UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 9

Sistema de Gestión de Datos Geológicos. Módulo: Inventario Nacional de Recursos de Aguas Minerales. Rol de Implementador.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autor: Reinaldo Alberto Alvarez Fernandez.

Tutor: Ing. Dianet Utria Pérez.

Ciudad de La Habana, Junio 2010.

"Año 52 de la Revolución."

DEDICATORIA

A mi abuela Marta, que sé que le hubiese gustado verme graduado.

A mis padres Osleida y Alberto que me guiaron y apoyaron en cada una de mis decisiones, aconsejándome y entregándome lo mejor de sus vidas.

A mi tía Vicenta que más que tía es una madre para mí, por estar siempre a mi lado sin importar lo que pase.

A mis hermanos que son la motivación principal de mi buen comportamiento al intentar darles un buen ejemplo a seguir.

A mi inseparable amigo Luis Orlando que más que un amigo ha sido un hermano mayor y un padre para mí.

A todos los amigos que me han acompañado todos estos años de vida.

AGRADECIMIENTOS

A dios por haberme dado la posibilidad de existir y no abandonarme en todo lo que he vivido.

A mis padres por haberme mostrado el camino a seguir, por exigirme y controlarme, verificando que siempre anduviese en buenos pasos.

A mi tía Vicenta por ser una madre incondicional.

A Luis Orlando mi gran amigo por apoyarme y guiarme cuando mis padres no podían estar todo el tiempo conmigo.

A mi novia por estar siempre a mi lado y esperarme en las madrugadas cuando regresaba de trabajar.

A mis amigos de mi Pre-Universitario.

A mis amigos de la UCI, de los grupos en los que he estado, de mi proyecto productivo (SGDG) y a los profesores del mismo que tanto molesté.

Y de forma muy especial a mi tutora Dianet porque sin ella sencillamente no podría ser ingeniero.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 9 de la Universidad de las Ciencias Informáticas y a la Oficina Nacional de Recursos Minerales a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ___ días del mes de _____ del año _____.

_____	_____
Reinaldo Alberto Alvarez Fernandez.	Ing. Dianet Utria Pérez
Autor	Tutor

DATOS DE CONTACTO

Síntesis del Tutor

Nombre: Dianet Utria Pérez.

Profesión: Ingeniero en Ciencias Informáticas.

Categoría docente: Adiestrado.

Año de graduado: 2008.

Correo: dutria@uci.cu

Breve descripción: Graduado en la Universidad de las Ciencias Informáticas, donde actualmente se desempeña en la facultad 9 como profesor de “Matemática Discreta”, una de las asignaturas del núcleo del primer año de la carrera que se estudia en dicha universidad. Diseñador de sistemas del proyecto: “Sistema para la Modelación de Fenómenos del Medio Ambiente” que pertenece al centro GEYSED de la nombrada facultad.

RESUMEN

La presente investigación tiene como principal objetivo la automatización de parte de las funciones que se llevan a cabo en la Oficina Nacional de Recursos Minerales perteneciente al Ministerio de la Industria Básica. Como resultado de la misma se espera obtener un software que automatice parte de las funciones correspondientes a la gestión de datos geológicos. Las características que debe poseer la aplicación se encuentran previamente definidas, de ahí que la presente investigación se enfoque en la construcción de dicha aplicación, para ello se caracterizan los lenguajes de programación, herramientas, framework y gestor de bases de datos usados para su implementación.

Además se exponen los resultados obtenidos a partir de la realización de las actividades correspondientes al rol de implementador que se desarrollan a favor del módulo Inventario Nacional de Recursos de Aguas Minerales que forma parte del proyecto “Sistema de Gestión de Datos Geológicos”; como son: la valoración del diseño propuesto por el analista, el estándar de código seleccionado con el objetivo de mantener un código legible; así como los diseños de casos de pruebas necesarios para comprobar el correcto funcionamiento de la aplicación.

PALABRAS CLAVES:
Implementador, Sistema de Gestión de Datos Geológicos, Inventario Nacional de Recursos de Aguas Minerales.

ÍNDICE

ÍNDICE DE TABLAS Y FIGURAS

INTRODUCCIÓN

El uso de las Tecnologías de la Información y la Comunicación (TIC) se acrecienta y se extiende sin detenerse, sobre todo en los países desarrollados. Esto puede provocar no sólo un incremento considerable de la brecha digital, sino también social. Tal es su repercusión que la era actual es llamada la era de la información. De ahí la importancia de no negarse al desarrollo y uso de estas tecnologías, sobre todo en los países subdesarrollados que deben trazarse estrategias para no quedarse detrás en esta área y evitar así que se acentúe la brecha en ambos sentidos.

Cuba ha reconocido la importancia que tienen las TIC y actualmente se encuentra en un proceso de digitalización y automatización de cuanta información y proceso le sea posible en cada uno de los sectores. Uno de estos sectores lo constituyen las Ciencias Geológicas de Cuba el cual es de vital importancia para el desarrollo de la economía del país.

La entidad encargada actualmente de garantizar el aprovechamiento racional de los recursos minerales del país y ejercer con eficiencia, rigor técnico y responsabilidad el control estatal sobre las actividades de la geología, minería y petróleo es la Oficina Nacional de Recursos Minerales (ONRM). La misma desarrolla desde 2006 el Programa de Informatización del Conocimiento Geológico cuyo objetivo general es “Implementar herramientas informáticas robustas, debidamente protegidas y aseguradas que sigan estándares informáticos, de calidad, legales y normativos de las geociencias para la conservación y administración del conocimiento geológico.”

Como parte de la industria cubana del software la Universidad de las Ciencias Informáticas (UCI) juega un papel esencial para llegar a alcanzar este objetivo, pues como universidad de nuevo tipo, intenta convertirse en líder nacional de este campo. En el marco de este programa de informatización se crea en la facultad 9 de dicha universidad un nuevo proyecto; “Sistema de Gestión de Datos Geológicos (SGDG)” el cual forma parte del Departamento Geoinformática. Este proyecto tiene como meta proveer a la ONRM de un sistema que le permita llevar a cabo las funciones relacionadas con la gestión de datos geológicos.

Una de las funciones fundamentales de la ONRM es el mantenimiento de las estadísticas de la información geológica almacenada en ella. Producto de estas estadísticas es que se realiza anualmente el Inventario Nacional de Recursos y Reservas, con el objetivo de que la dirección de la Revolución posea toda la información sobre el estado de las reservas y recursos minerales en el país, controlando y garantizando el uso de las mismas. El agua mineral al ser considerada un recurso mineral no está exenta de esto.

El control de este recurso mineral es un proceso complicado. Esto se debe fundamentalmente a:

- El gran volumen de información que se maneja, emitida en diferentes formatos, lo cual conlleva a que la información llegue incompleta y no cumpla todas las exigencias requeridas, lo que a su vez trae consigo todo un proceso tedioso que consiste en devolver la información a los Concesionarios que no son más que las empresas o entidades que poseen permisos legales para hacer uso del agua mineral.
- Los Concesionarios deben entregar la información a la oficina de forma presencial, lo cual provoca pérdida de tiempo en cuanto a la inmediatez de la información así como un engorroso proceso tanto para los Concesionarios como para la ONRM, teniendo en cuenta que estos pueden radicar en cualquier provincia del país.
- Se debe realizar un trabajo exhaustivo para poder estudiar y analizar la información almacenada, ya sea para aprobar dicha información o para conformar los diferentes inventarios que son necesarios.
- La información es poco accesible ya que cuando alguna persona desea consultarla, deberá dirigirse a la ONRM.

Las dificultades antes mencionadas conllevaron a que surgiera como parte del proyecto SGDG el módulo Inventario Nacional de Recursos de Aguas Minerales con el objetivo de automatizar estos procesos y así disminuir en gran medida estas dificultades. Para el desarrollo del SGDG se adoptó una de las metodologías de desarrollo de software de acorde al tamaño del proyecto, de forma tal que cada proceso, etapa, responsabilidad y actividad estén bien definidos de acuerdo a las características del mismo.
Como parte del ciclo de desarrollo de software, en correspondencia con la metodología utilizada, este módulo cuenta con el diseño de una aplicación informática para la realización del Inventario Nacional de

Recursos de Aguas Minerales, pero el mismo no ha sido desarrollado hasta la fecha. El hecho de que este diseño no sea funcional provoca que el módulo Inventario Nacional de Recursos de Aguas Minerales esté incompleto, por lo que aún la ONRM no posee una herramienta informática para llevar a cabo el inventario. Esto trae consigo además que el producto SGDG no pueda ser liberado en tiempo y forma, lo que trae como consecuencia la insatisfacción del cliente.

Como resultado del análisis de la situación antes expuesta se identifica como **problema a resolver** el siguiente: ¿Cómo lograr la funcionalidad del diseño propuesto del módulo Inventario Nacional de Recursos de Aguas Minerales que forma parte del Sistema de Gestión de Datos Geológicos?

Para ello la investigación centra su **objeto de estudio** en el proceso de implementación de sistemas de gestión de información y el **campo de acción** en la implementación del módulo Inventario Nacional de Recursos de Aguas Minerales que forma parte del Sistema de Gestión de Datos Geológicos.
El **objetivo general** que se persigue es elaborar la documentación técnica de la implementación del diseño propuesto correspondiente al módulo Inventario Nacional de Recursos de Aguas Minerales que forma parte del Sistema de Gestión de Datos Geológicos.

Para alcanzar el objetivo propuesto y teniendo como base el problema a resolver se formuló la siguiente **idea a defender "**Con la elaboración de la documentación técnica correspondiente a la implementación del diseño propuesto, se obtendrá la funcionalidad del módulo Inventario Nacional de Recursos de Aguas Minerales que forma parte del Sistema de Gestión de Datos Geológicos."

Con el fin de garantizar el cumplimiento del objetivo planteado se plantean las siguientes **tareas de la investigación**:

- Desarrollar habilidades en el uso de la herramienta Case Visual Paradigm.
- Desarrollar habilidades en el uso del lenguaje de programación PHP, del Framework Symfony y del gestor de Bases de Datos PostgreSQL.
- Argumentar el uso de PHP, PostgreSQL, Symfony.
- Caracterizar el rol Implementador según la metodología seleccionada.
- Caracterizar el diseño propuesto.

- Seleccionar y aplicar un estándar de codificación adecuado.
- Implementar los Casos de Uso propuestos por el analista.
- Documentar los principales algoritmos codificados.
- Validar la Solución Propuesta.

Para llevar a cabo la presente investigación se hace uso de los siguientes **métodos de la investigación científica**:

Métodos teóricos:

- Histórico-lógico: Este método se utiliza en el estudio del avance y las tendencias de las herramientas y tecnologías vinculadas con los clientes y servidores Web a través del paso de los años. Su uso se ve reflejado directamente en el estudio de lenguajes de programación en el lado del cliente y el servidor, el rendimiento y aprovechamiento de servidores Web y su integración con los gestores de bases de datos más usados por su robustez.

- Analítico-sintético: Su uso está enmarcado en el análisis, estudio y resumen de los temas de mayor importancia en el campo la gestión de la información en las aplicaciones Web. Este método es de gran utilidad en la obtención de información relevante acerca de las formas más eficientes de almacenar y procesar información en la Web actual.

- Modelación: Constituye un factor clave para percibir correctamente el modelo de diseño de la aplicación.

Método empírico:

- Observación: Se utiliza para obtener una visión satisfactoria acerca del proceso que se va a informatizar, así como de las aplicaciones de gestión de información existentes, incluidas las desarrolladas por el SGDG.

CAPÍTULO 1: Fundamentación Teórica

1.1 Introducción

Este capítulo se dedica al estudio y fundamentación de las diferentes técnicas y herramientas de la ingeniería de software que se utilizarán para el desarrollo de la aplicación informática. Se fundamentan los lenguajes de programación que se utilizarán así como los paradigmas usados para la implementación de los mismos. De igual forma se describen las tecnologías y Framework necesarios para el desarrollo de la aplicación, así como el Sistema Gestor de Bases de Datos que se usará para lograr la persistencia de los datos. Todo lo anterior se encuentra previamente definido en la arquitectura del proyecto Sistema de Gestión de Datos Geológicos al que pertenece este módulo.

1.2 Paradigmas de la Programación

Programar se puede definir en el mundo de las computadoras como la actividad mediante la cual se escribe un conjunto finito y ordenado de instrucciones que le indican a un computador, cómo lograr un resultado determinado. Utilizando un conjunto de símbolos junto a un conjunto de reglas para combinar dichos símbolos que se usan para expresar programas. Constan de un léxico, una sintaxis y una semántica. **(Lobos, 2006)**

A la hora de programar se usan distintos paradigmas:

- Programación Lineal.
- Programación Estructurada.
- Programación Modular.
- Programación Orientada a Objetos.
- Programación Genérica.

Programación Lineal

Normalmente, cualquier programador empieza escribiendo programas pequeños y simples. En este caso el "programa principal" consiste en una secuencia de códigos que modifican datos globales a través de todo el programa. Lo anterior se ilustra en la figura 1.1.

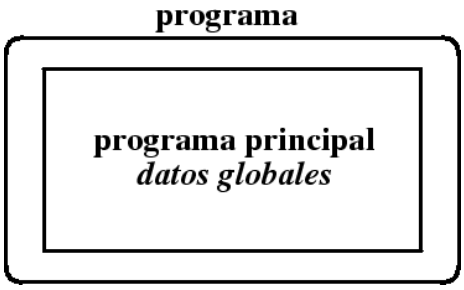

Figura 1. 1: Paradigma de programación lineal.

Es bien conocido que esta forma de programación tiene varias desventajas una vez que el programa es lo suficientemente grande. Por ejemplo, si la misma secuencia de declaraciones se necesita en lugares diferentes del programa, entonces esta secuencia debe ser copiada en cada lugar donde haga falta. Lo anterior generó la idea de extraer secuencias y ponerlas en algún "lugar" desde donde sea posible ejecutarlas para luego regresar al programa principal. El "lugar" se conoce como subrutina, procedimiento o función, dependiendo del lenguaje, y es la base de la programación estructurada.

Programación Estructurada

En la programación estructurada se pueden agrupar secuencias de declaraciones e instrucciones en un mismo lugar, las que después pueden ser usadas por el programa principal desde cualquier punto. Se dice que se hace una llamada para invocar al procedimiento. Después que la secuencia es procesada, el control de flujo regresa al programa principal justo después de donde se hizo la llamada, véase la figura 1.2.

