado; mostrar mensaje por pantalla "Datos incorrectos".

**S4.** En caso de éxito, es decir que no ocurra S1, S2 ni S3; mostrar mensaje por pantalla "Bomba ingresado correctamente".

**H1.2 Eliminar bomba**

**Descripción:** Permitirá eliminar una bomba asociada al huerto.

**Entradas:** *Id_bomba* (entero, hasta 10 caracteres, obligatorio).

**Proceso:** Una vez que el usuario seleccione la bomba a eliminar, se eliminaran sus registros de la base de datos, sin embargo no se eliminará su información asociada: las válvulas que tenga asociada y los eventos de regadío y fertilización asociados a ellas.

El dato de entrada es obligatorio, por lo que se debe validar que hayan sido ingresados y en el formato especificado, si falta o fue ingresado en formato erróneo, indicar error y permitir volver a intentarlo.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de que falte el dato especificado en la entrada; mostrar mensaje por pantalla "Dato requerido".

**S3.** En caso de éxito, es decir que no ocurra S1 ni S2; mostrar mensaje por pantalla "Bomba eliminada".

**H1.3 Editar bomba**

**Descripción:** Permitirá editar la información asociada a una bomba.

**Entradas:** **Id_bomba** (numérico, hasta 10 caracteres, obligatorio), **nuevo_nombre_bomba** (alfanumérico, hasta 30 caracteres, obligatorio), **nuevo_caudal** (flotante, hasta 5 caracteres, obligatorio), **nuevo_consumo_electrico** (flotante, hasta 5 caracteres, obligatorio).

**Proceso:** Una vez que el usuario haya seleccionado una bomba a editar, se desplegará su información actual, con posibilidad de editarla. Luego se validarán los datos de entrada y se guardarán en la base de datos.

Se debe validar que todos los datos de entrada obligatorios hayan sido ingresados y en el formato especificado, si alguno falta o fue ingresado en formato erróneo, indicar cuál es y permitir volver a intentarlo.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de que falten los datos especificados en la entrada; mostrar mensaje por pantalla "Datos requeridos".

**S3.** En caso de haber ingresado datos en un formato distinto al especificado; mostrar mensaje por pantalla "Datos incorrectos".

**S4.** En caso de éxito, es decir que no ocurra S1, S2 ni S3; mostrar mensaje por pantalla "Bomba editada correctamente".

**H2. Administrar Válvulas**

**H2.1 Ingresar válvula**

**Descripción:** Permitirá ingresar una nueva válvula al huerto, asociada una bomba. Todos los datos de entrada son obligatorios.

**Entradas:** *Id_válvula* (entero, hasta 10 caracteres, obligatorio), *nombre_válvula* (alfanumérico, hasta 30 caracteres, obligatorio) e *id_bomba* a la que pertenece (entero, hasta 10 caracteres, obligatorio).

**Proceso:** Una vez ingresado de entrada, se validará su formato y se ingresarán a la base de datos.

Se debe validar que todos los datos de entrada obligatorios hayan sido ingresados y en el formato especificado, si alguno falta o fue ingresado en formato erróneo, indicar cuál es y permitir volver a intentarlo.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de que falten los datos especificados en la entrada; mostrar mensaje por pantalla "Datos requeridos".

**S3.** En caso de haber ingresado datos en un formato distinto al especificado; mostrar mensaje por pantalla "Datos incorrectos".

**S4.** En caso de éxito, es decir que no ocurra S1, S2 ni S3; mostrar mensaje por pantalla "Válvula ingresado correctamente".

**H2.2 Eliminar válvula**

**Descripción:** Permitirá eliminar una válvula asociada al huerto.

**Entradas:** *Id_válvula* (entero, hasta 10 caracteres, obligatorio).

**Proceso:** Una vez que el usuario seleccione la válvula a eliminar, se eliminaran sus registros de la base de datos, sin embargo no se eliminará su información asociada: tal como sus nodos asociados, ni los planes de regadío y fertilización implementados para esa válvula.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de que falte el dato especificado en la entrada; mostrar mensaje por pantalla "Dato requerido".

**S3.** En caso de éxito, es decir que no ocurra S1 ni S2; mostrar mensaje por pantalla "Válvula eliminada".

**H2.3 Editar válvula**

**Descripción:** Permitirá editar la información asociada a una válvula.

**Entradas:** *Id_válvula* (entero, hasta 10 caracteres, obligatorio), *nuevo_nombre* (alfanumérico, hasta 30 caracteres, obligatorio) y *nuevo_id_bomba* a la que pertenece (entero, hasta 10 caracteres, obligatorio).

**Proceso:** Una vez que el usuario haya seleccionado una válvula a editar, se desplegará su información actual, con posibilidad de editarla. Luego se validarán los datos de entrada y se guardarán en la base de datos.

Se debe validar que todos los datos de entrada obligatorios hayan sido ingresados y en el formato especificado, si alguno falta o fue ingresado en formato erróneo, indicar cuál es y permitir volver a intentarlo.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de que falten los datos especificados en la entrada; mostrar mensaje por pantalla "Datos requeridos".

**S3.** En caso de haber ingresado los datos en un formato distinto al especificado en la entrada; mostrar mensaje por pantalla "Datos incorrectos".

**S4.** En caso de éxito, es decir que no ocurra S1, S2 ni S3; mostrar mensaje por pantalla "válvula editada correctamente".

**H3. Administrar Nodos**

**H3.1 Ingresar nodo**

**Descripción:** Permitirá ingresar un grupo de canales o sensores, llamado nodo, estos están asociados a una válvula y a su vez una válvula puede tener varios nodos.

**Entradas:** *id_nodo* (entero, hasta 10 caracteres, obligatorio), *nombre_nodo* (alfanumérico, hasta 30 caracteres, obligatorio), *tag_nodo* (alfanumérico, hasta 10 caracteres, obligatorio), *id_canal_interes*[2] (entero, hasta 10 caracteres, obligatorio), *Id_valvula* a la que pertenecen (entero, hasta 10 caracteres, obligatorio) e *id_canal_bateria*[3] (entero, hasta 10 caracteres, obligatorio).

**Proceso:** Una vez ingresado de entrada, se validarán los formatos y se ingresarán a la base de datos.

Se debe validar que todos los datos de entrada obligatorios hayan sido ingresados y en el formato especificado, si alguno falta o fue ingresado en formato erróneo, indicar cuál es y permitir volver a intentarlo.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de que falten los datos especificados en la entrada; mostrar mensaje por pantalla "Datos requeridos".

**S3.** En caso de haber ingresado datos en un formato distinto al especificado en la entrada; mostrar mensaje por pantalla "Datos incorrectos".

---

[2] Un canal de interés será aquel sensor cuya lectura de humedad representará a todo el nodo, será definido por el administrador.

[3] El canal batería será un quinto sensor, cuya lectura representará la cantidad de batería que le queda al nodo.

**S4.** En caso de éxito, es decir que no ocurra S1, S2 ni S3; mostrar mensaje por pantalla "Válvula ingresado correctamente".

**H3.2 Eliminar nodo**

**Descripción:** Permitirá eliminar un nodo de sensores asociada al huerto.

**Entradas:** *Id_nodo* (entero, hasta 10 caracteres, obligatorio).

**Proceso:** Una vez que el usuario seleccione el nodo a eliminar, se eliminaran sus registros de la base de datos y todos sus canales asociados.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de que falte el dato especificado en la entrada; mostrar mensaje por pantalla "Dato requerido".

**S3.** En caso de éxito, es decir que no ocurra S1 ni S2; mostrar mensaje por pantalla "Nodo eliminado".

**H3.3 Editar nodo**

**Descripción:** Permitirá editar la información asociada a un nodo.

**Entradas:** *Id_nodo* (entero, hasta 10 caracteres, obligatorio), *nuevo_nombre_nodo* (alfanumérico, hasta 30 caracteres, obligatorio), *nuevo_tag_nodo* (alfanumérico, hasta 10 caracteres, obligatorio), *nuevo_id_canal_interes* (entero, hasta 10 caracteres, obligatorio) y *nuevo_id_valvula* a la que pertenecen (entero, hasta 10 caracteres, obligatorio).

**Proceso:** Una vez que el usuario haya seleccionado un nodo a editar, se desplegará su información actual, con posibilidad de editarla. Luego se validarán los datos de entrada y se guardarán en la base de datos.

Se debe validar que todos los datos de entrada obligatorios hayan sido ingresados y en el formato especificado, si alguno falta o fue ingresado en formato erróneo, indicar cuál es y permitir volver a intentarlo.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de que falten los datos especificados en la entrada; mostrar mensaje por pantalla "Datos requeridos".

**S3.** En caso de haber ingresado datos en un formato distinto al definido en la entrada; mostrar mensaje por pantalla "Datos incorrectos".

**S4.** En caso de éxito, es decir que no ocurra S1, S2 ni S3; mostrar mensaje por pantalla "nodo editado correctamente".

**H4. Administrar Canales**

**H4.1 Ingresar canal**

**Descripción:** Permitirá ingresar un sensor, llamado canal, que pertenecerá a un nodo de canales. Uno de estos canales será el canal de interés, cuyos datos de lectura de humedad guiarán el regadío. Todos los datos de entrada son obligatorios.

**Entradas:** *Id_canal* (entero, hasta 10 caracteres, obligatorio), *número_canal* (entero, 1 caracter, obligatorio), *id_nodo* al que pertenece (entero, hasta 10 caracteres, obligatorio), *tipo_sensor* (entero, hasta 10 caracteres, obligatorio), *nombre_canal* (alfanumérico, hasta 100 caracteres, obligatorio) y *unidad_medida* (alfanumérico, hasta 10 caracteres, obligatorio).

**Proceso:** Una vez ingresado de entrada, se validará el formato y se ingresarán a la base de datos.

Se debe validar que todos los datos de entrada obligatorios hayan sido ingresados y en el formato especificado, si alguno falta o fue ingresado en formato erróneo, indicar cuál es y permitir volver a intentarlo.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de que falten los datos especificados en la entrada; mostrar mensaje por pantalla "Datos requeridos".

**S3.** En caso de haber ingresado datos en un formato distinto al especificado; mostrar mensaje por pantalla "Datos incorrectos".

**S4.** En caso de éxito, es decir que no ocurra S1, S2 ni S3; mostrar mensaje por pantalla "Datos ingresado correctamente".

**H4.3 Editar canal**

**Descripción:** Permitirá editar la información asociada a un canal.

**Entradas:** *Id_canal* (entero, hasta 10 caracteres, obligatorio) *nuevo_tipo* sensor (entero, hasta 10 caracteres, obligatorio), *nombre_canal* (alfanumérico, hasta 100 caracteres, obligatorio) y *nueva_unidad_medida* (alfanumérico, hasta 10 caracteres, obligatorio).

**Proceso:** Una vez que el usuario haya seleccionado un canal a editar, se desplegará su información editable. Luego se validarán los datos de entrada y se guardarán en la base de datos.

Se debe validar que todos los datos de entrada obligatorios hayan sido ingresados y en el formato especificado, si alguno falta o fue ingresado en formato erróneo, indicar cuál es y permitir volver a intentarlo.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de que falten los datos especificados en la entrada; mostrar mensaje por pantalla "Datos requeridos".

**S3.** En caso de haber ingresado datos en un formato distinto al especificado en la entrada; mostrar mensaje por pantalla "Datos incorrectos".

**S4.** En caso de éxito, es decir que no ocurra S1, S2 ni S3; mostrar mensaje por pantalla "canal editado correctamente".

## H5. Administrar cuentas de usuarios

### H5.1 Ingresar usuarios

**Descripción:** El usuario (administrador de huerto) podrá crear una cuenta de usuario, a partir del llenado de un formulario de inscripción donde irán los datos más relevantes del nuevo usuario.

**Entradas:** *Nombre* (texto, hasta 20 caracteres, obligatorio), *apellidos* (texto, hasta 40 caracteres, obligatorio), *RUT* (formato RUT Chileno, obligatorio), *tipo_usuario* (monitor, controlador o administrador_huerto), *correo* (formato e-mail, no obligatorio), *dirección* (alfanumérico, hasta 100 caracteres, obligatorio), *teléfono* (numérico, hasta 10 caracteres, no obligatorio) y *contraseña* (alfanumérico, 40 caracteres, obligatorio).

**Proceso:** Validar los datos de entrada e ingresarlos a la base de datos. El RUT ingresado será guardado en dos campos en la base de datos, en el correspondiente al RUT del usuario, siendo éste su identificador y en el campo correspondiente a la contraseña; sin puntos, con guión y digito verificador (ej. 12345678-9).

Se debe validar que todos los datos de entrada obligatorios hayan sido ingresados y en el formato especificado, si alguno falta o fue ingresado en formato erróneo, indicar cuál es y permitir volver a intentarlo.

**Salida:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de que falten los datos especificados en la entrada; mostrar mensaje por pantalla "Datos requeridos".

**S3.** En caso de haber ingresado datos en el formato distinto al indicado en la entrada; mostrar mensaje por pantalla "Datos incorrectos".

**S4.** En caso de éxito, es decir que no ocurra S1, S2 ni S3; mostrar mensaje por pantalla "Usuario ingresado, desde ahora podrá ingresar con su RUT como usuario y contraseña".

#### H5.2 Modificar usuarios

**Descripción:** El usuario administrador de huerto podrá editar la información de los demás usuarios asociados a su huerto.

**Entradas:** *RUT* (formato Rut Chileno, obligatorio), *nuevo_nombre* (texto, hasta 20 caracteres, obligatorio), *nuevo_apellidos* (texto, hasta 40 caracteres, obligatorio), *nuevo_tipo_usuario* (monitor, controlador o administrador_huerto. Obligatorio), *nuevo_correo* (formato e-mail, no obligatorio), *nueva_dirección* (alfanumérico, hasta 100 caracteres, obligatorio) y *nuevo_teléfono* (numérico, hasta 10 caracteres, no obligatorio).

**Proceso:** Validar los datos de entrada y modificar cuentas de usuario de la base de datos.

# TESIS 1

# FIA PYT-2009-0259

Se debe validar que todos los datos de entrada obligatorios hayan sido ingresados y en el formato especificado, si alguno falta o fue ingresado en formato erróneo, indicar cuál es y permitir volver a intentarlo.

**Salida:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de que falten los datos especificados en la entrada; mostrar mensaje por pantalla "Datos requeridos".

**S3.** En caso de haber ingresado datos en un formato distinto al indicado en la entrada; mostrar mensaje por pantalla "Datos incorrectos".

**S4.** En caso de éxito, es decir que no ocurra S1, S2 ni S3; mostrar mensaje por pantalla "Usuario Modificado".

**H5.3 Eliminar usuarios**

**Descripción:** El usuario (administrador de huerto) podrá eliminar a los demás usuarios asociados a su huerto.

**Entradas:** *RUT* del usuario a eliminar (formato Rut Chileno, obligatorio).

**Proceso:** Se listarán los usuarios existentes en el huerto, para que el administrador pueda seleccionar uno de ellos. Una vez seleccionado eliminar sus registros de la base de datos.

**Salida:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2.** En caso de éxito, es decir que no ocurra S1; mostrar mensaje por pantalla "usuario eliminado".

**H5.4 Listar usuarios del huerto**

**Descripción:** Esta función listará todos los usuarios del sistema que pertenezcan al huerto.

**Entradas:** No hay entradas.

**Proceso:** Se realizará una búsqueda de todos los usuarios registrados en el sistema y se mostrarán por pantalla.

**Salida:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2.** En caso de éxito, es decir que no ocurra S1; mostrar por pantalla a cada uno de los usuarios con la siguiente información: *tipo_usuario* (monitor, controlador o administrador_huerto), *nombre* (texto, hasta 20 caracteres), *apellidos* (texto, hasta 40 caracteres), *RUT* (formato Rut Chileno), *correo* (formato e-mail), *dirección* (alfanumérico, hasta 100 caracteres) y *teléfono* (numérico, hasta 10 caracteres).

**H6. Mostrar configuración actual del huerto**

**Descripción:** Esta función mostrará todos los elementos del huerto, como bombas, válvulas, nodos y canales.

**Entradas:** *id_bomba* (entero, hasta 10 caracteres).

**Proceso:** Se realizará una lectura de todos los elementos del huerto y se mostrarán por pantalla.

**Salida:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2.** En caso de éxito, es decir que no ocurra S1; mostrar *nombre_bomba* (alfanumérico, hasta 30 caracteres), *nombre_válvula* (alfanumérico, hasta 30 caracteres), *nombre_nodo* (alfanumérico, hasta 30 caracteres) y *nombre_canal* (alfanumérico, hasta 30 caracteres).

#### 4.3.1.5. Administración de cuentas (C)

**C.1 Autentificar cuenta de usuario**

**Descripción:** Esta función satisfará la necesidad de autentificar al usuario, para que ingrese al sistema de forma segura.

**Entradas:** *RUT* (formato RUT Chileno, obligatorio), *tipo_usuario* (monitor, controlador o administrador_huerto. Obligatorio) y *contraseña* (alfanumérico, hasta 40 caracteres, obligatorio).

**Proceso:** Validar los datos de entrada y si son correctos, verificar si existe el usuario en la base de datos mediante el RUT y la contraseña. Una vez encontrado se permitirá el ingreso al sistema, con las funciones que correspondan a su tipo de usuario (ver diagrama de casos de usos).

Se debe validar que todos los datos de entrada obligatorios hayan sido ingresados y en el formato especificado, si alguno falta o fue ingresado en formato erróneo, indicar cuál es y permitir volver a intentarlo.

**Salidas:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de que falten los datos especificados en la entrada; mostrar mensaje por pantalla "Datos requeridos".

**S3.** En caso de haber ingresado datos en un distinto al indicado en la entrada; mostrar mensaje por pantalla "Datos incorrectos".

**S4.** En caso de éxito, es decir que no ocurra S1, S2 ni S3, mostrar el índex del sistema.

**C.2 Editar datos de usuario**

**Descripción:** Todo usuario podrá editar la información asociada a su cuenta.

**Entradas:** *RUT* (formato RUT Chileno, obligatorio), *nuevo_correo* (formato e-mail, no obligatorio), *nueva_dirección* (alfanumérico, hasta 100 caracteres, obligatorio), *nuevo_teléfono* (numérico, hasta 10 caracteres, no obligatorio) y *nueva_contraseña* (alfanumérico, hasta 10 caracteres, obligatorio).

**Proceso:** Validar los datos de entrada y modificar la cuenta del usuario de la base de datos.

Se debe validar que todos los datos de entrada obligatorios hayan sido ingresados y en el formato especificado, si alguno falta o fue ingresado en formato erróneo, indicar cuál es y permitir volver a intentarlo.

**Salida:** Las salidas dependerán del resultado del proceso.

**S1:** En caso de error, por no obtener respuesta de la base de datos; mostrar mensaje por pantalla "Error de conexión".

**S2:** En caso de que falten los datos especificados en la entrada; mostrar mensaje por pantalla "Datos requeridos".

**S3.** En caso de haber ingresado datos en un formato distinto al indicado en la entrada; mostrar mensaje por pantalla "Datos incorrectos".

**S4.** En caso de éxito, es decir que no ocurra S1, S2 ni S3; mostrar mensaje por pantalla "Cuenta Modificada".

### 4.3.2. Requisitos no funcionales

#### 4.3.2.1. Seguridad

Todo ingreso al sistema es bajo la identificación de un usuario y su contraseña en una sesión persistente dentro del sistema. Las actividades de ingreso al sistema y salidas del sistema, deben ser registrada en una tabla de registro del actividades, identificando al usuario que lo realizo, la fecha, hora y la actividad realizada.

Además cada vez que un usuario que haya iniciado su sesión, y no haya interactuado con él en más de una hora; se cerrará el acceso al sistema por inactividad.

#### 4.3.2.2. Portabilidad

Es de gran importancia que el sistema sea multiplataforma, es decir, que se pueda acceder a él desde distintos navegadores, tanto de computadores, como de dispositivos móviles como celulares, ipad, pda, etcétera, por lo tanto el uso de bibliotecas o elementos gráficos es de libre elección, siempre y cuando éstas no utilicen flash, pues imposibilita la visualización desde equipos Apple, muy usados por los usuarios finales del sistema.

#### 4.3.2.3. Requisitos de Licencias

El uso de herramientas de desarrollo, complementables con las ya exigidas, es a elección de los desarrolladores. Sin embargo, toda herramienta utilizada debe tener licencia Open Source.

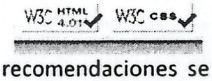
# Capítulo V: Análisis de Tecnologías de Desarrollo

## *5.1. Web 2.0 dentro de la Interfaz*

El primer aspecto que clasifica a este proyecto como Web 2.0 es justamente su desarrollo web, pues como se observa en el estado del arte, las aplicaciones similares existentes en el mercado son aplicaciones de escritorio, sin embargo en este caso hemos transformado un software comúnmente de escritorio hacia la plataforma Web. Además sin la interacción de los usuarios ésta web no tendría sentido, pues son ellos quienes le darán sentido a la web, implementando planes de regadío, de fertilización y configurando el huerto, lo que también clasifica al proyecto como Web 2.0.

Otras tecnologías de la Web 2.0, usadas en la interfaz, se describen a continuación.


El uso de CSS dentro de la interfaz permite un control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la mantención del mismo, pues los estilos se controlan solo de un documento y no en todas las páginas que conformen el sistema. Además, hace que la codificación sea más ordenada, claro de entender y se consigue reducir considerablemente las líneas de código (W3C, 2008).




La W3C es un Consorcio que desarrolló un conjunto de estándares de alta calidad para las tecnologías Web, siguiendo estas recomendaciones se puede certificar la calidad del sistema implementado, y es en ese punto donde radica la importancia de incluir este estandar dentro del proyecto (W3C, 2008).


jQuery es una biblioteca de JavaScript, libre y de código abierto, que simplifica la manera de interactuar con los documentos HTML, manejar eventos, desarrollar animaciones, que permiten hacer la interfaz mucho más atractiva al usuario y agregar interacción con la técnica AJAX a páginas web. Ofrece una serie de herramientas que de manera tradicional requerirían de mucho más líneas de código, por lo que la representación de los elementos de interacción a través de jQuery generará grandes resultados en menos tiempo y espacio (Fica, 2010).


Si bien Ajax es una forma asíncrona de comunicación con el servidor, lo que corresponde al *"modelo"* del sistema y no a la interfaz, será mencionada dentro de este proyecto ya que es necesario modelar la interfaz para el uso óptimo de Ajax, de esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas (w3schools, 2008).

## *5.2. MVC dentro de la Interfaz*

La separación de las responsabilidades otorga claras ventajas a la hora de desarrollar un sistema, pues permite definir claramente el rol de cada integrante del equipo desarrollador, se puede trabajar con múltiples lenguajes, distintos diseños de presentación, integrar distintas librerías y tecnologías sin alterar la lógica de negocio. Esta separación es fundamental para el desarrollo de arquitecturas consistentes, reutilizables y fácilmente mantenibles, dado que las correcciones solo se deben hacer en el lugar del error, lo que al final resulta en un ahorro de tiempo en desarrollo en posteriores proyectos o evolución de este mismo.

Sin embargo se debe dejar en claro que por el hecho de usar esta arquitectura se agrega un nivel extra de dificultad, pues la curva de aprendizaje es más alta, que las de otras arquitecturas, dado que los sistemas se complejizan por contener más archivos de configuración.

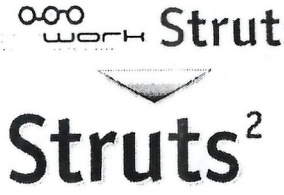
### 5.2.1. Frameworks para MVC

EL Framework o marco de trabajo en el desarrollo de software es una estructura de soporte y una metodología de trabajo que ayuda la organización y desarrollo de un proyecto. Permite pasar más tiempo identificando requerimientos del software que tratando con los tediosos detalles de bajo nivel necesarios para entregar un sistema funcional.

Típicamente, un Framework puede incluir soporte de programas, bibliotecas y un lenguaje de scripting para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Estos componentes son personalizables e intercambiables, tales como automatización de validadores, reutilización de código, conversiones de datos, entre otros.

J2EE es una herramienta bastante adecuada para el desarrollo de este proyecto, pues permite cubrir las necesidades de páginas dinámicas en JSP y la lógica de negocio mediante JAVA y a su vez mantiene interoperabilidad con otras tecnologías como XML, JavaScript, HTML, CSS, entre otras. Su licencia es Open Source, con una gran cantidad de herramientas gratuitas, tal como Eclipse, que nos permite desarrollar proyectos en MVC, a través del framework **Strust 2**.

### 5.2.2. Framework de desarrollo: Struts2




Struts 2, tal como se necesita, *"proporciona un conjunto de utilidades cuyo objetivo es facilitar y optimizar los desarrollos de aplicaciones Web con tecnologías J2EE, siguiendo el patrón MVC"* (Martín Sierra, 2008), introduciendo componentes que lo hacen flexible para el desarrollo de aplicaciones de web de cualquier tamaño; está diseñado y creado para agrupar todas las fases de desarrollo de una aplicación, incluso la mantención.

#### 5.2.2.1. Componentes de Struts 2

A continuación se darán a conocer las principales funciones y características de los componentes y capas de una aplicación Struts 2, a lo largo del ciclo de vida de una petición, es decir, desde que llega al controlador, hasta que se envía la respuesta al cliente.

*FilterDispatcher*

Este componente es la entrada de las peticiones de una aplicación (Ver elipses verdes del esquema 2), analiza la petición y determina el mapeo de la acción (URL). Por defecto, Struts2 buscará la extensión .action. Filter Dispatcher crea el ActionProxy, que es la clase que contiene toda la configuración e información de contexto para procesar la petición y debería contener los resultados de la ejecución después de que haya sido procesada.

**Comentario [YG14]:** Justificación esquema 2

*Interceptores*

Una vez determinada la acción a ejecutar, la petición pasa al control de la información entrante a través de los interceptores (Ver rectángulos naranjos del esquema 2). Los interceptores son una cadena de objetos que realizan tareas de pre-procesamiento, por

**Comentario [YG15]:** Justificación esquema 2

ejemplo validaciones de datos de un usuario. Para decidir cuáles de ellos se ejecutarán, se debe indicar en un archivo de configuración llamado **struts.xml.** El API de Struts incorpora un gran número de interceptores y a su vez se pueden implementar interceptores propios.

*Action*

Las Actions son el núcleo del Framework Struts 2 (Ver figuras moradas del esquema 2). Cada dirección URL tiene mapeada a una acción específica la cual provee lógica de proceso para la petición del usuario. Las Actions serán las encargadas de ejecutar la lógica necesaria para manejar una petición determinada. Un Action puede tener más de un result asociado. Esto permitirá enviar al usuario a una vista distinta dependiendo del resultado de la ejecución del Action.

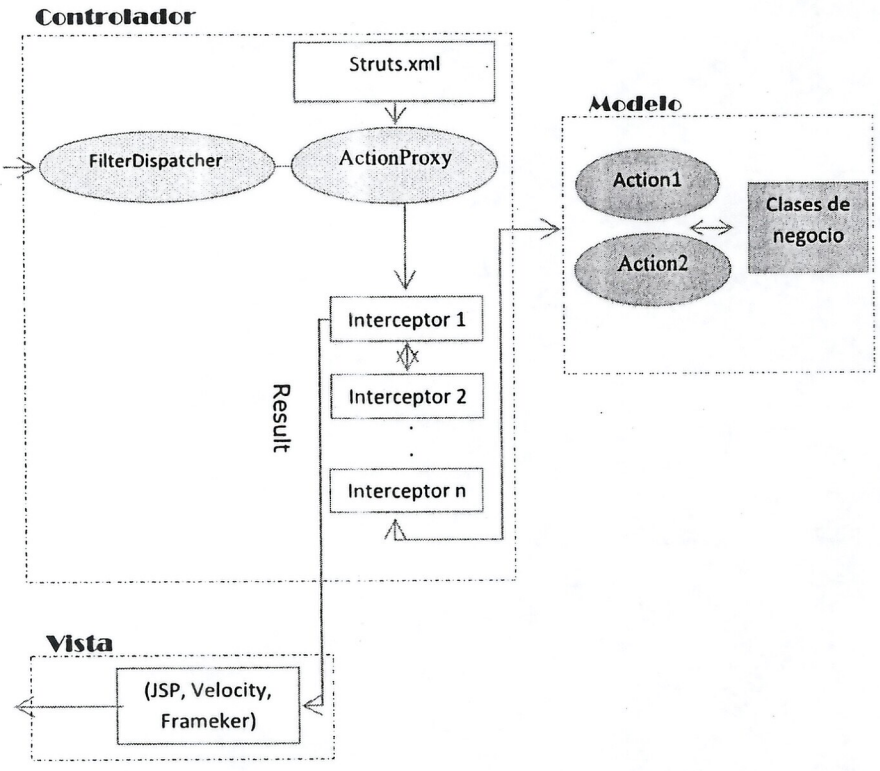
**Comentario [YG16]:** Justificación esquema 2

*Results*

En Struts 2, la capa de vista está encapsulada en la componente Result (Ver rectángulo azul del esquema 2). Después que un Action ha sido procesado se debe enviar la respuesta de regreso al usuario, esto se realiza usando results. Este proceso tiene dos componentes, el tipo del result y el result mismo.