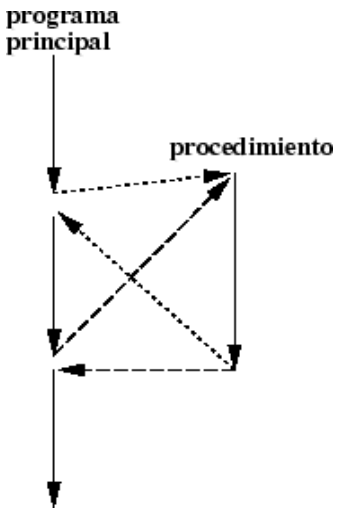

Figura 1. 2: Control de flujo.

El paradigma original de este tipo de programación consiste en:

- Buscar los procedimientos que se desean.
- Usar los mejores algoritmos existentes.

La inclusión de procedimientos hace a los programas estructurados y con menos posibilidad de errores. Por ejemplo, si un procedimiento es correcto, cada vez que sea ejecutado producirá, en principio, resultados correctos. Por el contrario, en caso de que el procedimiento contenga errores, se puede agilizar la búsqueda de estos últimos y repararlos rápidamente. De esta manera, el control de flujo de datos puede ser ilustrado como una gráfica jerárquica como se muestra en la figura 1.3.

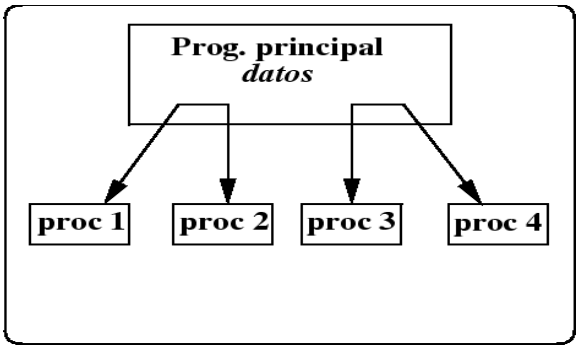

Figura 1. 3: Control de flujo en la programación estructurada.

Programación Modular

La programación modular permite agrupar procedimientos, que tienen una funcionalidad común, en módulos separados. Por lo tanto, un programa ya no consiste de una sola parte. El programa ahora se divide en varias partes más pequeñas que interactúan a través de interfaces y llamadas a procedimientos, véase figura 1.4.

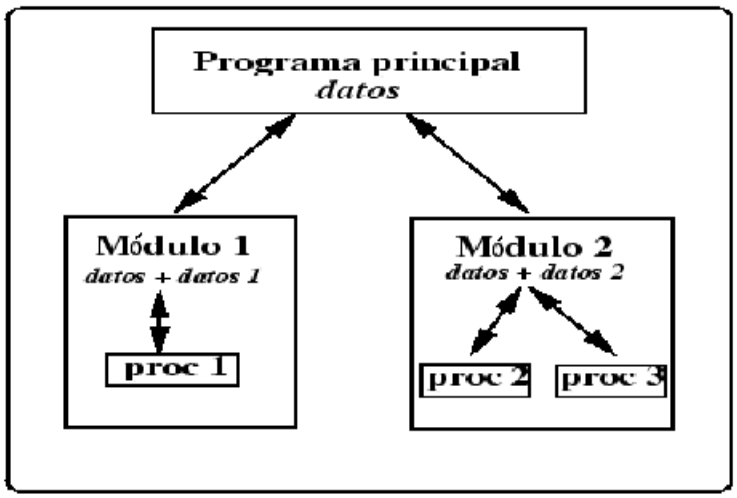

Figura 1. 4: Paradigma de la programación modular.

Cada módulo puede contener sus propios datos. Esto permite que cada módulo maneje un estado interno el cual es modificado por las llamadas a sus procedimientos. Dado que los módulos agrupan operaciones que son, en algún sentido comunes, entonces se dice que la programación modular es orientada a las operaciones y no a los datos. Se dice que las operaciones definidas en un módulo especifican los datos que serán usados.

El paradigma de la programación modular consiste en:

- Buscar los módulos que se quieran.
- Particionar el programa de tal manera que los datos se escondan dentro de los módulos.

Programación Orientada a Objetos (POO)

En la POO, la estructura es organizada por los datos. Primero se seleccionan las representaciones de los datos que mejor cumplen con los requerimientos y luego se decide qué hacer con ellas, que es justo lo

contrario a la programación modular. La abstracción de datos es fundamental para un buen diseño. En contraste con otras técnicas, ahora se va a interactuar con objetos, cada uno manteniendo su propio estado, ver figura 1.5.

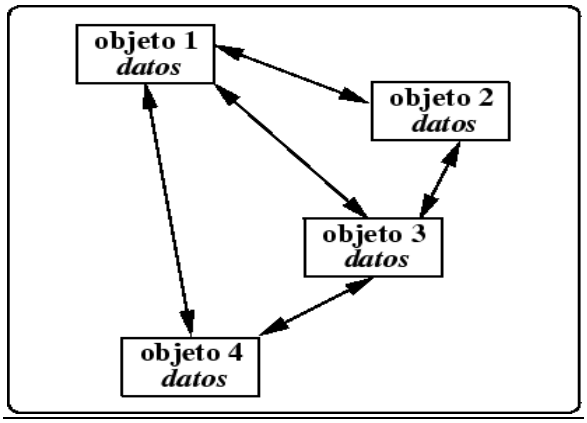

Figura 1. 5: Paradigma de programación orientada a objetos.

En la POO existen más conceptos y técnicas que ayudan al programador a ordenar sus ideas y, consecuentemente, a construir softwares más manejables y con un tiempo de vida mayor que usando otras técnicas.

El paradigma de la programación orientada a objetos es:

- Buscar las clases que se desean.
- Proveer de un conjunto completo de operaciones para cada clase.
- Especializar usando herencia.

Programación Genérica

Las herramientas ofrecidas por la POO parecen resolver muchos de los problemas que se presentan con las otras técnicas. Sin embargo, en algunos casos no es suficiente con esas herramientas y se tiene que recurrir a otras formas de programación.

Supóngase por ejemplo, que se desea hacer el código de la pila lo más general posible, es decir, que se puedan generar pilas de int, float, char, etc. y de tipos definidos por el usuario.

Como se mencionó antes, una pila es usada en muchas áreas, por lo que es posible pensarla como un concepto general, independiente de la noción del tipo que maneje. En general, si un algoritmo puede ser expresado independientemente de los tipos de datos, este debería ser codificado de esta manera para que sea lo más general posible. Esta es la idea detrás de la programación genérica.

El paradigma de la programación genérica es:

- Buscar los algoritmos que se desean.
- Parametrizarlos de tal manera que trabajen bien para una gran variedad de tipos y estructuras de datos.

Solamente algunos lenguajes permiten hacer programación genérica, C++ es uno de ellos (a través del uso de templates). La biblioteca estándar (STL) de C++ está construida bajo esta metodología.

Las plantillas ("Templates"), también denominadas tipos parametrizados, son un mecanismo que permite que un tipo pueda ser utilizado como parámetro en la definición de una clase o una función. **(Cruz, 2000)**

1.3 Lenguajes de Programación

Se puede definir la programación como la acción de proporcionar a un ordenador un conjunto de instrucciones sobre cómo debe trabajar con los datos proporcionados con el objetivo de darle solución a determinada cuestión.

Toda máquina incluyendo las computadoras, necesita un lenguaje para poder procesar las instrucciones que se le dan y de esta forma poder controlar su comportamiento. Ese lenguaje se conoce como lenguaje de programación, el cual permite establecer una relación con las computadoras. El mismo está conformado por reglas sintácticas y semánticas, por lo que consta de un léxico, una sintaxis y una semántica, con las que el programador trabajará a la hora de crear los programas y subprogramas.
Lo que diferencia al lenguaje de programación de otros como el humano, es que permite hacer las especificaciones de forma concisa, logrando que una instrucción sea interpretada siempre de la misma forma sin importar quien la haya escrito.

Los lenguajes de programación pueden clasificarse siguiendo varios criterios:

Según su nivel de abstracción:

- Lenguaje de bajo nivel (es el código fuente de la máquina, es decir el que la máquina puede interpretar).
- Lenguaje de nivel medio (un término entre el lenguaje de la máquina y el lenguaje natural).
- Lenguaje de alto nivel (los que están compuestos por elementos del lenguaje natural, es decir el humano, especialmente el inglés).

Según la forma de ejecución se clasifican en:

- Compilados. Usan un compilador (programas que permiten traducir un programa del lenguaje natural al lenguaje de bajo nivel).
- Interpretados. Usan un intérprete (los que sólo hacen la traducción de los datos que se van a utilizar en ese momento y no los guarda para usar posteriormente).

Entre los lenguajes de programación más conocidos y utilizados en la actualidad se pueden citar: Delphi, Visual Basic, Pascal, C++, Java, Python, etc.

1.3.1 HTML

El Lenguaje Generalizado Estándar para el Formato de Documentos (SGML) es un sistema encargado de definir lenguajes para dar formato a documentos (*markup languages*). Se utiliza un código de formato (*markup*) en sus documentos para representar información estructural, presentable y semántica junto con el contenido. El HTML es un ejemplo de lenguaje de formato de documentos.

Las siglas HTML, en inglés (*HyperText Markup Language*), significan en español lenguaje de marcado de hipertexto, es un estándar utilizado en todo el mundo cuyas normas define un organismo sin ánimo de lucro llamado W3C (*World Wide Web Consortium*).

HTML es un lenguaje de composición de documentos que se encarga de definir una sintaxis e insertar instrucciones especiales indicándole al navegador como desplegar el contenido en el documento, incluyendo texto, imágenes y otros medios soportados. También brinda la posibilidad de lograr que un

documento sea interactivo a través de ligas especiales de hipertexto, las cuales conectan distintos documentos, ya sea en su computadora o en otras, así corno otros recursos de Internet, como FTP.

HTML es el lenguaje con el que se definen las páginas Web. Básicamente se trata de un conjunto de etiquetas que sirven para definir el texto y otros elementos que compondrán una página Web. HTML es lo que se utiliza para crear todas las páginas Web de Internet, más concretamente, HTML es el lenguaje con el que se "escriben" las páginas Web. Los desarrolladores de navegadores dependen del estándar de HTML para programar el software que da formato y despliega documentos de HTML comunes. Los creadores de páginas utilizan el estándar para asegurarse de que sus documentos de HTML son correctos y efectivos. **(Musciano, et al., 1999)**

1.3.2 XHTML

El lenguaje extensible de marcado de hipertexto (XHTML) es bastante parecido al lenguaje HTML. De hecho, XHTML se puede definir como una adaptación de HTML al Lenguaje de Marcado Extensible (XML). XML tiene a su alrededor otras tecnologías que lo complementan, su principal ventaja es que permite compartir información entre aplicaciones.

Básicamente, HTML es descendiente directo del lenguaje SGML, mientras que XHTML lo es del XML, que a su vez, también es descendiente de SGML.

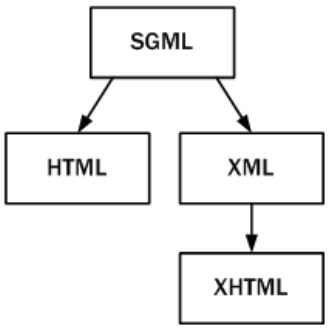

Figura 1. 6: Esquema de la evolución de HTML y XHTML

La diferencia más notable respecto a HTML es que se caracteriza por tener una sintaxis más estricta. Su aparición está dada por la necesidad de que se interprete correctamente cada información, sin tener

dependencia del dispositivo con que se accede a ella. Un sitio Web o un documento XHTML funcionan con todos los navegadores y al mismo tiempo en todos los dispositivos portátiles con soporte XML, como Laptops, TabletPcs, PDAs, Teléfonos WAP, Web Pads, Web TV entre otros.

1.3.3 CSS

CSS son las siglas en inglés de *Cascading Style Sheetsc,* las hojas de estilo en cascada es un lenguaje creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. Es la mejor forma de separar los contenidos y su presentación. El uso de CSS se hace imprescindible cuando se desea crear páginas Web complejas. **(Pérez, 2009)**

Mientras que el lenguaje HTML/XHTML se utiliza para marcar los contenidos, o sea, para definir que es un párrafo, una lista de elementos; el lenguaje CSS se utiliza para definir el aspecto de todos los contenidos, es decir, el color, tamaño y tipo de letra de los párrafos de texto, la separación entre los encabezados y contenidos, la tabulación con la que se muestran los elementos de una lista, etc.

La separación de los contenidos y su presentación provee numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados "documentos semánticos"). De igual forma, mejora el acceso al documento, disminuye la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes, aumentando en gran medida la reutilización del código.

Ventajas:

- Mejora el posicionamiento de los elementos Web.
- Permite que el código dedicado a la semántica de un documento y el encargado de darle formato al mismo estén separados por lo que el código es más claro a los ojos de los robots de búsqueda.
- Al usar CSS se logra que los documentos Web tengan un menor peso, lo cual es positivo para los robots de búsqueda y para los usuarios finales.
- Le provee al diseño gran flexibilidad ya que se puede cambiar en cualquier momento alguna parte o la totalidad del diseño de las páginas con sólo modificar las hojas de estilo, sin que esto implique modificar el contenido.

- Posee buena compatibilidad pues mantiene las reglas establecidas en las versiones anteriores.
- Permite diferenciar entre los estilos para imprimir / visualizar en pantalla.

1.3.4 DHTML

HTML dinámico (DHTML) es lo que hace posible crear una página Web sin las limitaciones del HTML, es muy abarcador y engloba muchas técnicas que se pueden realizar con multitud de lenguajes de programación y programas distintos.

DHTML de cliente

El DHTML de cliente es el que se desarrolla en el ámbito de una página Web, cuando la página se está viendo en la pantalla de los usuarios, es decir, en los navegadores. Este se utiliza para lograr algún efecto o interactividad en la página se posee al navegador como recurso, por eso se llama de cliente. La programación en el cliente tiene múltiples usos, como son efectos diversos en las páginas, sonidos, videos, menús interactivos, control y respuesta a las acciones de un usuario en la página, control sobre los formularios, etc. Para hacer muchas de estas cosas se pueden emplear varios lenguajes de programación como JavaScript y VBScript, también permite incluir programas como Flash.

1.3.4.1 JAVASCRIPT

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas Web dinámicas. Una página Web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. JavaScript es un lenguaje de programación interpretado, por lo que no se debe compilar para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

Javascript es un lenguaje con muchas posibilidades, permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones, estructuras de datos complejas, etc. Toda esta potencia de Javascript se pone a disposición del programador, que se convierte en el verdadero dueño y controlador de cada cosa que ocurre en la página. **(Alvarez, 2005)**

Ventajas:

- Es un lenguaje interpretado por lo que no implica un tiempo de compilación.
- Es multiplataforma, siempre que exista un navegador para JavaScript en las plataformas.
- Permite hacer casi todo tipo de validaciones, proporcionando una interfaz más interactiva sin necesidad de utilizar código del interpretado del lado del servidor Web.