**Comentario [YG17]:** Justificación esquema 2

El tipo del result indica cómo debe ser tratado el resultado que se le regresará al cliente. Por ejemplo un tipo de Result puede enviar al usuario de vuelta una JSP, otro puede redirigirlo hacia otro sitio, mientras otro puede enviarle un flujo de bytes.




Esquema 2: Componentes y capas de una aplicación Struts 2.

**Fuente:** (Martín Sierra, 2008)

Para comprender mejor de donde vienen los datos que muestra la interfaz, presentamos el siguiente diagrama del ciclo de vida de una petición, desde que llega al controlador, hasta que se envía la respuesta al cliente.

**Comentario [YG18]:** Justificación figura 7.

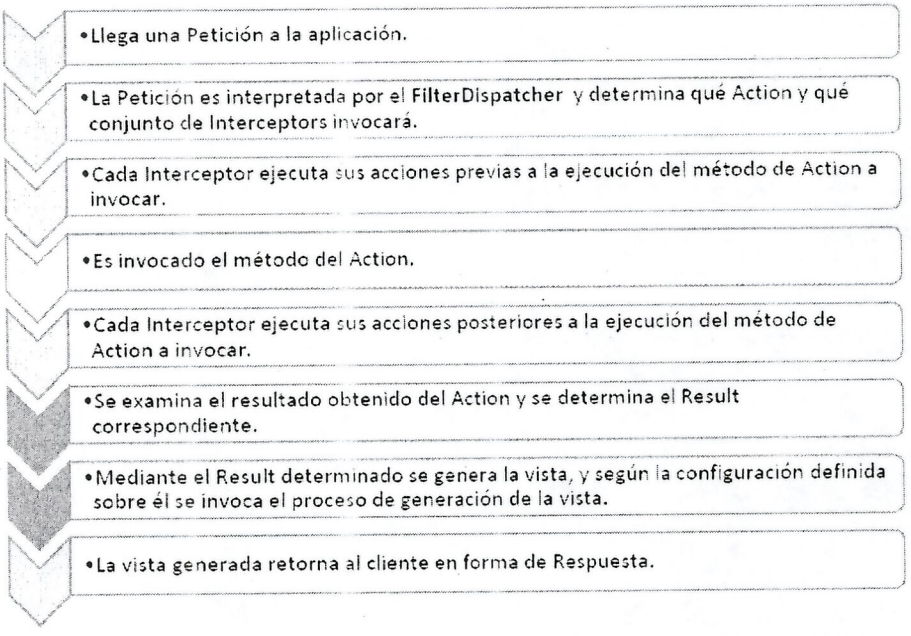



Figura 7: Ciclo de vida de una petición en el framework Struts 2.

Fuente: (Gómez García, 2008)

Struts permite incorporar código html en cada página.jsp, pero varía de este último en la utilización de tag para manipular los datos y controlarlos.

#### 5.2.2.1.1. Librería de Acciones Struts- Tags

Para la creación de vistas en Struts 2 se utilizan acciones JSP, acciones que forman parte de la librería struts-tags. Esta librería permite una intersección entre las acciones y las vistas, favoreciendo la mantenibilidad y permitiendo el encapsulamiento de la lógica, reduciendo así la repetición de código. Los tipos de Tag son:

***Acciones de Manipulación de datos (Data Tag):*** Acciones para la manipulación y extracción de datos; recuperan, añaden o manipulan el valor de una determinada variable.

**Acciones de Control (Control Tag):** Son utilizadas para el control del flujo de la aplicación.

**Acciones UI:** Se usan para la generación de controles gráficos en formularios HTML. Estos componentes encapsulan las funcionalidades necesarias para capturar los datos, validarlos y a su vez insertarlos con la ayuda de interceptores predefinidos de Struts 2 en las propiedades del objeto Action que atenderá la petición.

Todos tienen una variada cantidad de propiedades, que facilitan el desarrollo a bajo nivel.

**Comentario [YG19]:** Lo resumí y elimine el código, dado que aquí no se justificaba.

### 5.2.3. Integrando Struts2 con otras Tecnologías

Una de las ventajas que tiene Struts2 es la existencia de una gran cantidad Plugins que añaden funcionalidades al Framework y a su vez la integración con otros Frameworks que manejen distintos aspectos del proyecto, permitiendo incorporar en la vista diferentes tecnologías de representación que facilitan la integración y mejoran la productividad ampliando el rango de uso. Particularmente para este proyecto de desarrollo de una interfaz se han usado las siguientes tecnologías.

### 5.2.3.1. Eclipse

Para el desarrollo de una aplicación es muy recomendable de hacer uso de algún programa de ayuda al desarrollo. Eclipse es una plataforma con módulos base preparados para el desarrollo de aplicaciones Web en Java. Su uso se adecua al proyecto dado que es una herramienta libre, lo que está acorde a los objetivos de este proyecto, y porque gracias a Eclipse se puede realizar una buena estructuración de los recursos de la aplicación. Además proporciona un conjunto de métodos y propiedades que ayudan a depurar el código (debug) y analizarlo en detalle.

### 5.2.3.2. Tiles

Normalmente dentro del desarrollo de interfaces (html, jsp) se encuentran componentes en común, como la cabecera (Header), el cuerpo (Body), el pie de página (Footer), el menú, entre otros. Generalmente para mantener una concordancia de estos componentes a través de todo el sitio se usan plantillas (Templantes) y esquemas (Layout). En este proyecto usaremos un Framework que explora soluciones más poderosas y flexibles llamado **Tiles.** Para comprender estas soluciones y los atributos de calidad que otorga Tiles a la interfaz, visualizar los siguientes ejemplos.

Comúnmente se da esta situación.

**Comentario [YG20]:** En este caso se justifica la inclusión de código, por el motivo que se destaca.

Pag_1.jsp

```
<html>
<body>
   Header h...
   <p>contenido 1</p>
   Footer F...
</body>
</html>
```

Pag_n.jsp

```
<html>
<body>
   Header h..
   <p>contenido n</p>
   Footer F...
</body>
</html>
```

En el ejemplo anterior se ve la repetición en cada página del código en común, como header y footer, este modo de implementar un sistema web no es deseable porque cambios en un componente de vista requiere cambios de cada una de las páginas.

Cada una de las paginas es responsable de colocar (layout) sus componentes de vista. Esta solución simple carece de previsión y tiene un pesado costo de mantenimiento.

Como solución a lo anterior **Framework Tiles component view** permite la siguiente estructura.

Pag_1.jsp

```
<%@ taglib uri="/WEBINF/tiles.tld" prefix="tiles" %>
<tiles:insert page="/layout.jsp" flush="true">
<tiles:put name="header" value="/header.jsp"/>
<tiles:put name="body"  value="/aBody.jsp"/>
<tiles:put name="footer" value="/footer.jsp"/>
</tiles:insert
```

Pag_n.jsp

```
<%@ taglib uri="/WEBINF/tiles.tld" prefix="tiles" %>
<tiles:insert page="/layout.jsp" flush="true">
<tiles:put name="header" value="/header.jsp"/>
<tiles:put name="body"  value="/aBody.jsp"/>
<tiles:put name="footer" value="/footer.jsp"/>
</tiles:insert
```

Layout.jsp

```
<%@ taglib uri="/WEB-INF/tiles.tld" prefix="tiles" %>
<html>
<body>
   <%-- include header --%>
   <tiles:insert attribute="header"/>
    <%-- include body --%>
     <tiles:insert attribute="body"/>
     <%-- include footer --%>
     <tiles:insert attribute="footer"/>
</body>
</html>
```

En el ejemplo anterior se utiliza la característica de Tiles que permite definir los Layouts como plantillas. Desde estos layouts se puede insertar marcadores de lugar, en vez del componente de vista en sí, usando la etiqueta de Tiles **insert**. Así solo se necesita cambiar los componentes de vista comunes una vez, por tanto, esta solución elimina (en parte) la duplicidad de código y simplifica el mantenimiento. Estos layout reutilizables encapsulan todos los componentes comunes del sistema y permite reducir drásticamente el acoplamiento entre los componentes que se nos presentaban en el primer ejemplo.

Sin embargo, aún se encuentran algunas desventajas, dado que de todos modos se repite la configuración en común en cada una de las páginas.

Para solucionar completamente la duplicidad de código encontradas en el ejemplo anterior, **es que se integró Tiles con Struts 2,** dado que Tiles perfecciona las etiquetas que Struts provee, y a su vez Struts provee un archivo XML de configuración donde se especifican todas las páginas de contenido de una sola vez. Este archivo se llamará *"tiles.xml"*.

Tiles.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<component-definitions>
  <definition name="baseDef" path="/layout.jsp">
      <put name="header" value="/header.jsp"/>
      <put name="footer" value="/footer.jsp"/>
      <put name="body" value=""/>
  </definition>

  <definition name="pag_1" extends="baseDef">
      <put name="body" value="/aBody.jsp"/>
  </definition>

  <definition name="pag_2" extends="baseDef">
      <put name="body" value="/bBody.jsp"/>
  </definition>

  <definition name="pag_n" extends="baseDef">
      <put name="body" value="/cBody.jsp"/>
  </definition>
</component-definitions
```

Esta integración elimina todas las páginas de contenido .jsp poniendo sus definiciones en un archivo XML. Como se observa en el ejemplo anterior, una poderosa característica de Tiles es la herencia entre definiciones, donde se crea una definición "padre", con todos los Layout o características comunes y las demás definiciones "hijas" heredan de esta a través de la etiqueta **extends**. La definición hija solo debe definir sus componentes únicos.

Ahora para llamar a cada página y acceder al archivo XML *"tiles.xml"* se debe especificar dentro del archivo de configuración de Struts como otro parámetro en el *"struts.xml"* de la siguiente forma.

```
<init-param>
<param-name>definitions-config</param-name>
<param-value> /WEB-INF/tiles.xml </param-value>
</init-param>
```

*Ventajas de usar Tiles integrado con Struts 2*

Finalmente se implementó en la interfaz Tiles integrado con Struts2 dadas las notables ventajas encontradas y que se resumen a continuación.

Facilita el desarrollo y mantención: Dada la reducción demostrada del número de páginas, es posible disminuir considerablemente la complejidad de desarrollo, de administración y mantenimiento de la aplicación.

Reduce la repetición de código: Cuanto más código repetido más difícil es la tarea de desarrollar y mantener la aplicación, dado que un simple cambio puede desencadenar una serie de cambios en cascada en muchas páginas diferentes con consecuencias imprevisibles, sin embargo como Tiles encapsulada en una sola página todos los componentes comunes del sistema, dando la posibilidad de herencia entre definiciones, se logra eliminar por completo la repetición de código.

Reduce el acoplamiento: El acoplamiento es el grado de interdependencia entre entidades, Tiles permite reducir enormemente el acoplamiento entre los componentes, dado que encapsula el comportamiento de los layout en común.

### 5.2.3.3. Jquery

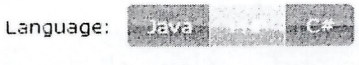
En la sección 3.2 de Web 2.0 se dio a conocer Jquery y sus aportes a la interfaz. Además de su uso como biblioteca se encontró el plugin Struts-JQuery, que permite importar las librerías JQuery. Como resultado se obtiene la posibilidad de, mediante el uso de una serie de Tags, producir código jQuery adaptado a una aplicación de una manera muy veloz y productiva. Una vez incorporado el plugin Struts2-jquery, el uso de los tags es bastante similar a los ya ejemplificados de struts. Simplemente se declara en la cabecera de la página a utilizarlo de la siguiente forma.

```
<%@taglib uri="/struts-jquery-tags" prefix="sj" %>
```

Luego para usar un elemento, se utiliza el prefijo definido en la cabecera, en este caso es **sj**, de la siguiente forma.

```
<sj:checkboxlist id="checks" label="Languaje" list="{'Java', 'PHP', C#'}"/>
```

De las líneas anteriores se obtiene el siguiente checkbox (el cuadro verde representa el check seleccionado), que permite la integración con tecnología Ajax y hacer mucho más atractiva de navegación al usuario, dado que esta forma de representar un checkbox es una forma de abstraer la cotidianeidad de encender un electrodoméstico de uso diario.




Aquí es donde reside la potencia y utilidad del plugin struts2-jquery. Con unos pocos tags se creó una lista de atractivos check box.

### 5.2.3.4. Dygraphs

Dygraphs es una biblioteca gráfica muy completa para crear gráficas en JavaScript. Tiene licencia Open source, lo que está acorde a los objetivos de este proyecto. Entre las características más interesantes de Dygraph es la opción de dibujar gráficos de varios tipos (curvas, torta, barras, logarítmicos, exponenciales, senoidales, entre otras), con la posibilidad de realizar zoom en ellos, y también visualizar popup con información del punto seleccionado al pasar el puntero del ratón en una determinada área gráfica, como se muestra en la Figura 8, gracias a que sus gráficas son desplegadas en formato SVG (Gráficos Vectoriales Escalables). Además al hacer uso de gráficos en este formato, se está adquiriendo un atributo de calidad al sistema, dado que SVG se convirtió en una recomendación del W3C en septiembre de 2001. (W3C, 2010)

**Comentario [YG21]:** Justificación figura 8.

Dygraphs ofrece compatibilidad con los navegadores más usado y por el hecho de no usar flash en sus gráficas, puede ser visualizado en navegadores de dispositivos móviles, como Iphone, Ipad, entre otros, acorde a las solicitudes planteadas por los usuarios. Otro de los motivos que impulsó a la elección de esta biblioteca, por sobre otras con gráficas igualmente atractivas, es su característica de ser liviano y rápido, con apenas 45 KB de promedio para cada gráfica y a su vez es capaz de procesar una gran cantidad de datos en forma simultánea, condición bastante común en el sistema implementado.

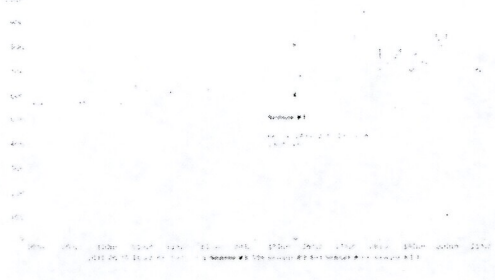


Figura 8: Gráfico generado con Dygraphs

**Fuente:** Propia.

# Capítulo VI. Diseño y Desarrollo de la Interfaz

## *6.1. Diseño de las Interfaces del Sistema*

El diseño de la interfaces del sistema se dividirá por módulos, los mismos módulos descritos en la especificación de requisitos. A su vez cada vez que se refiera a una funcionalidad se utilizará la misma notación dicha especificación de requisitos (Capítulo IV).

### 6.1.1. Ingreso al Sistema (c1)

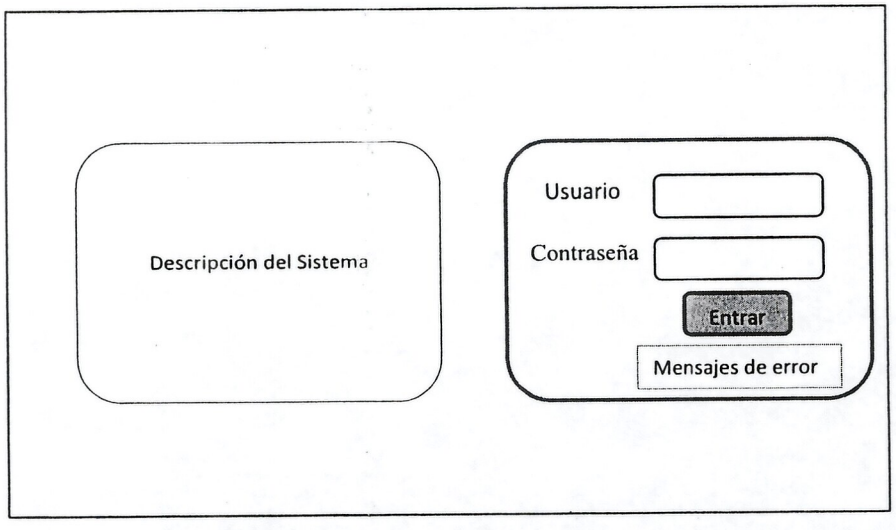



Figura 9: Diseño de la interfaz de ingreso al sistema.

Fuente: Propia.

### 6.1.2. Inicio del Sistema

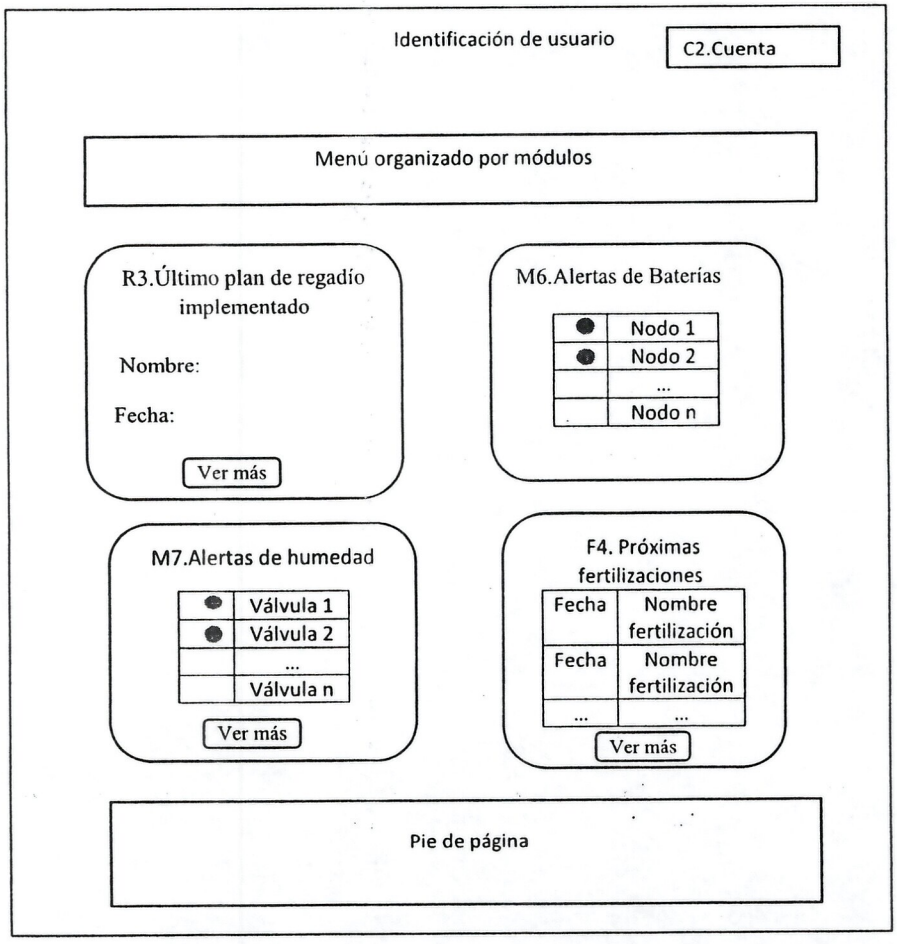



Figura 10: Diseño de la interfaz de inicio del sistema

Fuente: Propia.

### 6.1.3. Diseño de la Interfaz de Fertilización

#### 6.1.3.1. Diseño de la interfaz de administración de planes de fertilización

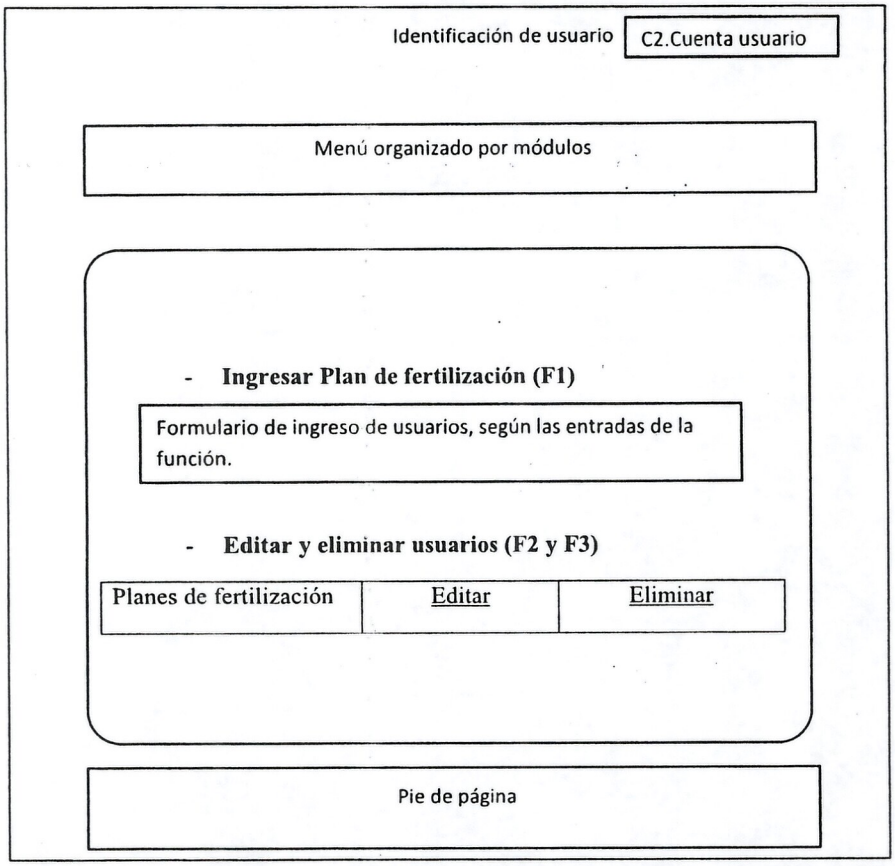



Figura 11: Diseño de la interfaz de administración de planes de fertilización.

Fuente: Propia.

### 6.1.3.2. Diseño de la interfaz para mostrar planes de fertilización activas (F4).

Este diseño de esta interfaz es análogo para mostrar las fertilizaciones pasadas (F5).

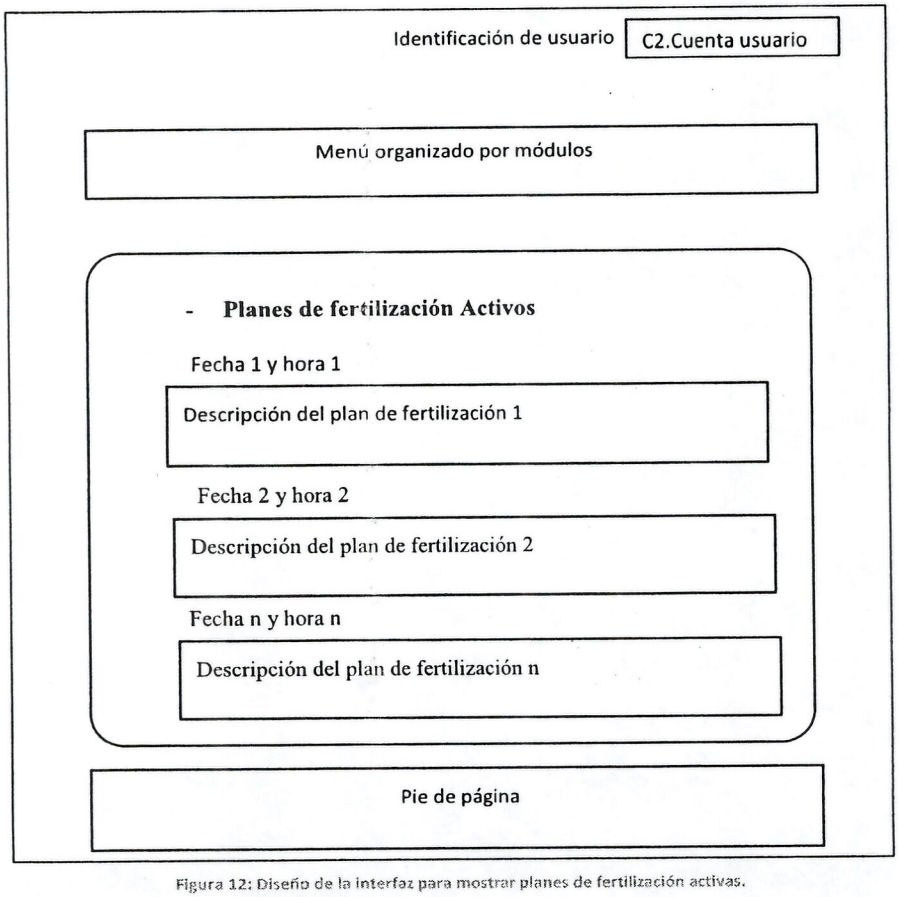



Figura 12: Diseño de la interfaz para mostrar planes de fertilización activas.