1.3.5 PHP

PHP es un acrónimo de (*Personal Home Page, Procesador de Hipertexto*), Pre-procesador de Hipertexto. Es un lenguaje de script incrustado dentro del HTML que se ejecuta del lado del servidor, el mismo actualmente es uno de los más usados en la creación de Web dinámicas. Su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas de sí mismo.

Este lenguaje se caracteriza por ser potente y a la vez simple ya que con poco esfuerzo permite embeber sus pequeños fragmentos de código dentro de la página HTML y realizar determinadas acciones. Por otra parte, con el objetivo manipular y gestionar información almacenada en bases de datos, PHP ofrece una gran variedad de funciones fáciles de usar. Una característica notable es el hecho de que permiten a la mayoría de los programadores crear aplicaciones complejas con poco ezfuerzo en el aprendizaje. También permite interactuar con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones.

Básicamente su interacción es la siguiente, el cliente hace una petición al servidor con el objetivo de que le envíe una página Web, el servidor ejecuta el intérprete de PHP, éste procesa el script solicitado que generará el contenido de manera dinámica. El resultado es enviado por el intérprete al servidor, y el servidor se lo envía al cliente.

Ventajas:

- Es un lenguaje multiplataforma.
- Brinda la posibilidad de conectarse con la mayoría de los manejadores de base de datos que se utilizan en la actualidad.

- Permite la lectura y manipulación de datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
- Provee la capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su página oficial.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite el uso de técnicas de Programación Orientada a Objetos.

1.4 Plataformas, Herramientas de desarrollo y Frameworks

1.4.1 Servidor Apache

Apache es un servidor Web HTTP de código abierto y multiplataforma. Este servidor Web está hecho con robustez y estabilidad por lo que tiene en estos momentos gran popularidad entre las empresas y desarrolladores Web, en lo que también ha influido la posibilidad de ser configurable.

El servidor Apache es un software que está estructurado en módulos. La configuración de cada módulo se hace mediante la configuración de las directivas que están contenidas dentro del módulo. Este servidor está estructurado en módulos, los mismos se pueden clasificar en tres categorías:

- **Módulos Base:** Módulo con las funciones básicas del Apache.
- **Módulos Multiproceso:** Son los responsables de la unión con los puertos de la máquina, aceptando las peticiones y enviando a los hijos a atender a las peticiones.
- **Módulos Adicionales:** Cualquier otro módulo que le añada una funcionalidad al servidor.

Por defecto se encuentran incluidos los módulos *base y multiproceso:*

- Autentificación: La identificación positiva de una entidad de red tal como un servidor, un cliente, o un usuario.
- Control de Acceso: La restricción en el acceso al entorno de una red. En el contexto de Apache significa normalmente la restricción en el acceso a ciertas *URLs*.

- Algoritmo: Un proceso definido sin ambigüedades o un conjunto de reglas para solucionar un problema en un número finito de pasos. Los algoritmos para encriptar se llaman normalmente *algoritmos de cifrado*. **(Cuenca, 2003)**

Ventajas:

- Se ejecuta correctamente en la mayoría de los Sistemas Operativos.
- Es una tecnología gratuita de código fuente abierto.
- Este servidor es altamente configurable de diseño modular.
- Existe una gran variedad de módulos para Apache.
- Apache permite adecuar las respuestas ante los posibles errores que se puedan dar en el servidor.
- Permite crear archivos logs personalizados en correspondencia con las necesidades administrador.

1.4.2 AJAX

El término AJAX es un acrónimo de Asynchronous JavaScript + XML, que se puede traducir como JavaScript asíncrono + XML. Se puede definir como una técnica de desarrollo Web, que utiliza otras tecnologías existentes. AJAX no es una tecnología en sí mismo. En realidad, se trata de la unión de varias tecnologías que se desarrollan de forma autónoma y que se unen de formas nuevas y sorprendentes.

Permite manejar e intercambiar datos de forma no asíncrona con un servidor Web, permitiendo realizar operaciones en PHP u otro lenguaje sin necesidad de refrescar la página.

Las tecnologías que forman AJAX son:

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

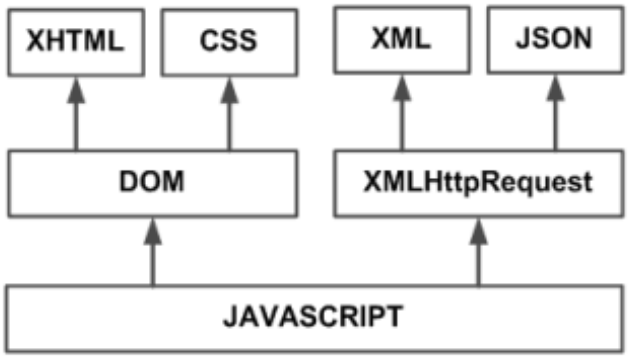

Figura 1. 7: Tecnologías agrupadas bajo el concepto de AJAX.

Desarrollar aplicaciones AJAX requiere un conocimiento avanzado de todas y cada una de las tecnologías anteriores. En las aplicaciones Web tradicionales, las acciones del usuario en la página (pinchar en un botón, seleccionar un valor de una lista, etc.) desencadenan llamadas al servidor. Una vez procesada la petición del usuario, el servidor devuelve una nueva página HTML al navegador del usuario.

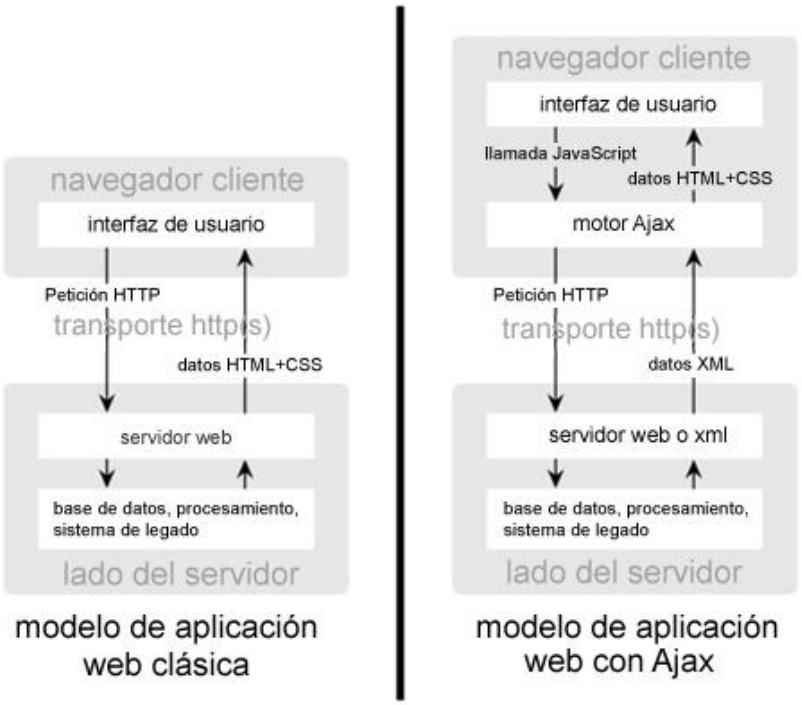

Figura 1. 8: La imagen de la izquierda muestra el modelo tradicional de las aplicaciones Web. La imagen de la derecha muestra el modelo de Ajax.

La técnica tradicional para crear aplicaciones Web funciona perfectamente, pero no crea una buena sensación al usuario. Al realizar peticiones al servidor con frecuencia, el usuario debe esperar a que se recargue la página con los cambios solicitados y la aplicación Web se convierte en algo más molesto que útil.

Ventajas:

- Las aplicaciones construidas con esta técnica eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor.
- El tiempo de espera para una petición se reduce ya que no se envía toda la página.
- Esta nueva capa intermedia mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor.
- Las peticiones más simples no requieren intervención del servidor, por lo que la respuesta es inmediata. Si la interacción del servidor requiere la respuesta del servidor, la petición se realiza de forma asíncrona mediante AJAX.
- AJAX en la mayoría de casos puede sustituir completamente a otras técnicas como Flash y en el caso de las aplicaciones más avanzadas, pueden sustituir a complejas aplicaciones de escritorio.
- Válido en cualquier plataforma y navegador.

1.4.3 Frameworks de desarrollo

Un Framework de desarrollo se encarga de definir y dar soporte a gran parte de los procesos que se llevan a cabo con frecuencia, es decir, implementa los principales patrones de la programación, tanto desde la visión de la arquitectura de un software hasta las implementaciones más usadas.

Por otra parte también brinda un conjunto de clases y librerías para ayudar a desarrollar y unir los diferentes componentes de un proyecto permitiendo que programadores y diseñadores se concentren en los requerimientos del software que se desea hacer y no tratando con los tediosos detalles de bajo nivel al proveer un sistema funcional. Facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.

1.4.3.1 Jquery como Framework para Javascript

jQuery es un Framework Javascript que sirve para la programación avanzada de aplicaciones, el mismo aporta una serie de funciones para realizar tareas habituales, exonerando de realizar tareas básicas debido a que ella están implementadas en el Framework. Convierte la programación del lado del cliente en un problema más sencillo, en resumen, simplifica muchos procedimientos Javascript que comúnmente se usan para programar la interacción de las páginas Web.

Por ejemplo, jQuery es de gran ayuda en la creación de interfaces de usuario, efectos dinámicos, aplicaciones que hacen uso de AJAX, etc. Al trabajar javascript con jQuery se obtiene una interfaz que con certeza funcionará correctamente en todos los navegadores. Simplemente con conocer las librerías del Framework y programar utilizando las clases, sus propiedades y métodos para la creación de las aplicaciones se pueden obtener los resultados deseados.

Ventajas:

- Es un producto estable, bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización del Framework.
- Es ligero y libre.
- Tiene una gran comunidad de creadores de plugins o componentes.
- Encadenamiento, una de las características más destacadas de jQuery es el encadenamiento, se refiere a que en una línea de código es posible cambiar un atributo a un elemento, ocultarlo y volver a mostrarlo con un efecto.
- Cuenta con una gran cantidad de selectores los que permiten tener acceso al DOM (Modelo de Objetos del Documento) de varias formas.
- Permite manipular los eventos (onclick, onmouseover, onmouseout, etc) de forma natural y permite trabajar con eventos como el funcional evento ready, dobleclick entre otros.
- De una manera sencilla, con muy pocas líneas de código es posible ejecutar acciones como post, get o un simple load gracias al AJAX con el que cuenta jQuery, eventos muy intuitivos como beforeSend, success, error, entre otros hacen muy fácil aplicar AJAX a una aplicación. **(Alvarez, 2009)**

1.4.3.2 Symfony como Framework de desarrollo

Como todo Framework de desarrollo, Symfony tiene como principal utilidad brindar a los desarrolladores la implementación de las funcionalidades más comunes y encapsular en funciones simples operaciones complejas. Symfony particularmente está diseñado para optimizar el desarrollo de aplicaciones Web. Se puede definir como un enorme conjunto de herramientas y utilidades que simplifican el trabajo de los desarrolladores Web, para ello separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación.

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios Web de comercio electrónico de primer nivel. Es compatible con la mayoría de los gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows. A continuación se muestran algunas de sus ventajas. **(Potencier, y otros, 2008)**

Ventajas:

- Este Framework es fácil de instalar y configurar en la mayoría de las plataformas, su correcto funcionamiento se garantiza en los sistemas Windows y *nix.
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible y configurable como para adaptarse a los casos más complejos.
- Basado en la premisa de *"convenir en vez de configurar"*, en la que el desarrollador sólo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la Web
- Posee buena estabilidad lo que permite el desarrollo de aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

1.4.4 Visual Paradigm como herramienta CASE

Visual Paradigm (VP) es una herramienta case (Computer-Aided Software Engineering), por lo que su función principal es ayudar a desarrollar. Utiliza el lenguaje de modelado de sistemas de software (UML) como lenguaje de modelado. VP soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Este software ayuda a construir aplicaciones con mayor rapidez, calidad y a un menor coste. Entre sus características se encuentra la de permitir dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Esta herramienta también proporciona abundantes tutoriales de UML.

Figura 1. 9: Relación de tecnologías.

Ventajas:

- Soporta múltiples usuarios trabajando sobre el mismo proyecto.
- Genera la documentación del proyecto automáticamente en varios formatos como Web y .pdf, y permite control de versiones.
- Posee buena robustez, usabilidad y portabilidad.
- Se integra con las siguientes herramientas Java:
  - Eclipse/IBM Web Sphere
  - JBuilder
  - NetBeans IDE
  - Oracle JDeveloper
  - BEA Web logic
- Está disponible en varias ediciones, cada una destinada a necesidades específicas: Enterprise, Professional, Community, Standard, Modeler y Personal.

1.4.5 Zend Studio for Eclipse

Es cierto que para desarrollar un proyecto pequeño se puede emplear cualquier editor de texto, pero cuando se trata de un trabajo grande se impone el uso de un buen entorno de desarrollo integrado (IDE), ya que su uso permite escribir con rapidez y efectividad el código fuente.

Zend Studio para Eclipse es un IDE que combina la tecnología de Zend y PHP de Eclipse Tools (PDT) para crear un IDE de gran potencia para el desarrollo de aplicaciones Web.

Ventajas:

- Depuración integrada de perfiles, código y permite probar las capacidades de cobertura.
- Apoyo con equipo de control de versiones de apoyo extensible, el apoyo a Zend Framework.
- Soporte para Web Services, amplio con múltiples idiomas y extensibilidad de la comunidad de código abierto de Eclipse.

Zend brinda la posibilidad de utilizar el Zend Studio for Eclipse para mejorar la calidad de los proyectos en PHP, agilizar los ciclos de desarrollos y simplificar la complejidad de los proyectos. El plugin incluye herramientas para edición, debugging, análisis, optimización y bases de datos, e incluso soportando los procesos del desarrollo por programación ágil.

1.5 Sistemas Gestores de Bases de Datos (SGBD)

Es el conjunto de programas que tienen como objetivo servir de interfaz entre la base de datos, el usuario y las aplicaciones. Entre las funcionalidades que debe permitir se encuentran:

- Definir una base de datos: Especificar tipos, estructuras y restricciones de datos.
- Construir la base de datos: Guardar los datos en algún medio controlado por el mismo SGBD.
- Manipular la base de datos: Realizar consultas, actualizarla, generar informes.

Así que se trata de un software de propósito general. Ejemplos de SGBD son Oracle y SQL Server de Microsoft.