Fuente: Propia.

### 6.1.4. Diseño de la Interfaz de Monitoreo

#### 6.1.4.1. Monitorear por nodos (M1)

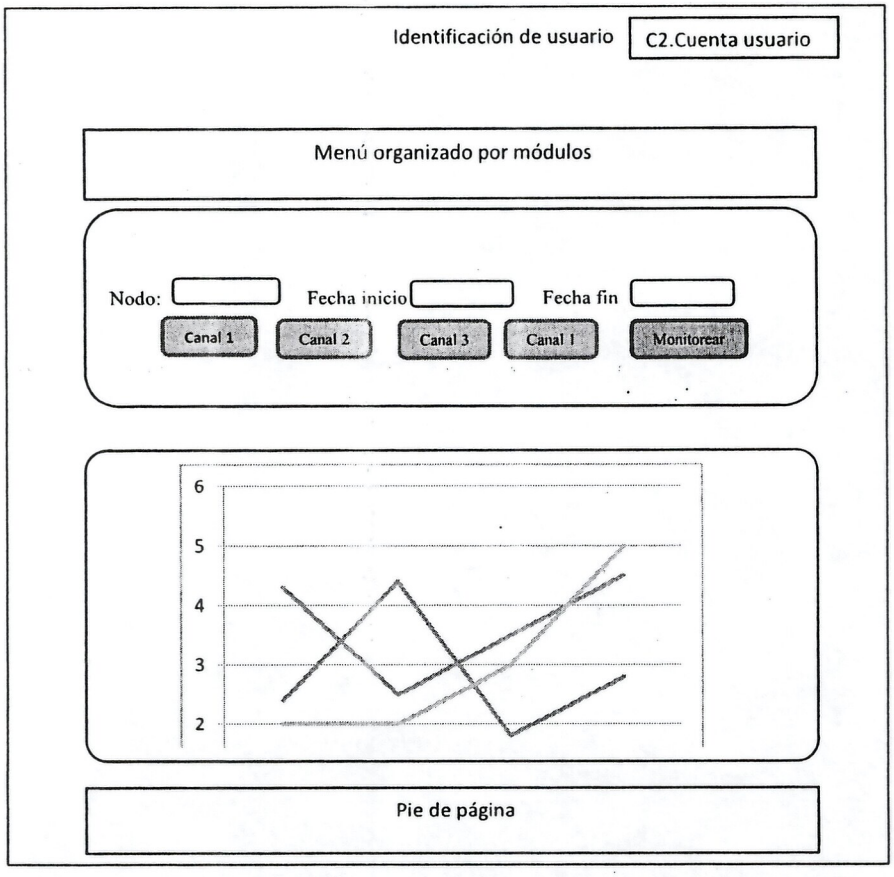



Figura 13: Diseño de la Interfaz de monitoreo; lectura de sensores.

Fuente: Propia.

### 6.1.4.2. Mostrar el estado de riego actual (M2)

En el gráfico mostrado a continuación cada barra azul representa la humedad capturada por el canal de interés, mientras que las barras rojas corresponden al umbral mínimo de humedad, definido para cada válvula.

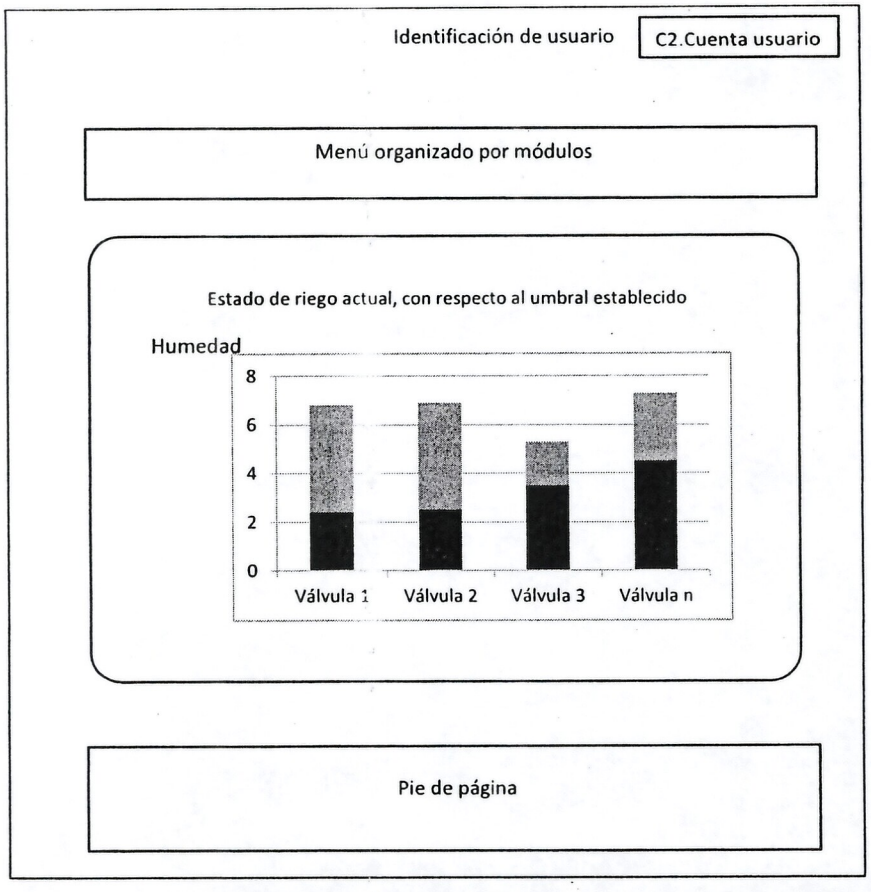



Figura 14: Diseño de la Interfaz de monitoreo para mostrar el estado de riego actual.

Fuente: Propia.

### 6.1.4.3. Mostrar los eventos de riego (M3)

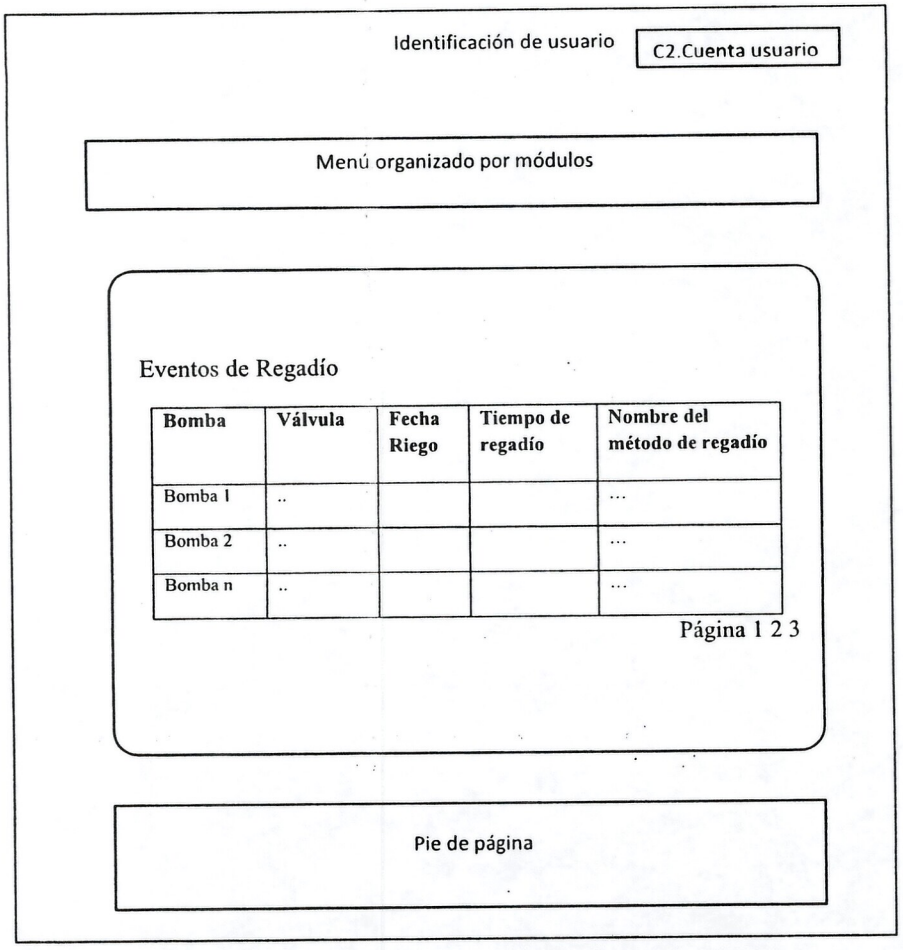



Figura 15: Diseño de la interfaz para mostrar los eventos de riego.

Fuente: Propia.

### 6.1.4.4. Mostrar la cantidad de agua usada en un periodo de tiempo (M4)

Análogo para mostrar la cantidad de energía eléctrica usada en un periodo de tiempo (M5).

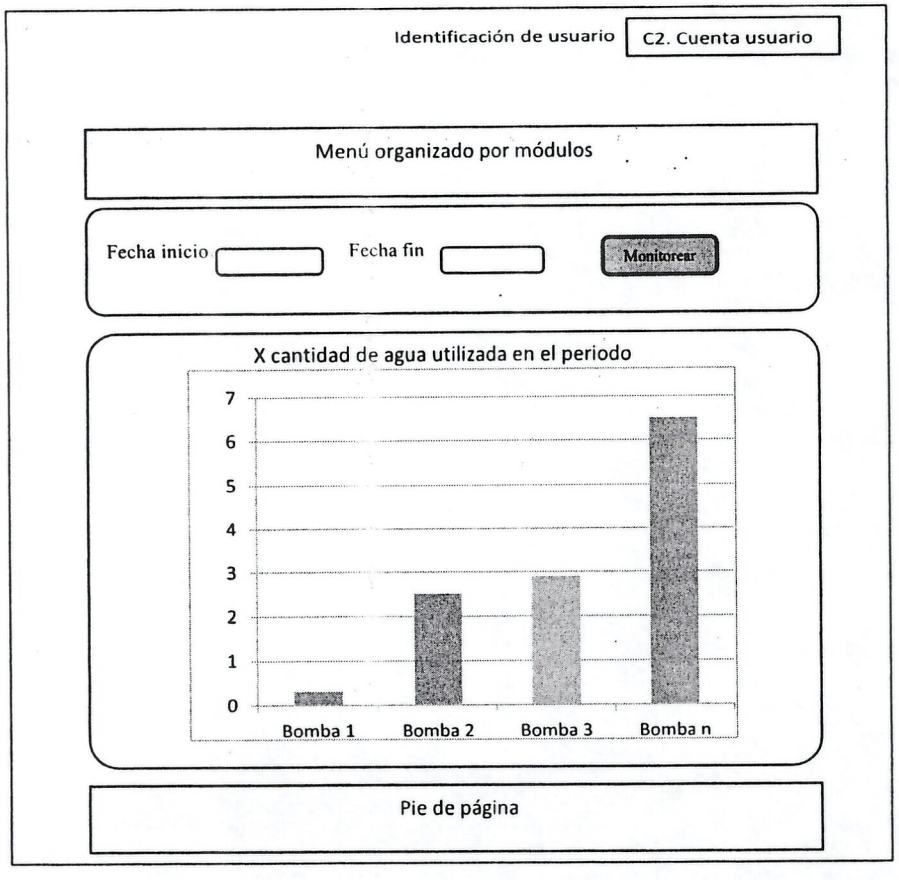



Figura 16: Diseño de la interfaz para mostrar la cantidad de agua usada, análogo para la mostrar la cantidad de energía eléctrica.

Fuente: Propia.

### 6.1.5. Diseño de la Interfaz de Riego

#### 6.1.5.1. Crear método de riego (R1)

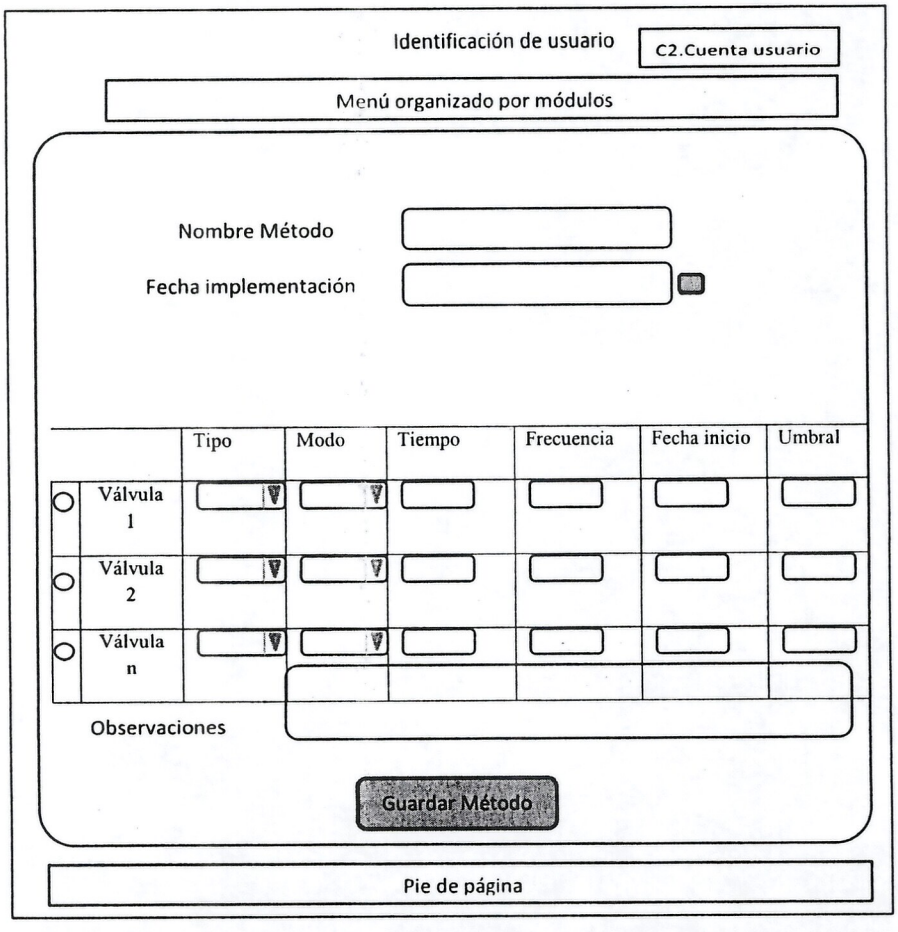



Figura 17: Diseño de la interfaz para crear un método de riego.

Fuente: Propia.

### 6.1.5.2. Mostrar Históricos de Métodos de Riego (R2)

El diseño de la ventana que emergerá en "Ver detalles" es similar a la tabla de "Método de Riego Actual" (ítem 6.1.5.3).

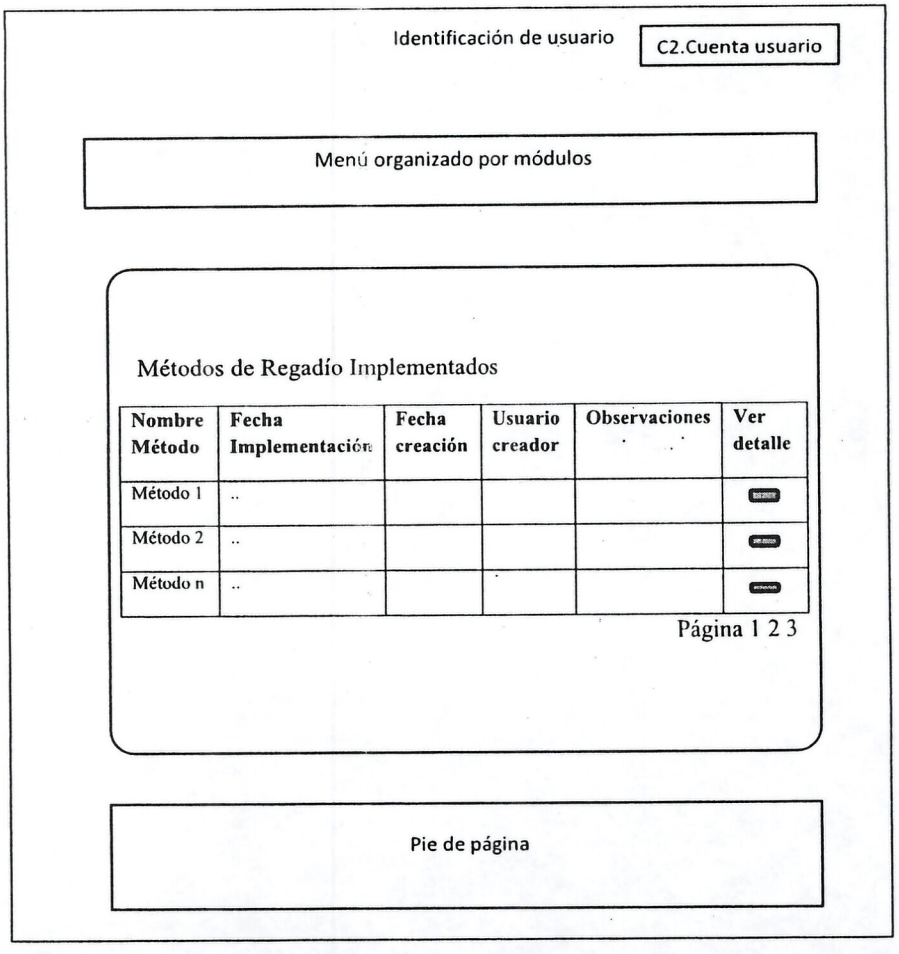



Figura 18: Diseño de la interfaz para mostrar históricos de métodos de riego.

Fuente: Propia.

### 6.1.5.3. Mostrar método de riego actual (R3)

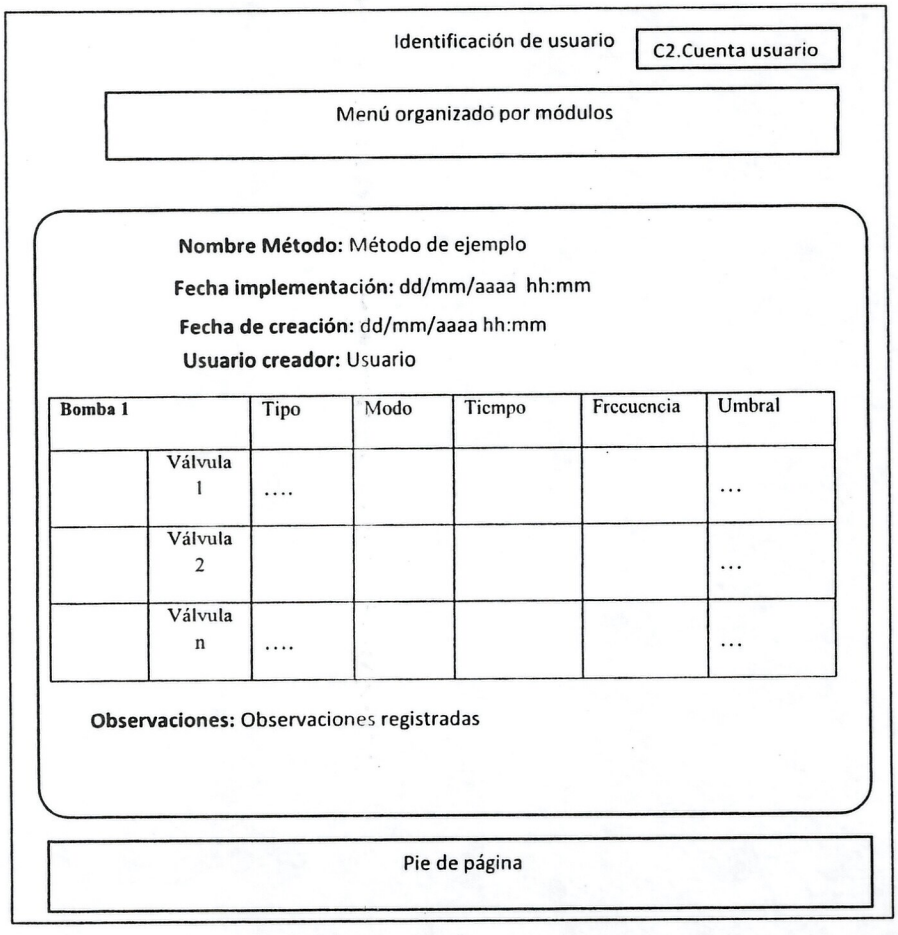



Figura 19: Diseño de la Interfaz para mostrar método de riego actual.

Fuente: Propia.

### 6.1.6. Diseño de la Interfaz de Administración de Huerto (H)

La imagen que se muestra a continuación ilustra el diseño de la interfaz para la administración del huerto, en particular muestra la administración de las bombas, análogo para las interfaces de la administración de válvulas y de nodos y canales.

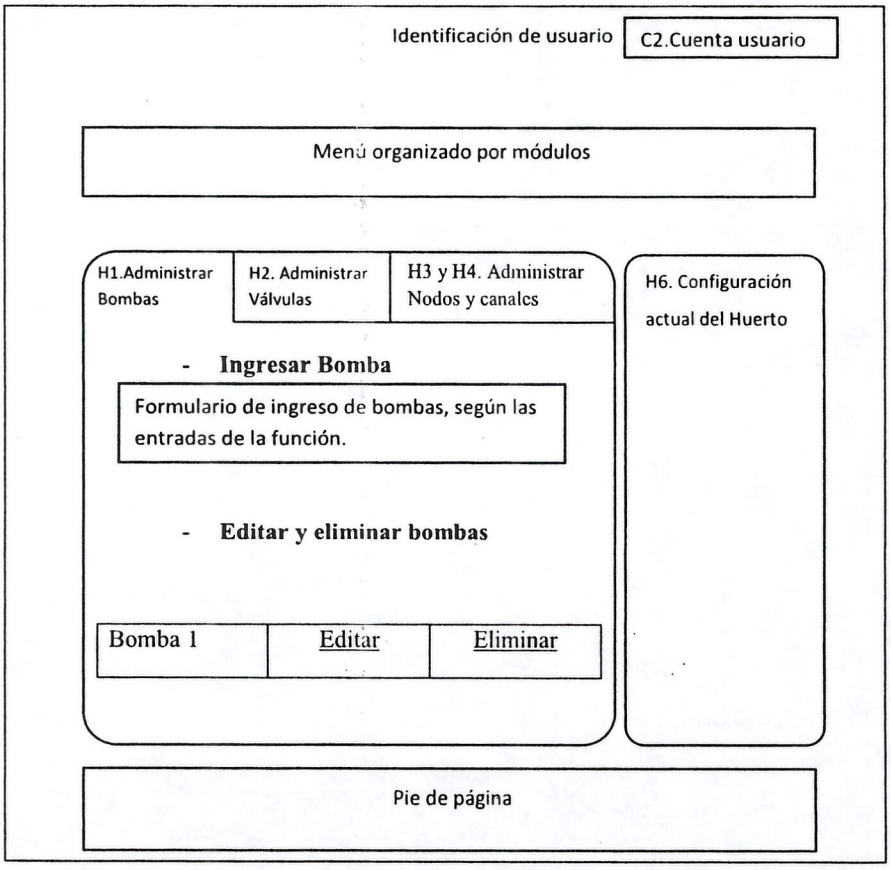



Figura 20: Diseño de la Interfaz de Administración de Huerto.

Fuente: Propia.

### 6.1.6.1 Administrar usuarios

El diseño de ingresar usuario sólo cambiará la tabla que se muestra en la siguiente imagen, por el formulario de ingreso de usuario, formulario descrito en la especificación del requisito H5.1. De la misma forma para el requisito no funcional que mostrará las actividades de los usuarios, se remplazará la tabla mostrada por una tabla con los datos descritos en la sección 3.3.2.2. (Requisito no funcional de seguridad).


Figura 21: Administrar usuarios.

Fuente: Propia.

## 6.2. *Desarrollo de la Interfaz del Sistema*

Tal como se dio a conocer en el capítulo V "análisis de tecnologías de desarrollo", se usó el framework Tiles para el manejo de la vista. El primer paso fue definir una plantilla o página "padre" con las características o Layout comunes encontrados en las distintas interfaces ya definidas.

Se definieron dos plantillas en el archivo de configuración, la primera se usó para el ingreso al sistema, dado que este no comparte muchas características con las demás interfaces.

*Definición de la plantilla de ingreso al sistema en tiles.xml*