Algunas de las características deseables en un Sistema Gestor de Bases de Datos son:

- Control de la redundancia: La redundancia de datos tiene varios efectos negativos (duplicar el trabajo al actualizar, desperdicio de espacio en disco, puede provocar inconsistencia de datos) aunque a veces es deseable por cuestiones de rendimiento.
- Restricción de los accesos no autorizados: cada usuario ha de tener unos permisos de acceso y autorización.
- Cumplimiento de las restricciones de integridad: El SGBD ha de ofrecer recursos para definir y garantizar el cumplimiento de las restricciones de integridad.

El gestor almacena una descripción de datos en un diccionario de datos, así como los usuarios y sus permisos. En los Sistemas Gestores de Bases de Datos se tienen:

Diccionario de datos (Alvarez, 2007)

Es una base de datos donde se guardan todas las propiedades de la base de datos, descripción de la estructura, relaciones entre los datos, etc.

El diccionario debe contener:

- La descripción externa, conceptual e interna de la base de datos.
- Las restricciones sobre los datos.
- El acceso a los datos.
- Las descripciones de las cuentas de usuario.
- Los permisos de los usuarios.
- Los esquemas externos de cada programa.

El administrador de la base de datos (Alvarez, 2007)

Es una persona o grupo de personas responsables del control del sistema gestor de base de datos.

Las principales tareas de un administrador son:

- La definición del esquema lógico y físico de la base de datos.
- La definición de las vistas de usuario.
- La asignación y edición de permisos para los usuarios.

- Mantenimiento y seguimiento de la seguridad en la base de datos.
- Mantenimiento general del sistema gestor de base de datos.

Los lenguajes (Alvarez, 2007)

Un sistema gestor de base de datos debe proporcionar una serie de lenguajes para la definición y manipulación de la base de datos. Estos lenguajes son los siguientes:

- Lenguaje de definición de datos (DDL), para definir los esquemas de la base de datos.
- Lenguaje de manipulación de datos (DML). Para manipular los datos de la base de datos.
- Lenguaje de control de datos (DCL). Para la administración de usuarios y seguridad en la base de datos.

1.5.1 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional orientado a objetos y libre, publicado bajo la licencia BSD(es la licencia de software otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*, pertenece al grupo de licencias de software Libre). PostgreSQL no es manejado por una sola empresa sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada PGDG (*PostgreSQL* Global Development Group).

Ventajas:

- Alta concurrencia: Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.
- Integridad de los datos: Claves primarias, llaves foráneas con capacidad de actualizar en cascada o restringir la acción y restricción no nula.
- Resistencia a fallas. Escritura adelantada de registros (WAL) para evitar pérdidas de datos en caso de fallos por: Energía, Sistema Operativo, Hardware.
- Es un SGBD Multiplataforma probado en Linux, Unix, BSD's, Mac OS X, Solaris, AIX, Windows.
- PITR. Puntos de recuperación en el tiempo.
- Tablespaces (Ubicaciones alternativas para los datos) .

- Replicación síncrona y asíncrona.
- Cumple con factores que determinan la calidad del software. (ISO 9126-1)
- Características operativas: Corrección, Fiabilidad, Eficiencia, Integridad, Facilidad de uso.
- Capacidad para soportar cambios: Facilidad de mantenimiento, Flexibilidad, Facilidad de prueba.

1.6 Metodologías de desarrollo de software

Desarrollar un software puede resultar una tarea compleja y tediosa dependiendo de los requerimientos del mismo. Cuando los requerimientos son muchos o complejos por el entorno donde se desarrolla el o los procesos que se desean automatizar, entonces surge la necesidad de organizar, planificar y controlar el trabajo, así como modelar algunos procesos mediante diagramas u otras utilidades, por lo que el empleo de una buena metodología de desarrollo de software, en correspondencia con las necesidades y dimensiones del proyecto que se desea realizar, se hace inminente.

Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Lo hacen desarrollando un proceso detallado con un fuerte énfasis en planificar inspirado por otras disciplinas de la ingeniería. No es más que el conjunto de pasos y procedimientos que deben seguirse para el desarrollo de software.

Características deseables de una metodología:

- Existencia de reglas predefinidas.
- Cobertura total del ciclo de desarrollo.
- Verificaciones intermedias.
- Planificación y control.
- Comunicación efectiva.
- Utilización sobre un abanico amplio de proyectos.
- Fácil formación.
- Herramientas CASE.
- Actividades que mejoren el proceso de desarrollo.
- Soporte al mantenimiento.
- Soporte de la reutilización de software.

De manera que el uso de una buena metodología resuelve entre otros, los siguientes problemas:

- Retrasos en la planificación: Llegada la fecha de entregar el software éste no está disponible.
- Sistemas deteriorados: El software se ha creado pero después de un par de años el coste de su mantenimiento es tan complicado que definitivamente se abandona su producción.
- Tasa de defectos: El software se pone en producción pero los defectos son tantos que nadie lo usa.
- Requisitos mal comprendidos: El software no resuelve los requisitos planificados inicialmente.
- Cambios de negocio: El problema que resolvía el software ha cambiado y el mismo no se ha adaptado.
- Falsa riqueza: El software hace muchas cosas técnicamente muy interesantes, pero no resuelven el problema del cliente, ni hace que éste gane más dinero.
- Cambios de personal: Después de unos años de trabajo los programadores comienzan a odiar el proyecto y lo abandonan.

1.6.1 Proceso Unificado de Desarrollo (RUP)

El Proceso Unificado es un proceso de software genérico que puede ser utilizado para una gran variedad de sistemas de software, se puede utilizar en diferentes tipos de organizaciones, diferentes niveles de competencia y diferentes tamaños de proyectos debido a su alto poder de configuración. El mismo necesita que el equipo de desarrollo le priorice la identificación temprana de riesgos críticos. Los resultados de cada iteración, en especial los de la fase de Elaboración, deben ser seleccionados de manera que permitan asegurar que los riesgos principales tienen la prioridad requerida.

El proceso se repite a lo largo de una serie de ciclos que forman la vida de un sistema, donde cada ciclo concluye con una versión del producto para los clientes y posee cuatro fases: inicio, elaboración, construcción y transición. Cada fase se divide a su vez en iteraciones. Con cada una de estas fases se persiguen hitos, a los que se le debe dar cumplimiento con la finalización de la misma:

- Inicio: El objetivo en esta etapa es determinar la visión del proyecto.
- Elaboración: En esta etapa el objetivo es determinar la arquitectura óptima.
- Construcción: El objetivo es llegar a obtener la capacidad operacional inicial.

- Transición: El objetivo es llegar a obtener el *release* del proyecto. Vale mencionar que el ciclo de vida que se desarrolla por cada iteración, es llevado bajo disciplinas como se muestra en la figura 1.10.

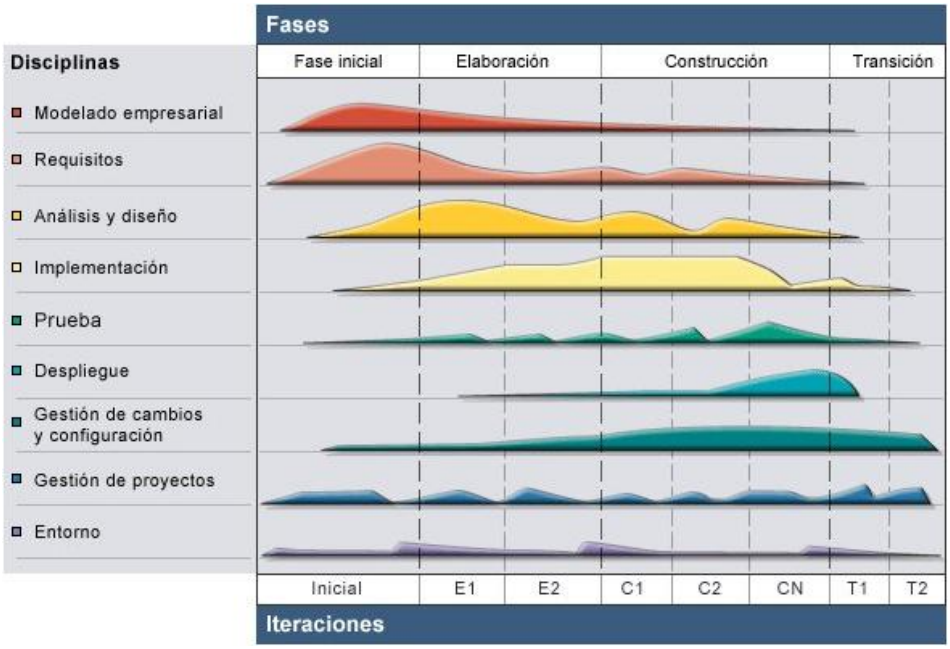

Figura 1. 10: Fases y flujos de trabajo.

Los principales aspectos que distinguen al Proceso Unificado de otras metodologías de su tipo se centran en que es: Dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental.

Dirigido por casos de uso

El objetivo principal de la creación de un software con calidad es la plena satisfacción del usuario. Esto trae como consecuencia que se haga necesario conocer lo que quieren y necesitan los usuarios del sistema, que pueden ser otros sistemas también. Por lo que RUP hace uso de los casos de usos para capturar los requerimientos funcionales, para definir y priorizar las tareas más importantes y significativas con el objetivo de desarrollar primero las más necesarias.

Centrado en la arquitectura

El concepto de arquitectura de software involucra los aspectos estáticos y dinámicos más significativos del sistema. También está influenciada por muchos otros factores, tales como la plataforma de software en la que se ejecutará, la disponibilidad de componentes reutilizables, consideraciones de instalación, sistemas legados, requerimientos no funcionales (ej. desempeño, confiabilidad). **(Espinoza, 2009)**

Iterativo e incremental

Elaborar un producto de software puede ser una tarea larga y tediosa cuando se trata de un proyecto grande, lo que puede implicar una larga duración en el ciclo de desarrollo del mismo. Esto trae consigo que se haga necesario dividir el trabajo en pequeños mini-proyectos donde la culminación de cada uno implique un incremento que tribute al producto en general. Para garantizar la productividad de cada iteración se debe planificar y controlar cada una de las tareas a cumplir.

RUP es precisamente la metodología de desarrollo de software seleccionada para implementar el producto resultante de SGDG. Dentro de esta metodología existen varios roles en los distintos flujos de trabajo que la conforman, cada uno juega un papel importante en el ciclo de desarrollo. Uno de los roles de gran importancia es el de implementador o desarrollador.

El implementador es el encargado de, metafóricamente, darle vida a las soluciones propuestas para satisfacer las necesidades del cliente y validar el correcto funcionamiento de las mismas. Entre sus tareas se encuentran:

- Analizar el comportamiento en tiempo de ejecución.
- Desarrollar productos de trabajo de instalación.
- Ejecutar pruebas de desarrollador.
- Implementar elementos de diseño.
- Implementar la prueba de desarrollador.
- Implementar los elementos de comprobación.

Funciones que se resumen en el desarrollo y las pruebas de los componentes de prueba y los subsistemas correspondientes. Para el correcto cumplimiento de sus responsabilidades debe tener

habilidades de programación, conocimiento del sistema o aplicación que se somete a prueba, familiaridad con las herramientas de prueba y de automatización de prueba.

1.7 Conclusiones parciales

Mediante el desarrollo de este capítulo se ha realizado un estudio y análisis de las características y ventajas que poseen las principales tecnologías y herramientas de la ingeniería del software con el objetivo de justificar su uso, teniendo en cuenta las particularidades del presente trabajo así como el hecho de que la mayoría de estas son libres y de código abierto, lo que reduce en gran medida el costo del proyecto, garantizando de esta forma poca dependencia hacia los productores de cada una de las tecnologías.

CAPÍTULO 2: Descripción de la Solución Propuesta

2.1 Introducción

Este capítulo se centra en valorar el diseño de clases propuesto por el analista, describir conceptos relacionados con las definiciones de estilo y estándar de código. También se valoran las posibles implementaciones teniendo en cuenta los componentes o módulos existentes y por último se muestra una descripción de las principales clases y funcionalidades.

2.2 Valoración crítica del diseño propuesto por el analista

El Framework de desarrollo Web Symfony está basado en un patrón clásico del diseño Web conocido como arquitectura Modelo Vista Controlador (MVC), que está formado por tres niveles:

- El Modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La Vista transforma el modelo en una página Web que permite al usuario interactuar con ella.
- El Controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

La arquitectura MVC es de uso frecuente en aplicaciones Web, este separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos, el modelo, la vista y el controlador. Con esta distribución se logra que una aplicación pueda ser utilizada en distintos dispositivos cambiando sólo las interfaces o vistas, manteniendo el controlador y el modelo, ejemplo de ello es un sitio Web que requiera ser utilizado desde computadores y dispositivos móviles.

El *modelo* tiene como objetivo principal encargarse de la abstracción relacionada con los datos, logrando que tanto la vista como el modelo no dependan de un gestor de bases de datos específico. Finalmente el *controlador* tiene como tareas interconectar el modelo con la vista, controlando y especificando la interacción entre los mismos, y se encarga de detalles como por ejemplo el protocolo utilizado para las

peticiones (HTTP, consola de comandos, email). A continuación se muestra una imagen de la relación entre las capas de este patrón:

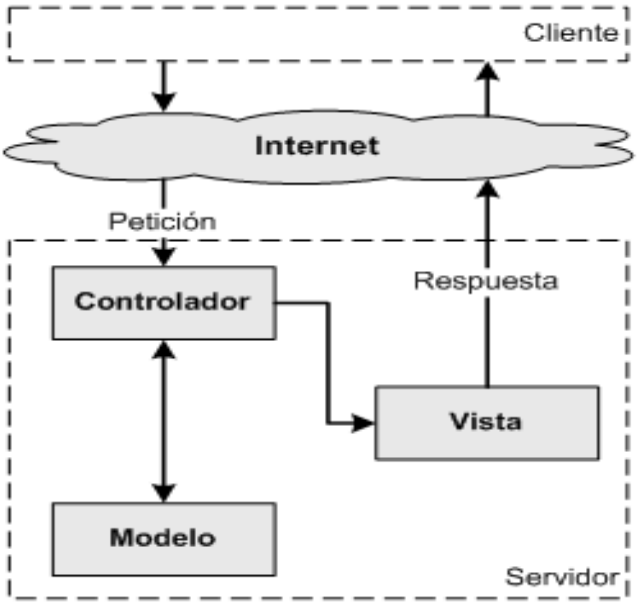

Figura 2. 1: El patrón MVC.

Valoraciones del diseño propuesto

El diseño propuesto fue realizado con la versión 1.1.4 de Symfony. Este Framework tiene su forma peculiar de implementar, la capa del ***controlador***, que contiene el código que enlaza la lógica de negocio con la presentación, está dividida en varios componentes que se utilizan para varios propósitos:

- El controlador frontal es el único punto de entrada a la aplicación. Carga la configuración y determina la acción a ejecutarse.
- Las acciones contienen la lógica de la aplicación. Verifican la integridad de las peticiones y preparan los datos requeridos por la capa de presentación.
- Los objetos request, response y session dan acceso a los parámetros de la petición, las cabeceras de las respuestas y a los datos persistentes del usuario. Se utilizan muy a menudo en la capa del controlador.
- Los filtros son trozos de código ejecutados para cada petición, antes o después de una acción. Por ejemplo, los filtros de seguridad y validación son comúnmente utilizados en aplicaciones Web. Puedes extender el Framework creando tus propios filtros.

Todas las peticiones Web son manejadas por un solo controlador frontal, que es el único punto de entrada de toda la aplicación en correspondencia con el entorno en el que se trabaje. Este controlador al recibir una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL escrita (o pinchada) por el usuario. Por ejemplo, la siguiente URL llama al script index.php (que es el controlador frontal) y será entendido como llamada a la acción miAccion del módulo mimodulo: *http://localhost/index.php/mimodulo/miAccion*

Las acciones son el corazón de la aplicación, puesto que contienen toda la lógica de la aplicación. Las mismas utilizan el modelo y definen variables para la vista. Cuando se realiza una petición Web en una aplicación Symfony, la URL define una acción y los parámetros de la petición. Las acciones son métodos con el nombre executeNombreAccion de una clase llamada nombreModuloActions que hereda de la clase sfActions y se encuentran agrupadas por módulos. **(Potencier, et al., 2008)**

La parte encargada de producir páginas Web como resultado de las acciones se llama vista. La ***vista*** en Symfony está compuesta por diversas partes, estando cada una de ellas construidas de manera que puedan ser fácilmente modificables por la persona encargada de trabajar con el diseño de las aplicaciones. Gracias a esto se puede conseguir que los diseñadores con mínimos conocimientos de PHP puedan trabajar ignorando en gran medida la lógica de la aplicación.

Una de las partes de gran importancia que forma la vista es la plantilla, su contenido está formado por código HTML y algo de código PHP sencillo relacionado con la llamada a las variables definidas en la acción. El contenido de la plantilla se inserta en la plantilla general (layout), o si se mira desde otro punto de vista, la plantilla general *decora* la plantilla. Este comportamiento es una implementación del patrón de diseño llamado "decorator" y que se muestra en la figura 2.3.

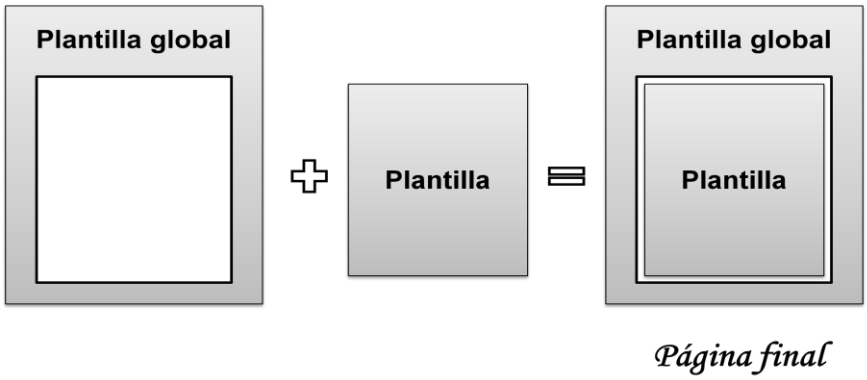

Figura 2. 2: Plantilla decorada con una plantilla global.

La lógica de negocio de las aplicaciones Web depende casi siempre de su ***modelo*** de datos. El componente que se encarga por defecto de gestionar el modelo en Symfony es una capa de tipo ORM (*mapeo de objetos a bases de datos*), que se encarga de mapear la base de datos utilizada en clases. En las aplicaciones Symfony, el acceso y la modificación de los datos almacenados en la base de datos se realiza mediante objetos; de esta forma nunca se accede de forma explícita a la base de datos. Este comportamiento permite un alto nivel de abstracción y permite una fácil portabilidad. **(Potencier, et al., 2008)**

La principal ventaja que aporta el ORM es la reutilización, ya que brinda la posibilidad de llamar a los métodos de un objeto de datos desde varias partes de la aplicación y desde diferentes aplicaciones, ya que la distribución de archivos en Symfony lo permite. Esta capa se encarga de encapsular la lógica de los datos, en la versión utilizada se usa propel como ORM, esta genera 5 clases por cada tabla de la base de datos para lograr utilizar objetos en vez de registros y clases en vez de tablas. Las clases de propel se encargan de contener el código relacionado con el acceso a datos y la lógica de los mismos, así como su mapeo.

En la implementación del diseño propuesto se pretende agrupar la mayor parte de la lógica en la capa del modelo, aumentando considerablemente la reutilización de sus clases. También se pretende lograr que cada clase se encargue de realizar las funcionalidades de la tabla que le corresponde en la BD para lograr un bajo acoplamiento entre las clases. Logrando con esto que las acciones se dediquen mayormente a enlazar al modelo con la vista, trayendo como consecuencia una alta cohesión.

Seguidamente se presenta un ejemplo del diseño que propone el analista para el caso de uso Gestionar Yacimiento.

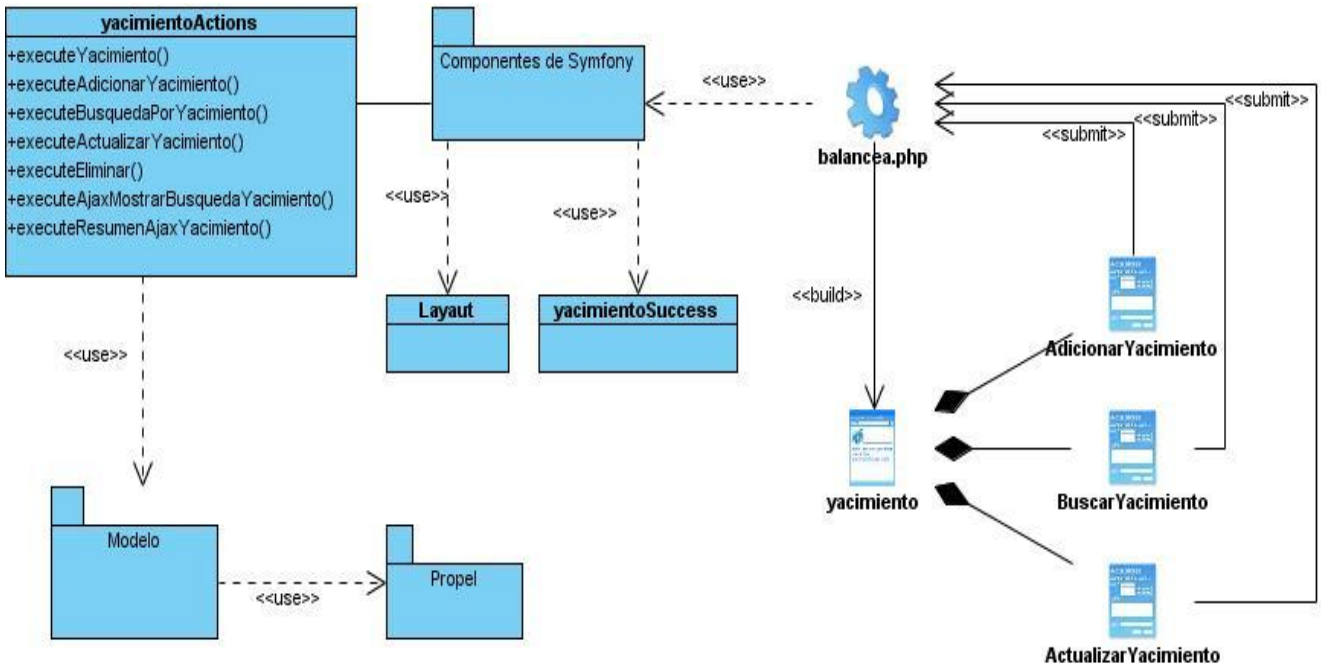

Figura 2. 3: Diagrama de Clases del Diseño. Gestionar Yacimiento.

2.3 Estilos de programación

El término estilo de programación se refiere específicamente a la manera en que se le da formato al código fuente. Un estilo organizado y legible se impone cuando se trabaja en proyectos que implican la escritura de grandes cantidades de líneas de código; al elegir un estilo de código legible se asegura que el mismo sea fácil de entender, no sólo para el programador sino también para otras personas interesadas en su revisión.

Un código desordenado trae como resultado que con el transcurso del tiempo ni el propio programador entienda lo que escribió, lo que trae como consecuencia que la ampliación, reparación o evaluación del mismo sea un trabajo tedioso. El hecho de que un programa no posea un estilo claro no implica que tenga

mal funcionamiento, aunque si está escrito desordenadamente tiene una probabilidad mayor de no funcionar correctamente y también dificulta su depuración.

El programador posee mecanismos básicos para comunicarse con sus lectores:

- El estilo.
- Los comentarios.
- Los nombres de las variables, constantes y métodos.
- Los espacios en blanco (el sangrado).

Estilo

Dentro del estilo se pueden incluir temas como el de las variables, expresiones y estructura de programa.

- **Variables**
  - Primeramente se debe lograr que las variables de las clases no sean públicas con el objetivo de que nadie pueda modificarlas desde fuera de la clase.
  - El tiempo de vida de las variables debe ser el mínimo posible, o sea, ser declaradas en un ámbito bastante reducido con el objetivo de no provocar confusiones en cuanto a su uso.
  - Es recomendable que las variables de ámbito amplio tengan nombres largos y significativos y las de uno reducido tengan nombres cortos, logrando que las variables que van a ser accedidas desde varios lugares queden claras para el lector.
  - Las variables deben ser inicializadas en el momento de su creación para prever que por algún motivo contengan valores desconocidos.
  - Una variable nunca debe tener más de un significado.

- **Expresiones**
  - Las expresiones numéricas o condiciones lógicas no deben ser complejas, si se tiene alguna con estas características es recomendable la utilización de variables o métodos auxiliares.
  - En la elaboración de una expresión condicional se debe poner en el **if** el caso normal y en el **else** el caso excepcional, para mayor entendimiento.
  - Se debe evitar las comparaciones con **true** y **false**.

- **Estructuras de programa**
  - Las instrucciones **break** o **continue** se deben usar sólo en casos en los que sean imprescindibles, para que los ciclos sean legibles, sin sorpresas.
  - Cuando se tienen métodos que devuelven un resultado dependiente de varias condiciones no se debe acumular, o sea, se debe de retornar en cuanto se tenga.
  - Un ciclo o **bucle** no debe ocupar más de una pantalla (unas 20 líneas de código) desde su inicio hasta su final, para que se pueda ver completamente sin necesidad de desplazamiento vertical, se recomienda usar métodos auxiliares.

Los comentarios

Al documentar el código se logra que no sólo el ordenador sepa qué hacer, sino que también cualquier persona sepa que hace cada funcionalidad y porqué. Las soluciones de problemas complejos equivalen a largas horas de razonamiento y lo más lógico es que quede documentado y que para una futura valoración tenga como base la solución anterior con el objetivo de no tener que repensarla.

Los programadores deben saber que sus programas no están exentos de errores y descubrirlos sólo es cuestión de tiempo. Por otra parte los programas sufren modificaciones a lo largo de su vida, debido a que en el momento que fueron escritos existían determinadas necesidades que pudieron cambiar, esto hace que surja la necesidad de modificarlos.

Preguntas que debe responder una buena documentación:

- ¿De qué se encarga una clase? ¿Un paquete?
- ¿Qué hace un método?
- ¿Cuál es el uso esperado de un método?
- ¿Para qué se usa una variable?
- ¿Cuál es el uso esperado de una variable?
- ¿Qué algoritmo se usa? ¿De dónde se ha sacado?
- ¿Qué limitaciones tiene el algoritmo?
- ¿Qué se debería mejorar, si hubiera tiempo?

Algunos tipos de comentarios

- javadoc
  Comienzan con los caracteres "/**", se pueden prolongar a lo largo de varias líneas (que probablemente comiencen con el caracter "*") y terminan con los caracteres "*/".
- una línea
  Comienza con los caracteres "//" y terminan con la línea.
- tipo C
  Comienzan con los caracteres "/*", se pueden prolongar a lo largo de varias líneas (que probablemente comiencen con el caracter "*") y terminan con los caracteres "*/".

Se debe incluir comentarios:

- Al principio de cada clase.
- Al principio de cada método.
- Ante cada variable de clase.
- Al principio de fragmento de código no evidente.
- A lo largo de los bucles.
- Siempre que se haga algo fuera de lo común.
- Siempre que el código no sea evidente.

Nomenclatura

Los nombres de las clases, métodos, variables y constantes son una forma bastante clara de transmitirle al lector lo que piensa el programador, por lo que es recomendable utilizar estos nombres de manera que sean lo más nemotécnicos posibles.

Sangrado

Cuando se tiene como objetivo la escritura de un código limpio y legible, el sangrado (cambios de línea y espacio en blanco al principio) se hace inevitable. Un correcto sangrado garantiza que el lector pueda entender la relación existente entre las instrucciones sin necesidad de buscar llaves u otros elementos. El sangrado es tan importante que hay lenguajes de programación que basan su sintaxis en el mismo.

Actualmente existen muchos estilos de programación con diferentes características, el programador es libre de elegir uno en específico o de usar lo mejor a su apreciación de cada uno. A continuación se mostrará una selección de los estilos más usados en la actualidad.

El estilo **K&R** usado mayormente en C y PHP, obtuvo este nombre porque fue usado por Kernighan y Ritchies en su libro "The C Programming Language". Básicamente consiste en abrir la llave en la misma línea de la declaración de la orden y cerrar la llave en el mismo nivel que la declaración, como se muestra en el ejemplo siguiente.

```
public static function AdicionarYacimiento($datosYacimiento){
  $yacimiento = new Tyacimientoagua();
  $yacimiento->setNombre($datosYacimiento[0]);
  $yacimiento->setIdprovincia($datosYacimiento[1]);
  $yacimiento->setFechaEstimacion($datosYacimiento[2]);
  $yacimiento->setIdmunicipio($datosYacimiento[3]);
  $yacimiento->save();
}
```

Como *ventajas* se tiene que la llave de apertura no requiere una línea para ella sola y la llave de cierre se alinea con el comienzo de la instrucción a la que pertenece, como *desventaja* se tiene que la llave de cierre utiliza una línea sólo para ella, esto era un problema para los tiempos en que se usaban terminales que mostraban 24 líneas.