```
<tiles-definitions>
<definition name="plantilla-login"
template="plantilla/login/plantilla.jsp">
      <put-attribute name="titulo" value="Monitor WEB | Proyecto FIA" />
      <put-attribute name="js" value="/plantilla/login/js.jsp" />
      <put-attribute name="css" value="/plantilla/login/css.jsp" />
      <put-attribute name="top" value="" />
      <put-attribute name="cuerpo" value="" />
      <put-attribute name="footer" value="/plantilla/login/footer.jsp" />
</definition>
</tiles-definitions>
```

Esta plantilla se definió de forma separada dado que conceptualmente conviene su distinción, pues no es parte de la navegación del sistema.

*Definición de la plantilla general del sistema en tiles.xml*

En todas las interfaces de navegación (excepto el ingreso al sistema) se encontraron los siguientes layout en común:

- **Top:** El contenido es común para todas las interfaces, este layout contiene el menú del usuario, donde se identifica el tipo de usuario, puede acceder a la administración de su cuenta y cerrar su sesión.
- **Menú principal:** Este contenido es variable dado que el menú de navegación cambia dependiendo del usuario que haya accedido al sistema.
- **Cuerpo:** Este layout siempre tiene información distinta, pues es el que diferencia a una interfaz de otra.
- **Footer:** Tiene contenido común para todas las interfaces.

También se definen atributos correspondientes a los siguientes archivos:

- Hojas de estilo (css).
- Archivo con funciones Javascript (js).