El estilo **Allman** fue definido por Eric Allman. Consiste en ubicar las llaves en una línea independiente e identar el código debajo de ellas. La llave de cierre tiene el mismo identado que la de inicio. Ejemplo:

```
public static function AdicionarYacimiento($datosYacimiento)
{
  $yacimiento = new Tyacimientoagua();
  $yacimiento->setNombre($datosYacimiento[0]);
  $yacimiento->setIdprovincia($datosYacimiento[1]);
  $yacimiento->setFechaEstimacion($datosYacimiento[2]);
  $yacimiento->setIdmunicipio($datosYacimiento[3]);
  $yacimiento->save();
}
```

Su principal *ventaja* está en que se puede diferenciar claramente cuando se está leyendo una instrucción de bloque o de una condicional. Y como *desventaja* tiene el uso de una línea para cada llave.

El estilo **GNU** es bastante similar al **Allman** con la diferencia que este estilo propone que las llaves sean identadas por 2 espacios y el código que contiene identada por 2 espacios adicionales. Ejemplo:

```
public static function AdicionarYacimiento($datosYacimiento)
  {
    $yacimiento = new Tyacimientoagua();
    $yacimiento->setNombre($datosYacimiento[0]);
    $yacimiento->setIdprovincia($datosYacimiento[1]);
    $yacimiento->setFechaEstimacion($datosYacimiento[2]);
    $yacimiento->setIdmunicipio($datosYacimiento[3]);
    $yacimiento->save();
  }
```

También se utilizan con gran frecuencia el estilo de Kernel Normal Form, el que se usa mayormente para el código de la distribución del software del sistema operativo de Berkeley y el Whitesmiths también llamado estilo Wishart.

2.4 Estándares de codificación

Un estándar de codificación integral contiene la mayoría de los aspectos concerniente a la generación de código fuente y las reglas que se siguen para la escritura del mismo. La buena utilización de un estándar permite minimizar los costos que surgen en el desarrollo de un sistema. Principalmente a los programadores que trabajan en equipo se las hace necesario que el código esté lo más estandarizado posible, de manera que al leer el código parezca que fue escrito por un solo programador. Evidentemente esto permite que la reutilización de instrucciones hechas por otra persona se convierta en una tarea cómoda.

Si se da el caso de que un sistema utilice código existente, el estándar debe contemplar cómo trabajar con este código. Todo sistema necesita de mantenimientos a largo o corto plazo y es por esto que la legibilidad del código es un factor determinante en el proceso de mantenimiento del mismo, a un código

legible es fácil añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar su rendimiento.

Por las razonas antes expresada es altamente recomendable que se utilice un estándar de código para la creación de proyectos desarrollados por un equipo de trabajo o en sistemas de gran alcance, pero sólo si se va a aplicar desde el inicio hasta el fin del proyecto. Su aplicación dará como resultado un código fácil de entender y mantener.

Con el objetivo de garantizar la legibilidad y mantenimiento del código fuente en la implementación del Módulo Inventario Nacional de Recursos de Aguas Minerales se utilizará el estándar que se expone a continuación.

Estándares de codificación PHP:

Descripción de las clases:

A cada clase la antecede una breve descripción de la misma utilizando un comentario multilínea que debe tener el siguiente formato:

```
/**
 * class: MiClase
 * description: Esta clase es para ...
 * update date: 23-10-2008
*
 * @package sgdg
 * @subpackage yacimientos
 * @author Nombre del Desarrollador
 * @version Sistema de Gestión de Datos Geológicos versión 1.0
 */
```

Descripción de las funciones

Las funciones deben tener una breve descripción de las mismas dentro de un comentario multilínea con el siguiente formato:

```
/**
  * function: MiFuncion
  * description: Esta función es para ....
 *
  * @author Nombre del Desarrollador
  * @param string $nombre , int edad ...
  * @return bool $estado
  */
```

Un ejemplo de cómo queda una clase con sus funciones:

```
<?php
/**
 * class: yacimientosActions
 * description: Esta clase contiene las acciones del Módulo yacimientos.
 * update date: 23-10-2008
 *
 * @package    sgdg
 * @subpackage yacimientos
 * @author     Reinaldo Alberto Alvarez Fernandez
 * @version    Sistema de Gestión de Datos Geológicos versión 1.0
 */
class yacimientosActions extends sfActions
{
 /**
  * function: executeIndex
  * description: Esta función permite ejecutar la accion index.
  *
  * @author Ing. Dianet Utria.
  * @param sfRequest $request
  * @return
```

```
  */
  public function executeIndex($request)
  {
/**
   * function: executeAdicionarYacimiento
   * description: Esta función permite agregar un yacimiento
   * y luego redireccionar al listado de yacimientos existentes.
   *
   * @author Reinaldo Alberto Alvarez Fernandez
   * @param
   * @return string Header con la localización de la página principal
   */
  public function executeLogin()
  {
          return  $this->redirect('yacimientos/index');
  }
}
?>
```

Comentarios de las variables:

Los comentarios de las variables se deben caracterizar por ser cortos y concisos, el mismo debe estar ubicado al final de la línea de código en cuestión, el tipo de comentario a usar es el de una sola línea.

Ejemplo:

```
 public function ProcesarDatos()
  {
    $contador = 0;      // Contador de yacimientos
    $estado = false;    // Estado de los yacimientos
  }
```

Programación en las clases de las Acciones y las de Acceso a Datos

Con el objetivo de proveer al sistema de alta cohesión, las funciones englobadas en las clases de las Acciones no deben incluir funcionalidades que se encarguen de consultas a la base de datos, estas funcionalidades serán implementadas en el modelo y serán invocadas desde las acciones. Ejemplo:

```
public function executeActualizar()
{
  if($this->getRequest()->getMethod() != sfRequest::POST)
  {
    $id = $this->getRequestParameter('dep');
    Propel::getConnection()->executeQuery('set search_path to balancea');
    $this->deposito = YacimientoPeer::retrieveByPK($id);
    $this->clasificacion = YacimientoPeer::ObtenerClasificacion();
    $this->provincia = YacimientoPeer::ObtenerProvincia();
    }
  }
```

El estándar que se utiliza pone en práctica dos notaciones muy importantes a la hora de codificar, las que se nombran a continuación:

Notación **PascalCasing**: Los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas iniciando cada palabra con letra mayúscula; Ejemplo: **N**otacion**P**ascal**C**asing.

Notación **CamellCasing**: Los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas iniciando cada palabra con letra mayúscula excepto la primera que debe iniciar con minúscula. Ejemplo: notacionCamellCasing.

Además de lo planteado anteriormente, existe otro conjunto de cuestiones que el implementador debe tener presente, que están incluidas en el estándar seleccionado. Algunas de estas cuestiones se exponen a continuación debido a su importancia.

Sobre la Organización Visual

- No manejar en los programas más de una instrucción por línea.
- Declarar las variables en líneas separadas.
- Añadir comentarios descriptivos junto a cada declaración de variables, si es necesario.
- La indentación tendrá una longitud de cuatro espacios.
- Insertar una línea en blanco antes y después de una declaración de datos que aparezca entre instrucciones ejecutables.
- Deben incluirse espacios en ambos lados de los operadores binarios.
- Los operadores unarios (++,--, etc.) deben ponerse junto a sus operandos sin espacios intermedios.

Características a considerar para lograr un buen programa:

- Ejecución correcta: Funciona de acuerdo a las especificaciones.
- Ejecución eficiente: Uso mínimo de recursos de tiempo y memoria.
- Fácil de leer y comprender.
- Fácil de depurar.
- Fácil de mantener.
- Interfaz de usuario independiente de las funciones de cálculo.

Para mayor compresión se puede consultar el anexo A.

2.5 Análisis de posibles implementaciones, componentes o módulos ya existentes

Debido al desarrollo que ha alcanzado el hombre en todas las esferas, cuando se tiene como objetivo la creación de algo nuevo, antes que todo se debe investigar qué existe acerca del tema en cuestión. El resultado de esta búsqueda es lo que ayudará a la hora de tomar la decisión de seguir o no con el proyecto planificado. Bajo esta filosofía es que esencialmente se debe trabajar hoy en el mundo de las investigaciones científicas y especialmente en el mundo de la informática y el desarrollo de software.

Puede que al realizar una búsqueda se tenga como resultado que no existe información sobre el tema en el que se quiere trabajar por lo que no quedaría más remedio que crearlo todo desde cero. Ahora, si la

búsqueda arroja información útil sólo quedaría planificar la forma en que se utilizará la misma. En el caso de esta investigación son de gran importancia las implementaciones y trozos de código que puedan ayudar a la construcción del sistema de gestión.

La reutilización de código evita entre otras cosas, la pérdida de tiempo en la creación de elementos que existen y se pueden utilizar. Es por este motivo que se decide utilizar un Framework de desarrollo para la implementación del sistema, permitiendo a los programadores ocuparse de las cosas específicas del negocio en cuestión. El uso de symfony proporciona una gran cantidad de código que se puede reutilizar. Ejemplo de ello son los helpers, que no son más que trozos de código html encapsulados en funciones PHP.

Symfony también brinda la posibilidad de crear fragmentos de código reutilizable desde cualquier lugar de la aplicación. Si se necesita reutilizar código de las plantillas (*Success*) el Framework permite crear elementos parciales, elementos en los cuales se pueden escribir código HTML y PHP. De la misma forma que se aplica el patrón MVC en las acciones y plantillas también se puede manifestar en los elementos parciales, para eso se hace necesario utilizar componentes. Un componente es como una acción, solo que está relacionado con un elemento parcial específico y su principal función es encargarse de la lógica relacionada con los datos que se mostrarán en el elemento parcial al que pertenezca.
También para generar los reportes en formato pdf se utiliza la librería *fpdf* la cual brinda un conjunto de funciones que permiten crear ficheros y darle la estructura necesaria a la información que se desea exportar.

2.6 Descripción de las nuevas clases u operaciones fundamentales

A continuación se muestra un ejemplo de las nuevas funcionalidades generadas con el objetivo de garantizar el buen funcionamiento del módulo Inventario Nacional de Recursos de Aguas Minerales.

Tabla 2. 1: Descripción de la clase yacimientoActions

Nombre: yacimientoActions
Tipo de la clase: Controladora

Para cada responsabilidad	
Nombre:	**Descripción:**
executeYacimiento()	Es la función principal para poder acceder al módulo de la gestión de los yacimientos.
executeBuscarYacimiento ()	Es la función que se encarga de buscar los yacimientos por distintos criterios.
executeAdicionarYacimiento ()	Es una función encargada de enlazar la vista y el método de la capa modelo encargado de adicionar un yacimiento.
executeActualizarYacimiento ()	Es una función encargada de enlazar la vista y el método de la capa modelo encargado de modificar un yacimiento.
executeEliminar ()	Es una función encargada de enlazar la vista y el método de la capa modelo encargado de eliminar un yacimiento.
executeAjaxMunicipio ()	Es la función que se encarga de llamar a la función que retorna los municipios de una provincia dada.

Tabla 2. 2: Descripción de la clase del modelo TyacimientoaguaPeer

Nombre: TyacimientoaguaPeer	
Tipo de la clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre:	**Descripción:**
MostrarTodos ()	Devuelve todos los yacimientos de la BD.
BuscarYaciminetoTexto ()	Devuelve todos los yacimientos de la BD que coincidan con los criterios de búsquedas establecidos en la función.
CantidadYaciminetoTexto ()	Devuelve la cantidad de yacimientos que coincidan con los criterios de búsquedas

	establecidos en la función.
Adicionar ()	Función encargada de adicionar un yacimiento.
Actualizar ()	Función encargada de actualizar un yacimiento a partir de su identificador.
Eliminar ()	Función encargada de eliminar un yacimiento a partir de su identificador.
EliminarDesdeFuente ()	Función encargada de eliminar un yacimiento de la Base de datos, a partir de su identificador.
SelYacimientoPorFuente ()	Devuelve un yacimiento a partir del identificador de una fuente.
BusquedaGeneral ()	Devuelve todos los yacimientos de la BD que coincidan con los criterios de búsquedas establecidos en la función.
resumenYacimiento ()	Devuelve una selección de datos de un yacimiento, fuentes y recursos del mismo.
reporteConcesionarioYacimientosFuente()	Devuelve todos de datos de un yacimiento, fuentes y recursos del mismo, en correspondencia con los parámetros recibidos.

Tabla 2. 3: Descripción de la clase Tyacimientoagua

Nombre: Tyacimientoagua	
Tipo de la clase: Entidad	
Atributo	
idyacimiento	Entero
nombre	varchar
idprovincia	Entero

fecha_estimacion	date
idmunicipio	Entero
aTprovincia	Objeto
Para cada responsabilidad	
Nombre:	**Descripción:**
save()	Se encarga de Adicionar un yacimiento en la Base de Datos.
delete()	Se encarga de eliminar un yacimiento de la Base de Datos.
doSelect()	Función encargada de ejecutar las consultas.
getIdyacimiento ()	Devuelve el identificador de un yacimiento determinado.
getNombre ()	Devuelve el nombre de un yacimiento determinado.
getIdprovincia ()	Devuelve el identificador de la provincia de un yacimiento determinado.
getFechaEstimacion ()	Devuelve la fecha de un yacimiento determinado.
getIdmunicipio ()	Devuelve el identificador del municipio de un yacimiento determinado.
setIdyacimiento ()	Cambia el identificador de un yacimiento específico.
setNombre ()	Cambia el nombre de un yacimiento específico.
setIdprovincia()	Cambia el identificador de la provincia de un yacimiento específico.
setFechaEstimacion()	Cambia la fecha de un yacimiento específico.
setIdmunicipio()	Cambia el identificador del municipio de un yacimiento específico.

2.7 Conclusiones Parciales

Concluido el análisis y valoración del diseño propuesto por el analista se ha determinado que el mismo proporciona una base sólida para llevar a cabo las actividades correspondientes a la implementación. La selección de un estándar de código legible y organizado le proporciona el código fuente generado claridad y alta posibilidad de reutilización. Este estándar también garantiza que las revisiones y refactorizaciones de código futuras se puedan llevar a cabo con gran facilidad.

CAPÍTULO 3: Validación de la Solución Propuesta

3.1 Introducción

Todo sistema implementado por los humanos puede tener errores y por esto que se hace necesario encontrar la forma adecuada de mitigarlos. Por tal motivo al obtener un conjunto de código fuente es recomendable validar el buen funcionamiento del mismo, teniendo como medida los requerimientos del software en cuestión. En este capítulo se pretende diseñar los casos de pruebas que tengan una alta probabilidad de encontrar errores. Los diseños de casos de pruebas se realizarán mediante pruebas de caja negra y dentro de esta el método de partición equivalente.

3.2 Descripción general de las pruebas

Las pruebas de software tienen como principal meta demoler el mismo. Contradictoriamente a lo que se piensa acerca de que una prueba es buena cuando no encuentra errores, se puede decir que una prueba tiene éxito cuando descubre en el software errores que hasta el momento no se habían detectado. Las pruebas de software no garantizan que el software se encuentre libre de defectos, sólo muestran hasta qué punto el sistema funciona de acuerdo a los requisitos especificados. Entre otras cosas, las pruebas también se pueden ver como una medida de la fiabilidad y la calidad general del producto.

A continuación se comentarán brevemente algunos de los métodos de diseños de casos de pruebas existentes.

Pruebas de Validación: Estas pruebas están encaminadas a comprobar que el software cumpla con los requisitos del análisis.

Pruebas de Caja Blanca: Conocida también como prueba de caja de cristal es un tipo de prueba que garantiza que se ejecuten al menos una vez todos los caminos independientes de cada módulo. También

garantiza que se ejerciten todas las decisiones lógicas, siendo estas verdaderas o no, de igual forma comprueba que los ciclos (*bucles*) se ejecuten con sus límites operacionales.

Pruebas de Caja Negra: Nombrada también prueba de comportamiento tiene como campo de acción los requisitos funcionales. Las pruebas de caja negra tienen como metas detectar funciones con mal funcionamiento o ausentes, errores de interfaz, errores de rendimiento, entre otros.

En esta investigación se utiliza para el diseño de casos de prueba el método de prueba de caja negra conocido como **partición equivalente**. Este método se centra en la evaluación de clases de equivalencia para una condición determinada. Una clase de equivalencia puede definirse como un conjunto de objetos que estén relacionados simétrica o transitivamente, la misma define un conjunto de estados válidos y no válidos.

3.3 Búsqueda o diseño de las pruebas que permitan validar la solución propuesta

A continuación se exponen las secciones a probar para 5 de los casos de usos del sistema, las matrices de datos correspondientes a los mismos se presentan en el Anexo B.

3.3.1 Gestionar Yacimiento

Descripción General: El CU comienza cuando un Administrador desea adicionar, modificar o eliminar algún yacimiento. El sistema brinda las opciones de adición, búsqueda, modificación y eliminación al Administrador. Este selecciona la opción deseada, realiza la operación y termina el CU con la actualización de los datos.

Condiciones de Ejecución: El usuario debe estar autenticado y poseer los privilegios necesarios para realizar cualquiera de las acciones antes mencionadas.

Tabla 3.1: DCP Gestionar Yacimiento

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Seleccionar Opción.	**EC 1.1:** Mostrar pestañas.	Se muestran dos pestañas "Buscar" y "Adicionar" para la adición, modificación, eliminación y búsqueda de yacimientos.
SC 2: Adicionar Yacimiento.	**EC 2.1:** Selección de la opción "Adicionar".	El sistema muestra el formulario con los campos a llenar correspondiente a esta opción: *Provincia, Municipio, Nombre, Fecha de Estimación*. Así como un botón para ir a las páginas siguientes y adicionar las fuentes de agua que poseerá, así como los recursos disponibles de esta.
	EC 2.2: Introducir datos del yacimiento correctamente.	El sistema crea un nuevo yacimiento y lo muestra en la lista de los yacimientos existentes que contiene la página de la opción "Buscar".
	EC 2.3: Dejar campos en blanco o introducir datos del yacimiento de forma incorrecta.	El sistema marca en rojo los campos que no han sido introducidos o que presentan datos incorrectos.
	EC: 2.4: Existencia del yacimiento.	El sistema muestra un mensaje informando al usuario sobre la existencia de ese yacimiento "Este yacimiento ya ha sido adicionado anteriormente".
	EC: 2.5: Cancelar adición.	El sistema muestra un mensaje de confirmación "¿Está seguro que desea salir?". De manera que si es aceptado el mensaje, se muestra la pestaña "Buscar" con todo el listado de los yacimientos existentes.
SC 3: Buscar Yacimiento.	**EC 3.1:** Listar todos los yacimientos.	Se muestra un listado con todos los yacimientos.

	EC 3.2: Búsqueda con resultados.	Se permite la entrada de un valor de búsqueda y se deben mostrar los resultados obtenidos de acuerdo a dicho valor.
	EC 3.3: Búsqueda sin resultados.	Se permite la entrada de un valor de búsqueda, pero al no obtenerse resultados para este valor se debe mostrar un mensaje que informe de ello al usuario.
	EC 3.4: No se introduce texto alguno.	Se muestran todos los yacimientos existentes hasta ese momento.
	EC 3.5: Generar documento Word.	Se genera un documento Word con los resultados obtenidos en una búsqueda determinada.
	EC 3.6: Generar documento PDF.	Se genera un documento PDF con los resultados obtenidos en una búsqueda determinada.
	EC 3.7: Utilizar paginado.	Se debe poder avanzar o retroceder en la visualización de los resultados obtenidos.
SC 4: Modificar Yacimiento.	**EC 4.1:** Selección de la opción “Modificar”.	El sistema muestra el panel con un formulario que contiene los mismos datos descritos en el EC 2.1 del Yacimiento para que el usuario modifique el/los que desea.
	EC 4.2: El usuario modifica datos correctamente.	El sistema modifica los datos del yacimiento y muestra el listado de yacimientos.
	EC 4.3: El usuario modifica datos en los campos pero incorrectamente.	El sistema marca en rojo los campos que presentan datos incorrectos.
	EC 4.4: Cancelar la acción.	El sistema mantiene los datos del Yacimiento y vuelve a la pestaña “Buscar”.
SC 5: Eliminar Yacimiento.	**EC 5.1:** Eliminar datos.	El sistema muestra un mensaje de confirmación

		"¿Está seguro que desea eliminarlo?"
	EC 5.2: Aceptar.	El sistema elimina el yacimiento y actualiza la lista de estos.
	EC 5.3: Cancelar.	El sistema no elimina el yacimiento.

3.3.2 Gestionar Fuente de Agua

Descripción General: El CU comienza cuando el Administrador desea adicionar, buscar, modificar o eliminar una fuente de agua, ya sea un pozo o un manantial. El sistema brinda las opciones pertinentes. El Administrador selecciona la opción deseada, realiza la operación y termina el CU con la actualización de los datos de dicha fuente de agua.

Condiciones de Ejecución: El usuario debe estar autenticado y poseer los privilegios necesarios para realizar cualquiera de las acciones antes mencionadas y además debe existir al menos un Yacimiento.

Tabla 3.2: DCP Gestionar Fuente

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Seleccionar Opción.	**EC 1.1:** Mostrar pestañas.	Se muestran dos pestañas "Buscar" y "Adicionar" para la adición, modificación, eliminación y búsqueda de una fuente de agua.
SC 2: Adicionar Fuente de Agua.	**EC 2.1:** Selección de la opción "Adicionar".	El sistema muestra el formulario con los campos a llenar correspondiente a esta opción: *Yacimiento, Nombre, Número, Abatimiento máximo, tipo de fuente*. Así como un botón para ir a la página siguiente y adicionar los recursos disponibles de esta.
	EC 2.2: Introducir datos de la fuente de agua correctamente.	El sistema crea una nueva fuente de agua y la muestra en la lista de las fuentes de agua existentes que contiene la pestaña "Buscar".

	EC 2.3: Dejar campos en blanco o introducir datos de la fuente de agua de forma incorrecta.	El sistema marca en rojo los campos que no han sido introducidos o que presentan datos incorrectos.
	EC: 2.4: Existencia de la fuente de agua.	El sistema muestra un mensaje informando al usuario sobre la existencia de esa fuente de agua "Esta fuente de Agua ya ha sido adicionada anteriormente".
	EC: 2.5: Cancelar adición.	El sistema muestra un mensaje de confirmación "¿Está seguro que desea salir?". De manera que si es aceptado el mensaje, se muestra la pestaña "Buscar" con todo el listado de las fuentes de agua existentes.
SC 3: Buscar Fuente de Agua.	**EC 3.1:** Listar todas las fuentes de agua.	Se muestra un listado con todas las fuentes de agua.
	EC 3.2: Búsqueda con resultados.	Se permite la entrada de un valor de búsqueda y se deben mostrar los resultados obtenidos de acuerdo a dicho valor.
	EC 3.3: Búsqueda sin resultados.	Se permite la entrada de un valor de búsqueda, pero al no obtenerse resultados para este valor se debe mostrar un mensaje que informe de ello al usuario.
	EC 3.4: No se introduce texto alguno.	Se muestran todas las fuentes de agua existentes hasta ese momento.
	EC 3.5: Generar documento Word.	Se genera un documento Word con los resultados obtenidos en una búsqueda determinada.
	EC 3.6: Generar documento PDF.	Se genera un documento PDF con los resultados obtenidos en una búsqueda determinada.
	EC 3.7: Cantidad de resultados.	Se debe visualizar en cada pantalla la cantidad de resultados obtenidos, indicada por el usuario.

	EC 3.8: Utilizar paginado.	Se debe poder avanzar o retroceder en la visualización de los resultados obtenidos.
SC 4: Modificar Fuente de Agua.	**EC 4.1:** Selección de la opción "Modificar".	El sistema muestra el panel con un formulario que contiene los mismos datos descritos en el EC 2.1 de la Fuente de Agua para que el usuario modifique el/los que desea.
	EC 4.2: El usuario modifica datos correctamente.	El sistema actualiza los datos de la fuente de agua y muestra el listado con todas las fuentes de agua existentes.
	EC 4.3: El usuario modifica datos en los campos pero incorrectamente.	El sistema marca en rojo los campos que presentan datos incorrectos.
	EC 4.4: Cancelar la acción.	El sistema mantiene los datos de la fuente de agua y vuelve a la pestaña "Buscar".
SC 5: Eliminar Fuente de Agua.	**EC 5.1:** Eliminar datos.	El sistema muestra un mensaje de confirmación "¿Está seguro que desea eliminarla?"
	EC 5.2: Aceptar.	El sistema elimina la fuente de agua y actualiza la lista de estas.
	EC 5.3: Cancelar.	El sistema no elimina la fuente de agua.

3.3.3 Gestionar Recurso Disponible

Descripción General: El CU comienza cuando el Administrador desea adicionar, buscar, modificar o eliminar un recurso disponible. El sistema brinda las opciones pertinentes. El Administrador selecciona la opción deseada, realiza la operación y termina el CU con la actualización de los datos de dicho recurso.

Condiciones de Ejecución: El usuario debe estar autenticado y poseer los privilegios necesarios para realizar cualquiera de las acciones antes mencionadas y además debe existir al menos una fuente de agua.

Tabla 3.3: DCP Gestionar Recurso Disponible

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Seleccionar Opción.	**EC 1.1:** Mostrar pestañas.	Se muestran dos pestañas “Buscar” y “Adicionar” para la adición, modificación, eliminación y búsqueda de un recurso disponible.
SC 2: Adicionar Recurso Disponible.	**EC 2.1:** Selección de la opción “Adicionar”.	El sistema muestra el formulario con los campos a llenar correspondiente a esta opción: *Yacimiento, Fuente, Materia Prima, Mineralización, PH, Temperatura, Fecha, Medidos, Indicados, Inferidos, Sectores, Macroelementos (HCO3, SO4, Cl, Ca, Na, K, Mg), Microelemento (H2O, H2SiO3, NO3, NO2), Microbiológicos (CT, CF, EF, SD).*
	EC 2.2: Introducir datos del recurso disponible correctamente.	El sistema crea un nuevo recurso disponible y lo muestra en la lista de los recursos disponibles existentes que contiene la pestaña “Buscar”.
	EC 2.3: Dejar campos en blanco o introducir datos del recurso disponible de forma incorrecta.	El sistema marca en rojo los campos que no han sido introducidos o que presentan datos incorrectos.
	EC: 2.4: Cancelar adición.	El sistema muestra un mensaje de confirmación “¿Está seguro que desea salir?”. De manera que si es aceptado el mensaje, se muestra la pestaña “Buscar” con todo el listado de las fuentes de agua existentes.
SC 3: Buscar Recurso Disponible.	**EC 3.1:** Listar todas los recursos disponibles.	Se muestra un listado con todos los recursos disponibles.

	EC 3.2: Búsqueda con resultados.	Se permite la entrada de un valor de búsqueda y se deben mostrar los resultados obtenidos de acuerdo a dicho valor.
	EC 3.3: Búsqueda sin resultados.	Se permite la entrada de un valor de búsqueda, pero al no obtenerse resultados para este valor se debe mostrar un mensaje que informe de ello al usuario.
	EC 3.4: No se introduce texto alguno.	Se muestran todos los recursos disponibles existentes hasta ese momento.
	EC 3.5: Generar documento Word.	Se genera un documento Word con los resultados obtenidos en una búsqueda determinada.
	EC 3.6: Generar documento PDF.	Se genera un documento PDF con los resultados obtenidos en una búsqueda determinada.
	EC 3.7: Cantidad de resultados.	Se debe visualizar en cada pantalla la cantidad de resultados obtenidos, indicada por el usuario.
	EC 3.8: Utilizar paginado.	Se debe poder avanzar o retroceder en la visualización de los resultados obtenidos.
SC 4: Modificar Recurso Disponible.	**EC 4.1:** Selección de la opción “Modificar”.	El sistema muestra el panel con un formulario que contiene los mismos datos descritos en el EC 2.1 del Recurso Disponible para que el usuario modifique el/los que desea.
	EC 4.2: El usuario modifica datos correctamente.	El sistema muestra un mensaje de confirmación “¿Realmente desea modificar este Recurso Disponible?”.
	EC 4.3: Aceptar acción.	El sistema actualiza los datos de la fuente de agua y actualiza el listado en la pestaña “Buscar”.

	EC 4.4 El usuario modifica datos en los campos pero incorrectamente.	El sistema marca en rojo los campos que presentan datos incorrectos.
	EC 4.5: Cancelar la acción.	El sistema mantiene los datos de la fuente de agua y vuelve a la pestaña "Buscar".
SC 5: Eliminar Recurso Disponible.	**EC 5.1:** Eliminar datos.	El sistema muestra un mensaje de confirmación "¿Está seguro que desea eliminarla?"
	EC 5.2: Aceptar.	El sistema elimina el recurso disponible y actualiza la lista de estos.
	EC 5.3: Cancelar.	El sistema no elimina el recurso disponible.
SC 6: Ver detalles de Recurso Disponible.	**EC 6.1:** Selección de la opción "Ver detalles".	El sistema muestra una ventana de diálogo donde visualiza todos los campos que fueron descritos en EC 2.1 y que fueron entrados por el usuario para el Recurso Disponible seleccionado.

3.3.4 Generar Inventario

Descripción General: El CU comienza cuando el Administrador desea generar un inventario. El sistema brinda las opciones pertinentes. El Administrador selecciona la opción deseada, y el sistema genera el inventario deseado.