```
<tiles-definitions>
<definition name="plantilla-core" template="plantilla/core/plantilla.jsp">
      <put-attribute name="titulo" value="Monitor WEB | Proyecto FIA "/>
      <put-attribute name="js" value="/plantilla/core/js.jsp" />
      <put-attribute name="css" value="/plantilla/core/css.jsp" />
      <put-attribute name="top" value="/plantilla/core/top.jsp" />
      <put-attribute name="menu_principal"
      value="/plantilla/core/menu_principal.jsp" />
      <put-attribute name="cuerpo" value="/plantilla/core/default.jsp" />
      <put-attribute name="footer" value="/plantilla/core/footer.jsp" />
</definition>
</tiles-definitions>
```

Luego para que cada interfaz herede la definición anterior, se usó la propiedad **extends** para definir la página "padre" a la que pertenece. Todos los atributos definidos son heredados por la página hija. Sin embargo, para cambiar el contenido se usó la propiedad de tiles **put-attribute**, que hace referencia al layout que se está modificando con la propiedad **name**, donde su nuevo contenido se explicita con la propiedad **value**, tal como se ejemplifica a continuación.

```
<!-- Interfaz de ingreso al sistema -->

<definition name="login" extends="plantilla-login">

      <put-attribute name="titulo" value="FIA - Monitor WEB | Login" />
      <put-attribute name="cuerpo" value="/pages/login/index.jsp" />

</definition>


<!-- Interfaz de inicio del sistema -->

<definition name="menu-inicio" extends="plantilla-core">

      <put-attribute name="titulo" value="FIA - Monitor WEB | Inicio" />
      <put-attribute name="js" value="/pages/inicio/js.jsp" />
      <put-attribute name="cuerpo" value="/pages/inicio/index.jsp" />

</definition>


<!-- Interfaz Riego-nuevo -->

<definition name="menu-riego-riegonuevo" extends="plantilla-core">

      <put-attribute name="titulo" value="FIA-Monitor WEB| Nuevo Riego" />
      <put-attribute name="js" value="/pages/riego/riegonuevo_js.jsp" />
      <put-attribute name="cuerpo" value="/pages/riego/riegonuevo.jsp" />