Condiciones de Ejecución: El usuario debe estar autenticado y poseer los privilegios necesarios para realizar cualquiera de las acciones antes mencionadas.

Tabla 3.4: DCP Generar Inventario

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Seleccionar Opción.	**EC 1.1:** Selección de la opción "Reportes".	El sistema despliega el menú para que el usuario seleccione el inventario que desea generar.

SC 2: Generar Inventario para la ACE.	**EC 2.1:** Selección de la opción "Generar Inventario para la ACE".	El sistema genera el Inventario para la ACE y brinda la opción de Exportarlo a Formato.
	EC 2.2: Generar documento Word.	Se genera un documento Word con el Inventario para la ACE.
	EC 2.3: Generar documento PDF.	Se genera un documento PDF con el Inventario para la ACE.
SC 3: Generar Inventario para los Concesionarios.	**EC 3.1:** Selección de la opción "Generar Inventario para los Concesionarios".	El sistema muestra un formulario donde el usuario podrá seleccionar el concesionario deseado.
	EC 3.2: Selección del concesionario.	El sistema genera el Inventario para dicho concesionario y brinda la opción de Exportarlo a Formato.
	EC 3.3: No se selecciona concesionario alguno.	El sistema no genera ningún inventario.
	EC 3.4: Generar documento Word.	Se genera un documento Word con el Inventario para la ACE.
	EC 3.5: Generar documento PDF.	Se genera un documento PDF con el Inventario para la ACE.

3.3.5 Realizar Búsqueda Avanzada

El CU comienza cuando el Consultor desea realizar alguna búsqueda por determinados criterios. El sistema brinda las opciones pertinentes al Consultor. El Consultor selecciona la opción deseada y termina el CU mostrando el resultado de la búsqueda.

Condiciones de Ejecución: El usuario debe estar autenticado y poseer los privilegios necesarios para realizar cualquiera de las acciones antes mencionadas.

Tabla 3.5: DCP Búsqueda Avanzada 1

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Seleccionar Opción.	**EC 1.1** Selección de la opción "Consultas".	El sistema muestra las siguientes opciones de búsqueda: • Por Yacimiento • Por Fuente de Agua • Por Recurso • Por Control Externo • Por Inventario • Por Registros Legales
SC 2: Por Yacimiento.	**EC 2.1:** Selección de la opción "Por Yacimiento".	El sistema muestra un formulario con los siguientes criterios de búsqueda: *Provincia, Municipio, Nombre, Fecha de Estimación, Cantidad de Fuentes de Agua, Tipo de Materia Prima, Concesionario.*
	EC 2.2: Búsqueda con resultados.	El sistema visualiza el resultado de dicha búsqueda en un listado teniendo en cuenta el criterio y datos del usuario, brindando la posibilidad de Exportar a Formato y Ver Detalles.
	EC 2.3: Búsqueda sin resultados.	El sistema muestra un mensaje informado al usuario que no existen elementos para dicha búsqueda.
	EC 2.4: Generar documento Word.	Se genera un documento Word con el resultado de la búsqueda realizada.
	EC 2.5: Generar documento PDF.	Se genera un documento PDF con el resultado de la búsqueda realizada.
	EC 2.6: Ver detalles.	Se muestra una ventana de diálogo con los datos del yacimiento deseado.

SC 3: Por Fuente de Agua.	**EC 3.1:** Selección de la opción "Por Fuente de Agua".	El sistema muestra un formulario con los siguientes criterios de búsqueda: *Provincia, Municipio, Yacimiento, Concesionario, Tipo de Materia Prima, Abatimiento Máximo.*
	EC 3.2: Búsqueda con resultados.	El sistema visualiza el resultado de dicha búsqueda en un listado teniendo en cuenta el criterio y datos del usuario, brindando la posibilidad de Exportar a Formato y Ver Detalles.
	EC 3.3: Búsqueda sin resultados.	El sistema muestra un mensaje informado al usuario que no existen elementos para dicha búsqueda.
	EC 3.4: Generar documento Word.	Se genera un documento Word con el resultado de la búsqueda realizada.
	EC 3.5: Generar documento PDF.	Se genera un documento PDF con el resultado de la búsqueda realizada.
	EC 3.6: Ver detalles.	Se muestra una ventana de diálogo con los datos de la fuente de agua deseada.
SC 4: Por Recurso.	**EC 4.1:** Selección de la opción "Por Recurso".	El sistema muestra un formulario con los siguientes criterios de búsqueda: *Provincia, Municipio, Materia Prima, Yacimiento, Pozo, Mineralización, PH, Temperatura, Parámetros de Calidad (Macroelemento, Microelemento, Microbiológico).*
	EC 4.2: Selección del tipo de recurso.	El sistema muestra en el mismo formulario otros criterios correspondiente a su selección: • Si selecciona Recurso Disponible, muestra además en el formulario: Caudal Medido, Indicado e Inferido. • Si selecciona Recurso de Explotación, muestra además en el formulario: Caudal

		Probado, y probable. • Si selecciona Recurso en Explotación, muestra además en el formulario: Caudal Probado.
	EC 4.3: Búsqueda con resultados.	El sistema visualiza el resultado de dicha búsqueda en un listado teniendo en cuenta el criterio y datos del usuario, brindando la posibilidad de Exportar a Formato y Ver Detalles.
	EC 4.4: Búsqueda sin resultados.	El sistema muestra un mensaje informado al usuario que no existen elementos para dicha búsqueda.
	EC 4.5: Generar documento Word.	Se genera un documento Word con el resultado de la búsqueda realizada.
	EC 4.6: Generar documento PDF.	Se genera un documento PDF con el resultado de la búsqueda realizada.
	EC 4.7: Ver detalles.	Se muestra una ventana de diálogo con los datos del recurso deseado.
SC 5: Por Control Externo.	**EC 5.1:** Selección de la opción "Por Control Externo".	El sistema muestra un formulario con los siguientes criterios de búsqueda: *Provincia, Provincia, Municipio, Materia Prima, Yacimiento, Mineralización, PH, Temperatura, Parámetros de Calidad (Macroelemento, Microelemento, Microbiológico), Desviación, Fecha.*
	EC 5.2: Búsqueda con resultados.	El sistema visualiza el resultado de dicha búsqueda en un listado teniendo en cuenta el criterio y datos del usuario, brindando la posibilidad de Exportar a

		Formato y Ver Detalles.
	EC 5.3: Búsqueda sin resultados.	El sistema muestra un mensaje informado al usuario que no existen elementos para dicha búsqueda.
	EC 5.4: Generar documento Word.	Se genera un documento Word con el resultado de la búsqueda realizada.
	EC 5.5: Generar documento PDF.	Se genera un documento PDF con el resultado de la búsqueda realizada.
	EC 5.6: Ver detalles.	Se muestra una ventana de diálogo con los datos del control externo deseado.
SC 6: Por Inventario.	**EC 6.1:** Selección de la opción "Por Inventario".	El sistema muestra un formulario con los siguientes criterios de búsqueda: *Tipo de Inventario, Año, Materia Prima, Recursos (caudal) disponible por cada materia prima, Recursos (caudal) de explotación por cada materia prima, cantidad de Recursos en Explotación.*
	EC 6.2: Búsqueda con resultados.	El sistema visualiza el resultado de dicha búsqueda en un listado teniendo en cuenta el criterio y datos del usuario, brindando la posibilidad de Exportar a Formato y Ver Detalles.
	EC 6.3: Búsqueda sin resultados.	El sistema muestra un mensaje informado al usuario que no existen elementos para dicha búsqueda.
	EC 6.4: Generar documento Word.	Se genera un documento Word con el resultado de la búsqueda realizada.
	EC 6.5: Generar documento PDF.	Se genera un documento PDF con el resultado de la búsqueda realizada.

	EC 6.6: Ver detalles.	Se muestra una ventana de diálogo con los datos del inventario deseado.
SC 7: Por Registros Legales.	**EC 7.1:** Selección de la opción "Por Registros Legales".	El sistema muestra un formulario con los siguientes criterios de búsqueda: *Tipo de registro.*
	EC 7.2: Búsqueda con resultados.	El sistema visualiza el resultado de dicha búsqueda en un listado teniendo en cuenta el criterio y datos del usuario, brindando la posibilidad de Exportar a Formato y Ver Detalles.
	EC 7.3: Búsqueda sin resultados.	El sistema muestra un mensaje informado al usuario que no existen elementos para dicha búsqueda.
	EC 7.4: Generar documento Word.	Se genera un documento Word con el resultado de la búsqueda realizada.
	EC 7.5: Generar documento PDF.	Se genera un documento PDF con el resultado de la búsqueda realizada.
	EC 7.6: Ver detalles.	Se muestra una ventana de diálogo con los datos del registro legal deseado.

3.4 Conclusiones Parciales

En este capítulo se han diseñado los casos de prueba necesarios para validar y medir el funcionamiento del módulo Inventario Nacional de Recursos de Aguas Minerales. Los mismos como se expresó anteriormente, no garantizan que dicho módulo quede libre de defectos sino que permiten detectarlos y corregirlos. De manera que la aplicación de los casos de prueba permitirán conocer hasta qué punto el sistema cumple con los requerimientos especificados por el cliente.

CONCLUSIONES

El presente trabajo concluye con la generación de la documentación técnica como resultado del desarrollo de las actividades correspondientes al Rol de Implementador del Módulo Inventario Nacional de Recursos de Aguas Minerales que forma parte del Sistema de Gestión de Datos Geológicos; alcanzándose así el resultado esperado. El mismo estuvo guiado por un conjunto de aspectos que se detallan a continuación:

1. Las herramientas, tecnologías, lenguajes y gestores utilizados están en correspondencia con los principios tecnológicos impulsados por la Universidad.
2. El diseño propuesto por el analista es claro, simple, y orientado al implementador.
3. Los estilos y estándares de implementación potencian la organización, la legibilidad y reusabilidad del código implementado.
4. La utilización de implementaciones, componentes y módulos permitieron reducir en gran medida, el tiempo empleado en la escritura del código fuente, así estandarizar la propuesta con el resto de los módulos que componen el proyecto.
5. La aplicación desarrollada se rige por normas de diseño y codificación establecidas en el proyecto Sistema de Gestión de Datos Geológicos.
6. La elaboración de los diseños de casos de prueba permiten detectar errores tempranos, agilizar y simplificar el flujo de trabajo de pruebas.

RECOMENDACIONES

Una vez concluida la presente investigación se recomienda lo siguiente:

1. Que se realice la validación del sistema mediante el uso de varios tipos de pruebas.
2. Que la aplicación sea utilizada en la ONRM para el mantenimiento y control de los recursos de aguas minerales.
3. Agregar la funcionalidad de graficar parámetros cuantitativos y cualitativos.
4. Implementar los requerimientos funcionales correspondientes a la importación de archivos XML.

REFERENCIAS BIBLIOGRÁFICAS

- Alvarez, Miguel Angel. 2005. desarrolloweb.com. [En línea] 2005. [Citado el: 20 de 10 de 2009.] http://www.desarrolloweb.com/manuales/20. B82780297 .
- Alvarez, Miguel Angel 2009. dsarrolloweb.com. [En línea] 25 de 4 de 2009. [Citado el: 1 de 1 de 2010.]
- Alvarez, Sara. 2007. desarrolloweb.com. [En línea] 31 de 7 de 2007. [Citado el: 1 de 2 de 2010.]
- Cruz, Luis Miguel de la. 2000. *Introducción a la Programación Orientada a Objetos con C++.* 2000.
- Cuenca, Carlos Luis. 2003. desarrolloweb.com. [En línea] 20 de 3 de 2003. [Citado el: 5 de 12 de 2009.] http://www.desarrolloweb.com/articulos/1112.php.
- Espinoza, M.C. José Martín Olguín. 2009. UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA . . [En línea] 2009. [Citado el: 25 de 3 de 2010.] http://yaqui.mxl.uabc.mx/~molguin/as/RUP.htm.
- Lobos, Maria Elena de. 2006. mailxmail.com. [En línea] 12 de 4 de 2006. [Citado el: 15 de 11 de 2009.] http://www.mailxmail.com/curso-aprende-programar/concepto-lenguaje-programacion.
- Musciano, Chuck y Kemedy, Bill. 1999. *HTML La guía completa.* Mexico : McGRAW-HILL INTERAMERICANA EDITORES, 1999. 1-56592-235.
- Pérez, Javier Eguíluz. 2009. *Introducción a CSS.* 2009.
- Potencier, Fabien y Zaninotto, François. 2008. *Symfony la guía definitiva.* 2008.

BIBLIOGRAFÍA

- Alvarez, Miguel Angel. 2005. desarrolloweb.com. [En línea] 2005. [Citado el: 20 de 10 de 2009.] http://www.desarrolloweb.com/manuales/20. B82780297 .
- Alvarez, Miguel Angel. 2009. dsarrolloweb.com. [En línea] 25 de 4 de 2009. [Citado el: 1 de 1 de 2010.]
- Alvarez, Sara. 2007. desarrolloweb.com. [En línea] 31 de 7 de 2007. [Citado el: 1 de 2 de 2010.]
- Cruz, Luis Miguel de la. 2000. *Introducción a la Programación Orientada a Objetos con C++.* 2000.
- Cuenca, Carlos Luis. 2003. desarrolloweb.com. [En línea] 20 de 3 de 2003. [Citado el: 5 de 12 de 2009.] http://www.desarrolloweb.com/articulos/1112.php.
- Espinoza, M.C. José Martín Olguín. 2009. UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA . . [En línea] 2009. [Citado el: 25 de 3 de 2010.] http://yaqui.mxl.uabc.mx/~molguin/as/RUP.htm.
- Lanzillotta, Analía. 2004. Master Magazine. [En línea] 2004. [Citado el: 2 de 11 de 2009.] http://www.mastermagazine.info/termino/5560.php.
- Lobos, Maria Elena de. 2006. mailxmail.com. [En línea] 12 de 4 de 2006. [Citado el: 15 de 11 de 2009.] http://www.mailxmail.com/curso-aprende-programar/concepto-lenguaje-programacion.
- Musciano, Chuck y Kemedy, Bill. 1999. *HTML La guía completa.* Mexico : McGRAW-HILL INTERAMERICANA EDITORES, 1999. 1-56592-235.
- Pérez, Javier Eguíluz. 2009. *Introducción a CSS.* 2009.
- Pérez, Javier Eguíluz. 2009. *Introducción a XHTML.* 2009.
- Potencier, Fabien y Zaninotto, François. 2008. *Symfony la guía definitiva.* 2008.
- S, Christian Van Der Henst. 2004. [En línea] 2004. [Citado el: 1 de 11 de 2009.]