</definition>
```

# Capítulo VII. Análisis de resultados

## *7.1. Pruebas de Validación*

Dado que el sistema fue desarrollado en un modelo de desarrollo evolutivo, las pruebas realizadas en conjunto con el usuario y sus respectivas correcciones fueron ejecutadas en repetidas ocasiones a lo largo del desarrollo del sistema y no solamente al final.

Basando el desarrollo en el estudio de usabilidad (ítem 3.2) se evaluó la correctitud de la interfaz en dos aspectos: Aspectos visuales y aspectos interactivos.

### 7.1.1. Aspectos Visuales e interactivos

- ✓ **La organización perceptual y la tarea del usuario**

Dado que la organización perceptual de la información debe estar extrapolada a cómo el usuario lleve a cabo la tarea en la actualidad sin el uso del sistema, los resultados son los siguientes.

*Monitoreo para usuario monitor*

Los agricultores están muy familiarizados con manómetros de presión, medidores de agua, entre otros, por su uso en los huertos. Por esto se pensó que una forma familiar de presentar las lecturas de los sensores en el instante podría ser a través de imágenes similares a manómetros y medidores: relojes que indicarán el estado de humedad por válvulas, agrupadas por la bomba a la que pertenecen. De esta forma se podría ver sin de una forma familiar para los usuarios el estado de humedad del huerto, donde el rango amarillo representa un nivel bajo de humedad (nivel definido por el usuario), verde un nivel óptimo y rojo un exceso de humedad. Lo anterior se muestra en la Figura 22.

**Comentario [YG22]:** Justificación figura 22.

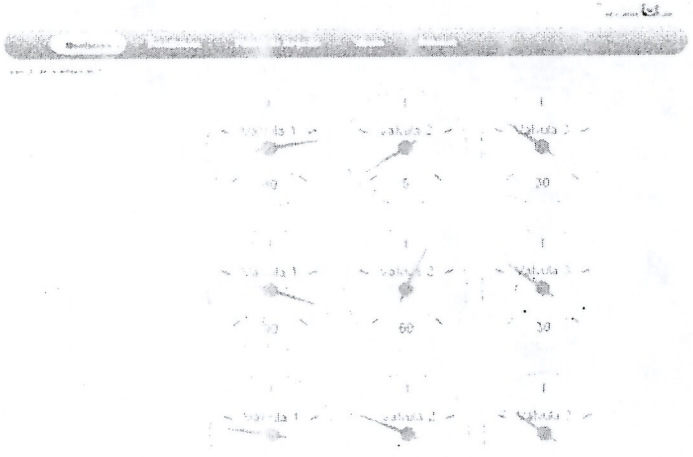


Figura 22. Interfaz de monitoreo versión 1.

Fuente: propia.

Sin embargo, luego de realizar las pruebas con los usuarios, ellos comentaron que si bien la visualización era atractiva, no cumplía con lo que ellos esperaban, pues en el huerto existen una gran cantidad de válvulas y el hecho de que cada válvula esté representada por un reloj implicaría un gran listado de relojes y no les permitiría conocer al instante si hay alguna válvula con alguna alerta de humedad.

Ante la anterior problemática se decidió usar el concepto de "semáforo", tal como se muestra a continuación, en la Figura 23.

**Comentario [YG23]:** Justificación imagen 23.

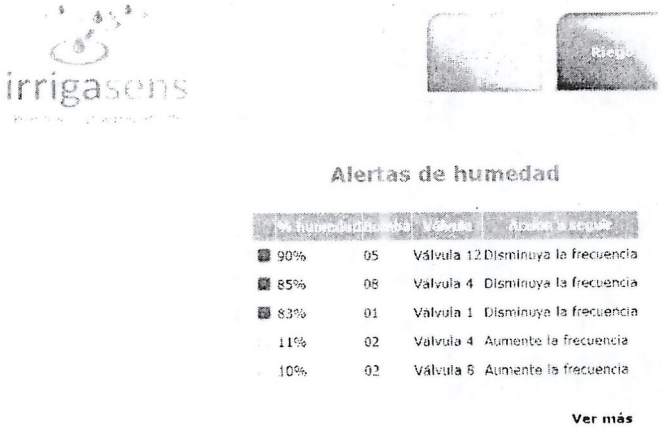



Figura 23. Interfaz de monitoreo versión 2.

Fuente: propia.

Esta interfaz de monitoreo se mantuvo como definitiva, cumpliendo con el mismo requisito anterior (M7.Alertas de humedad), pero ordenando las válvulas según su estado hídrico, donde solo se mostrarán aquellas que su nivel de humedad esté fuera del rango permitido. La principal diferencia visual con respecto a la primera versión es que se muestra mucha más información en un espacio menor, lo que permite monitorear las alertas de todo el huerto, sin ni siquiera necesitar más de una página. Además, permite mostrar una acción a seguir.

*Percepción y acceso al conocimiento: Conocimiento por medio de imágenes*

Para facilitar la navegación se usaron representaciones anexas como son las imágenes, lo que se observa claramente en el ejemplo anterior, y también en el menú de navegación, como se muestra en la Figura 24.

**Comentario [YG24]:** Justificación imagen 24.


Figura 24: Menú de navegación versión 1.

Fuente: propia.

Sin embargo, siguiendo el principio iv de los aspectos interactivos (Sección 3.2.2), a pesar que a los desarrolladores y diseñadores les pareciera atractivo, son los usuarios finales los que determinan cuando algo es fácil de usar, por lo tanto ante su petición se eliminaron las imágenes en el menú de navegación incluidas en la versión 1 (ver Figura 24). El resultado fue el que se muestra en la Figura 25.

**Comentario [YG25]:** En este punto se explica el cambio en menú de navegación, no de los relojes.

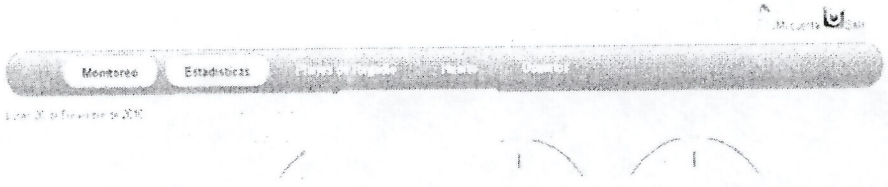



Figura 25. Menú de navegación versión 2.

Fuente: propia.

Luego, dado que el proyecto se asesoró con un diseñador gráfico, éste le dio al proyecto una imagen empresa. Se indicó el uso de una determinada gama de colores en la interfaz, de acuerdo a la imagen empresa ya definida y se realizó el último cambio.

Comentario [YG26]: Justificación figura 26.


Figura 26. Menú de navegación versión final.

Fuente: propia.

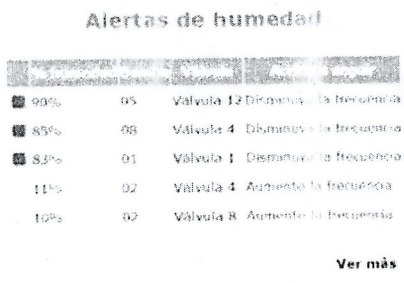
Esta última versión fue aceptada por los usuarios, justificando que la diferenciación por colores les permitía recordar con mayor facilidad la ubicación de cada módulo. Además, los colores usados se aproximan a sus modelos mentales (inciso 3.2.2), dado que por ejemplo, el azul se relaciona con agua del riego y el verde a la fertilización.

Para conocer el resultado de las demás interfaces implementadas, ver el manual de usuarios, incluido en el anexo 5 del presente documento.

### 7.1.2. Niveles de Conocimiento

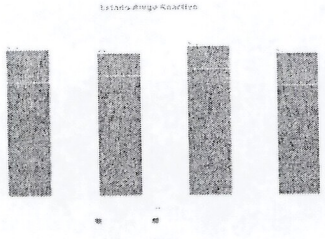
Una de las problemáticas que se abordó en el desarrollo de esta interfaz fue la diversidad de las capacidades y conocimientos de los usuarios finales (ver definición de usuarios en ítem 4.2.3). Por este motivo se tuvo que presentar el monitoreo del estado de humedad del huerto de distintas formas, con distinto procesamiento de los datos. El resultado final se muestra a continuación.

✓ *Monitoreo nivel 1*




El monitoreo nivel 1 fue diseñado pensando en el usuario monitor. Esta manera de monitorear no requiere tener un alto nivel de conocimiento agrícola, pues no se debe analizar ningún parámetro. Solo el usuario tendrá que observar si aparece un ícono rojo palpitante, lo que significará que la válvula indicada tiene un exceso de humedad; si el ícono es amarillo significará que la válvula tiene déficit de humedad. En ambos casos se mostrará una acción a seguir en la última columna de la derecha de la tabla. Al final del cuadro gris que contiene esta información, se encuentra un botón llamado "Ver más". Al presionarlo podrá acceder al monitoreo nivel 2.

✓ *Monitoreo nivel 2*

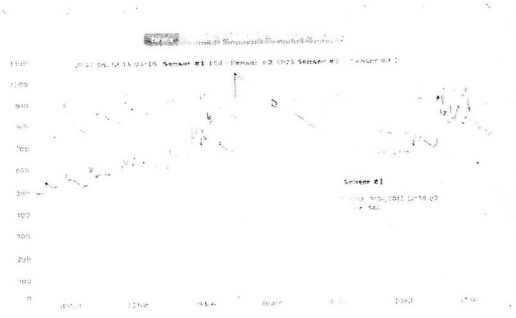


El monitoreo nivel 2 está diseñado para el administrador de huerto, que puede o no tener conocimiento agrícola, por lo que se diseñó un monitoreo que requiere un poco de análisis, ya que no tiene una acción a seguir. Aun así no se

muestra los datos como valores numéricos, sino su representación en barras gráficas.

Este monitoreo muestra gráficamente el nivel de humedad del huerto por válvula; el azul representa la humedad actual y la barra roja el nivel mínimo de humedad al que debería llegar esa válvula (umbral de regadío).

✓ *Monitoreo nivel 3*



Este monitoreo está diseñado para un usuario más experimentado, como por ejemplo el usuario controlador. El monitoreo se realiza por nodo de sensores, desplegando en un gráfico de líneas la lectura del sensor.

Estos monitoreos, si bien fueron desarrollados para ciertos usuarios, estarán disponibles para todos los usuarios del sistema, dado que al usuario controlador le es útil tener disponible un monitoreo rápido de todo el huerto (nivel 1 y 2), como también el usuario monitor puede llegar a aprender a usar los niveles 2 y 3. Por otro lado, el usuario administrador de huerto puede tener diversos conocimientos, por lo que para él es de suma importancia disponer de todas las formas de monitoreo.

Finalmente con la definición de estos distintos tipos de monitoreos, según los niveles de conocimiento de los usuarios, se logró *"Implementar una interfaz de monitoreo y planificación de riego remota, que permita desplegar información para los distintos niveles de conocimiento de usuarios"*, tal como se había planteado en el objetivo general de este proyecto.

## 7.2. *Pruebas de Verificación*

Con el objetivo de comprobar la completitud del sistema con respecto a los requisitos funcionales encontrados, se ejecutó el siguiente plan de pruebas unitarias.

Las pruebas serán presentadas por módulo. Primero se comprobó la presencia de los requisitos en el sistema y si cumple con el diseño de la interfaz definido (completitud).

### 7.2.1. Pruebas Módulo Monitoreo

Tabla 1.Pruebas Módulo Monitoreo.

| Requisito | Se implementó | | No se implementó |
|---|---|---|---|
| | Acorde al diseño | No acorde al diseño | |
| **M1.** Monitorear por nodo | ✓ | | |
| **M2.** Mostrar el estado de riego actual | ✓ | | |
| **M3.** Mostrar los eventos de riego | ✓ | | |
| **M4.** Mostrar la cantidad de agua usada en un periodo de tiempo | ✓ | | |
| **M5.** Mostrar la cantidad de energía eléctrica usada en un periodo de tiempo | ✓ | | |
| **M6.** Mostrar el estado de las baterías de los nodos | ✓ | | |
| **M7.** Mostrar alertas de humedad | ✓ | | |

### 7.2.2. Pruebas Módulo Fertilización

Tabla 2.Pruebas Módulo Fertilización.

| Requisito | Se implementó | | No se implementó |
|---|---|---|---|
| | Acorde al diseño | No acorde al diseño | |
| **F1.** Ingresar plan de fertilización | ✓ | | |
| **F2.** Editar plan de fertilización | ✓ | | |
| **F3.** Eliminar plan de fertilización | ✓ | | |
| **F4.** Mostrar fertilizaciones activas | ✓ | | |
| **F5.** Mostrar fertilizaciones pasadas | ✓ | | |

### 7.2.3. Pruebas Módulo Riego

Tabla 3. Pruebas Módulo Riego.

| Requisito | Se implementó | | No se implementó |
|---|---|---|---|
| | Acorde al diseño | No acorde al diseño | |
| **R1.** Crear método de riego | ✓ | | |
| **R2.** Mostrar históricos de métodos de riego | ✓ | | |
| **R3.** Mostrar método de riego actual | ✓ | | |

### 7.2.4. Pruebas Módulo Administración de huerto

Tabla 4. Pruebas Módulo Administración de huerto.

| Requisito | Se implementó | | No se implementó |
|---|---|---|---|
| | Acorde al diseño | No acorde al diseño | |
| **H1** Administrar bombas | ✓ | | |
| **H2** Administrar válvulas | ✓ | | |
| **H3** Administrar nodos | ✓ | | |
| **H4** Administrar canales | ✓ | | |
| **H5** Administrar cuentas de usuarios | ✓ | | |
| **H6.** Mostrar configuración actual del huerto | ✓ | | |

### 7.2.5. Administración de Cuentas

Tabla 5. Administración de Cuentas.

| Requisito | Se implementó | | No se implementó |
|---|---|---|---|
| | Acorde al diseño | No acorde al diseño | |
| **C1** Autentificar cuenta de usuario | ✓ | | |
| **C2** Editar datos de usuario | ✓ | | |

# Conclusiones

Al estudiar los procesos productivos de la agricultura tradicional se observó que la misión de la informática en esta área es entregar herramientas que permitan optimizar los tradicionales procesos productivos. En este contexto, el sistema de monitoreo y planificación de regadío permite optimizar el uso de los recursos hídricos, a través de redes de sensores con los que se obtienen datos de humedad del huerto, datos que ayudan a ejecutar planes de regadío ajustados a las necesidades reales de los predios.

**Comentario [YG27]:** Idea principal.

**Comentario [YG28]:** Justificación de lo que optimiza.

Por otro lado, al momento de realizar la especificación de requisitos se observó que el modelo de desarrollo evolutivo es bastante adecuado para proyectos cuyos requisitos funcionales no están del todo claros, pues permite refinarlos a través de las diferentes versiones hasta que se logra un sistema acorde a las necesidades del usuario.

**Comentario [YG29]:** Párrafo que concluye a partir de la ejecución del objetivo específico "Especificar requisitos (funcionales y no funcionales) del sistema de monitoreo y planificación de riego remota."

En la siguiente etapa, donde se estudió de herramientas informáticas, bibliotecas y frameworks; se concluyó que éstas facilitan enormemente el desarrollo de sistemas, pues permiten pasar más tiempo identificando requerimientos que tratando con tediosos detalles de bajo nivel, necesarios para entregar un sistema funcional.

**Comentario [YG30]:** Párrafo que concluye a partir de la ejecución del objetivo específico "Estudiar y seleccionar distintas herramientas informáticas (Open Source o Freeware) a utilizar en el desarrollo de la aplicación"

Uno de los mayores desafíos abordados en el desarrollo de la interfaz, fue la integración de los distintos tipos de usuarios, con distintos conocimientos. Esta problemática llevó a definir niveles de conocimiento para el monitoreo del huerto, así una misma funcionalidad fue desarrollada con tres niveles de procesamiento de los datos, lo que resultó útil no solamente para usuarios básicos, sino también para que los usuarios avanzados pudieran realizar un monitoreo rápido del huerto.

**Comentario [YG31]:** Párrafo que concluye a partir del cumplimiento del objetivo específico:
–Diseñar e implementar una Interfaz, organizada en niveles de conocimiento, desde las distintas perspectivas, roles y desempeños de los usuarios.

En el estudio de usabilidad y al ejecutar las pruebas del sistema, se confirmó que es el usuario quien definirá cuando un sistema es fácil de usar y no los desarrolladores, pues ellos conocen sus procesos productivos y cuál es la manera más clara y fácil de remplazar su actual metodología de trabajo por otra mayormente automatizada. Por lo tanto una

**Comentario [YG32]:** Conclusión a partir del cumplimiento del objetivo
–"Ejecutar un plan de pruebas y realizar las mejoras pertinentes."

conclusión de este trabajo es la importancia de incluir tempranamente al usuario en el desarrollo del sistema, así los cambios pueden realizarse en una etapa temprana, teniendo menor repercusión en el resto de componentes del sistema.

Finalmente, el sistema implementado, y en particular la interfaz web, permitió romper las barreras geográficas y acceder a un monitoreo y control del huerto de manera remota, lo que disminuye el tiempo de reacción ante alguna necesidad del recurso agua en el huerto.

# Referencias Bibliográficas

**Berry, D.** (01 de Octubre de 2000). IBM. Recuperado el 05 de Septiembre de 2010, de The user experience, The iceberg analogy of usability: http://www.ibm.com/developerworks/library/w-berry/.

**Burrell, J., Brooke, T., & Beckwith, R.** (2004). Vineyard Computing: Sensor Networks in Agricultural Production. *Pervasive Computing*, 41-42.

***Clima Frutal.*** (2010). Recuperado el 25 de Septiembre de 2010, de http://climafrutal.wordpress.com/el-arandano/

**Decagon.** (2010). *Decagon Devices.* Recuperado el Julio de 2010, de http://www.decagon.com/products/data-loggers-and-collectors-2/data-management-software/datatrac/.

**Fica, S.** (2010). jQuery Chile. Recuperado el 13 de 01 de 2012, de http://www.jquerychile.cl/acerca-de/

**Galli Granada, R.** (16 de Julio de 2001). Bulma. Recuperado el 7 de Julio de 2011, de http://bulma.net/body.phtml?nIdNoticia=734

**Gómez García, Á.** (2008). IMPLEMENTACIÓN DE UNA APLICACIÓN WEB UTILIZANDO FRAMEWORKS J2EE. Barcelona.

**Granollers i Saltiveri, T.**, Lorés Vidal, J., & Cañas Delgado, J. (s.f.). Diseño de Sistemas Centrados en el Usuario . Barcelona: Editorial UOC.

**Hartson, H.** (Noviembre de 1998). Human-computer interaction: Interdisciplinary roots and trends. *Journal of System and Software* , 103-118.

**IEEE Std 1471-2000**. (2007). IEEE Recommended Practice for Architectural Description of Software-Intensive Systems –Description.

**Instituto de Investigaciones Agropecuarias.** (2008). Agricultura de Precisión: Una herramienta para mejorar la rentabilidad en trigo. *INIA Tierra adentro* , 24-27.

**ISO 9241-11.** (1998). Ergonomic requirements for office work . *Guidance on usability* .

**Martín Sierra, A. J.** (2008). *Struts.* México: Alfaomega Grupo Editor.

**Ministerio de Agricultura.** (2008). FORMULARIO DE PRESENTACIÓN DE PROYECTOS TICs 2008/2009.

**O'rreilly, T.** (14 de Junio de 2010 ). *QUÉ ES WEB 2.0. PATRONES DEL DISEÑO Y MODELOS DEL NEGOCIO PARA LA SIGUIENTE GENERACIÓN DEL SOFTWARE.* Recuperado el 22 de Noviembre de 2010, de http://www.willydev.net/Willydev_old/Root/InsiteCreation/v1.0/1.1/Abril/WillyDev.Web2.0.pdf

**Sommerville, I.** (2005). Procesos del Software. En *Ingeniería de Software* (págs. 63,64). Madrid: Pearson Educación S.A.

**Sule, G. D.** (2008). Recuperado el 1 de Julio de 2011, de http://gabrielsule.blogspot.com/2008/03/top-ten-de-preguntas-para-aspnet-mvc.html

**Toni Granollers i Saltiveri, J. L.** (2005). *Diseños de Sistemas Interactivos Centrados en el Usuario.* Barcelona: Editorial UOC.

**Universidad de Talca.** (2009). *Estudio de la agricultura de precisión en Chile: Estado del Arte, ámbitos de aplicación y perspectivas.* Talca.

**Vázquez, J. J.** (15 de Mayo de 2010). Tecsisa. Recuperado el 1 de Julio de 2011, de http://blogs.tecsisa.com/articulos-tecnicos/por-que-soa/

**W3C,** (2010). Recuperado el 13 de Diciembre del 2011, de W3C: http://www.w3.org/Graphics/SVG/)

**W3C.** (09 de 01 de 2008). W3C. Recuperado el 13 de 01 de 2012, de http://www.w3c.es/divulgacion/guiasbreves/hojasestilo

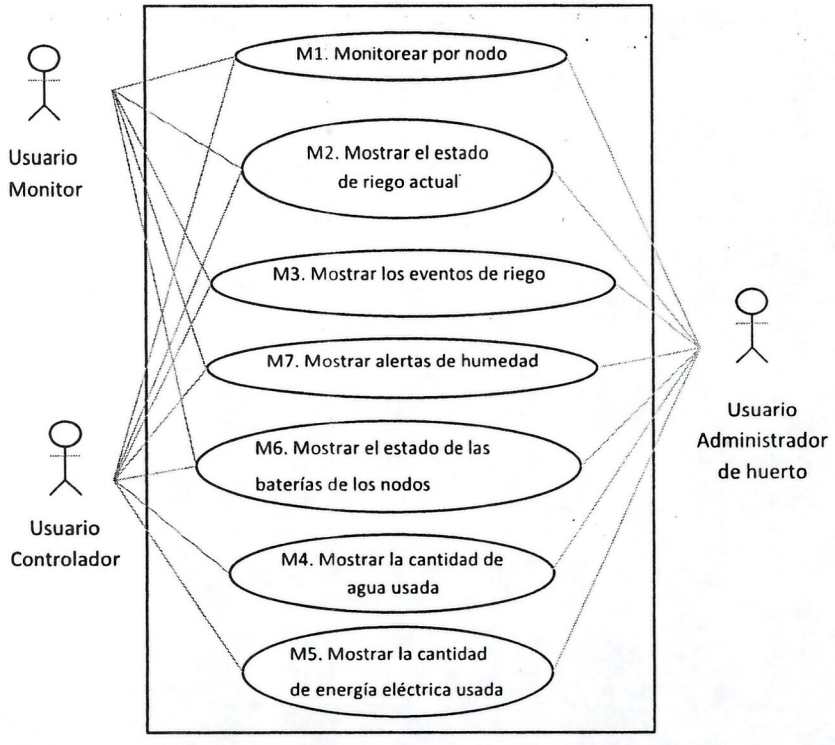
**w3schools.** (2008). w3schools. Recuperado el 2012 de 01 de 13, de http://www.w3schools.com/ajax/ajax_intro.asp

**Wikipedia. (25 de Mayo de 2010)**. Wikipedia. Recuperado el 20 de Agosto de 2010, de Agricultura de precisión: http://es.wikipedia.org/wiki/Agricultura_de_precision

# ANEXOS

## *Anexo 1: Diagrama de casos de usos para módulo de monitoreo*

Los diagramas de casos de uso que se muestras a continuación están divididos por módulos, los mismos módulos ya definidos en la especificación de requisitos (capítulo iv). A su vez cada función representada por una elipse esta explicada de forma detallada en el mismo capítulo. Para mantener una congruencia durante todo el documento, se usaron las mismas simbologías (M1, M2, M3, etc.).

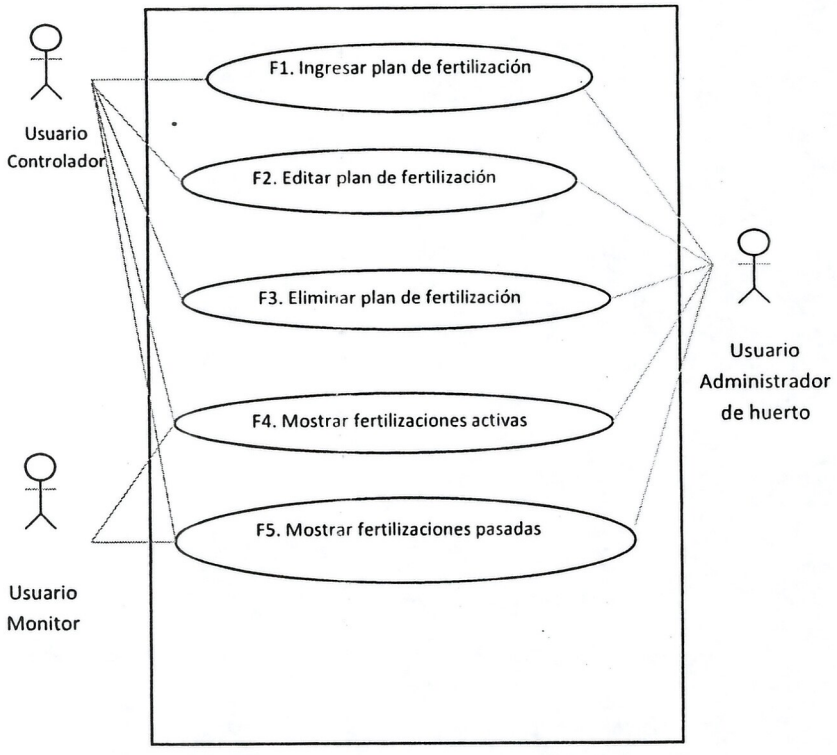



Esquema 3: Diagrama de casos de usos para módulo de monitoreo.

Fuente: propia.

## *Anexo 2: Diagrama de casos de uso para módulo de fertilización*

Este diagrama hace referencia a la interacción de los usuarios definidos en la sección 4.2.3 con las funciones especificadas la sección 4.3.1.2.

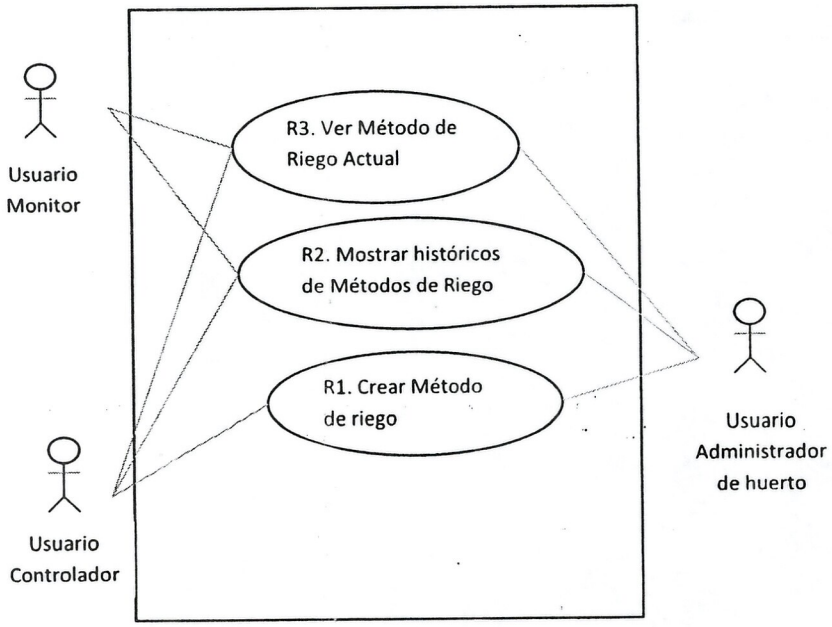



Esquema 4: Diagrama de casos de uso para módulo de fertilización.

Fuente: propia.

## *Anexo 3: Diagrama de casos de usos para módulo de riego*

Este diagrama hace referencia a la interacción de los usuarios definidos en la sección 4.2.3 con las funciones especificadas la sección 4.3.1.3.

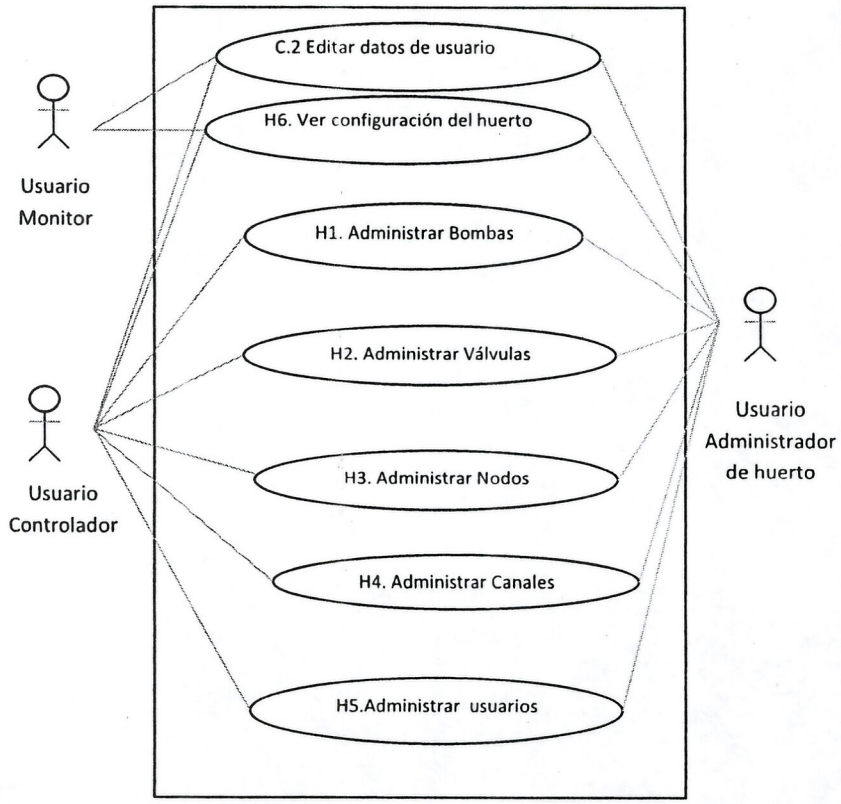



Esquema 5: Diagrama de casos de usos para módulo de riego.

Fuente: propia.

## *Anexo 4: Diagrama de casos de usos para módulo Administración de huerto y administración de cuentas.*

Este diagrama hace referencia a la interacción de los usuarios definidos en la sección 4.2.3 con las funciones especificadas las secciones 4.3.1.4 y 4.3.1.5.




Esquema 6: Diagrama de casos de usos para módulo de riego.

Fuente: propia.

## *Anexo 5: Manuales de Usuarios*

Se elaboraron tres manuales de usuarios, cada uno de ellos está dirigido a un tipo de usuario diferente de los ya definidos (usuario monitor, usuario controlador y usuario administrador de huerto).