Universidad de las Ciencias Informáticas

Facultad 9

Oficina Nacional de Recursos Minerales

# SISTEMA DE GESTIÓN DE DATOS GEOLÓGICOS

# MÓDULO: REGISTRO MINERO. ROL: IMPLEMENTADOR

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

**Autor: Joel Macías Roque.**

**Tutor: Ing. Vladimir Martell Fernández.**

Ciudad de la Habana, Junio de 2010
"Año 52 de la Revolución"

# *Dedicatoria*

*A mi mamá que me dio la vida y desde entonces siempre me ha apoyado y alentado a seguir adelante aunque todo esté en mi contra. Por creer en mí y ayudarme a ser cada día mejor persona.*

*A mis hermanos, que son de las personas más importantes en mi vida.*

*A toda mi familia, que siempre han estado pendientes de mí, en las buenas en las no tan buenas.*

*A mis amigos, por el apoyo incondicional de siempre.*

*A todos ustedes, lectores de estas páginas...*

# *Agradecimientos*

*Agradezco a mi familia por todo el apoyo durante toda mi vida, especialmente a mi madre, a mi padre y a mi hermano Josiel por todo el cariño, y sacrificio.*

*Agradezco además a todos mis grandes amigos, los que están cerca y los que no, por toda la ayuda, por compartir buenos y malos momentos.*

*Muchas gracias a Zuleira, mi novia, por su amor incondicional y su confianza durante todo este tiempo.*

*Agradezco particularmente a mi tutor y amigo Ing. Vladimir Martell Fernández, gracias por la confianza.*

*Agradezco al tribunal de tesis por sus precisas acotaciones que le dieron a este trabajo mayor calidad y rigor científico.*

*Muchas gracias también a todas las personas del Laboratorio 302 del Departamento Geoinformática, especiales para todo el equipo de desarrollo del SGDG, por todas las horas de trabajo que pasamos juntos.*

*Joel Macías Roque*

# *Declaración de Autoría*

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas y 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 _____.

| | |
|---|---|
| _____ | _____ |
| Joel Macías Roque | Ing. Vladimir Martell Fernández |

## Datos del Tutor:

**Nombre y apellidos:** Ing. Vladimir Martell Fernández.

**Correo electrónico:** vmartell@uci.cu

**Categoría docente:** Instructor Recién Graduado.

**Año de graduación:** 2008.

**Profesión:** Ingeniero en Ciencias Informáticas.

**Breve descripción:** Graduado en la Universidad de las Ciencias Informáticas. Actualmente profesor del Departamento Técnicas de Programación y 2do Jefe del Departamento Geoinformática del Centro de Desarrollo “Geoinformática y Señales Digitales” de la facultad 9.

En Cuba, la informatización de la sociedad se define como el proceso de utilización ordenada y masiva de las Tecnologías de la Información y las Comunicaciones para satisfacer las necesidades de información y conocimiento de todas las personas y esferas de la sociedad. Este proceso busca lograr más eficacia y eficiencia, que permitan una mayor generación de riquezas y hagan sustentable el aumento sistemático de la calidad de vida de los cubanos.

El sector geólogo-minero del país no se encuentra exento de este proceso de informatización, jugando un papel destacado la Oficina Nacional de Recursos Minerales, precisamente la presente investigación consiste en la realización de las actividades al rol de implementador para lograr la funcionalidad del módulo Registro Minero que forma parte de la aplicación Sistema de Gestión de Datos Geológicos, que se desarrolla con el objetivo de brindar, a la mencionada entidad cubana, un sistema de gestión de tipo web para manipular de forma segura la información que se genera y transfiere en la Oficina Nacional de Recursos Minerales.

**PALABRAS CLAVES**

Implementador, Sistema de Gestión de Datos Geológicos, Registro Minero.

# Índice de Contenido


INTRODUCCIÓN ---- 1
CAPÍTULO 1 ---- 6
1.1 Introducción ---- 6
1.2 Paradigmas de la Programación ---- 6
1.3 Lenguajes de Programación. ---- 7
1.3.1 HTML ---- 8
1.3.2 XML ---- 9
1.3.3 Javascript ---- 9
1.3.4 CSS ---- 10
1.3.5 PHP ---- 11
1.4 Sistemas Gestores de Bases de Datos ---- 12
1.4.1 PostgreSQL ---- 12
1.5 Plataformas, Herramientas de desarrollo y Frameworks ---- 13
1.5.1 Servidor HTTP Apache ---- 13
1.5.2 AJAX ---- 14
1.5.3 Frameworks de desarrollo ---- 15
1.5.3.1 Symfony como framework de desarrollo para aplicaciones web ---- 16
1.5.3.2 JQuery como framework para Javascript ---- 17
1.5.4 Visual Paradigm como herramienta CASE ---- 18
1.5.5 Zend Studio for Eclipse como IDE para PHP ---- 18
1.6 Metodologías de Desarrollo ---- 19
1.6.1 Rational Unified Process ---- 19
1.6.1.1 Caracterización del rol implementador según RUP ---- 20
1.7 Conclusiones parciales ---- 21
2.1 Introducción ---- 22
2.2 Valoración Crítica del diseño propuesto por el analista ---- 22
2.3 Análisis de posibles implementaciones, componentes o módulos ya existentes. ---- 26
2.4 Descripción de las nuevas clases u operaciones fundamentales. ---- 27

2.5 Estilos de Programación. 30
2.6 Estándares de Codificación 33
2.7 Conclusiones parciales 38
3.1 Introducción 39
3.2 Descripción general de las pruebas 39
3.3 Diseño de las pruebas que permitan validar la solución propuesta 40
3.3.1 Gestionar Obligaciones 40
Escenarios a probar en el Caso de Uso 40
Descripción de las variables 42
SC1: Buscar Obligaciones 43
SC: 2 Insertar Obligaciones 44
SC: 3 Modificar Obligaciones 44
SC: 4 Eliminar Obligaciones 45
3.3.2 Gestionar Medidas 45
Escenarios a probar en el Caso de Uso 45
Descripción de las variables 47
SC1 Buscar Medidas 48
SC2 Insertar Medidas 48
SC3 Modificar Medidas 50
SC4 Eliminar Medidas 51
3.4 Conclusiones Parciales 53
CONCLUSIONES 54
RECOMENDACIONES 55
REFERENCIAS BIBLIOGRÁFICAS 56
BIBLIOGRAFÍA 58
GLOSARIO DE TÉRMINOS 61
ANEXOS 63

## TABLAS

Tabla 1: Estándar de Codificación del SGDG .....69
Tabla 2: Componentes visuales .....69

## FIGURAS

Figura 1: Tecnologías que forman AJAX.....14
Figura 2: Propuesta de Asignación .....20
Figura 3: Funcionamiento del patrón MVC .....23
Figura 4: Modelo de Diseño Caso de Uso Gestionar Medidas .....24
Figura 5: Modelo Clásico de aplicaciones web.....63
Figura 6: Modelo AJAX de aplicaciones web .....64

# INTRODUCCIÓN

Desde la década de los ’90 a las Tecnologías de la Información y las Comunicaciones (en lo adelante TIC) se les atribuye el protagonismo de los grandes cambios y transformaciones de la sociedad, tanto es así que a los tiempos actuales se les denomina “Sociedad de la Información o del Conocimiento”. La presencia de las TIC en la mayoría de las actividades humanas es innegable y su constante utilización ha provocado un cambio positivo en la sociedad y en consecuencia una mejora en la calidad de vida de los ciudadanos.

Las TIC, siguiendo el ritmo de los continuos avances científicos, contribuyen a la rápida obsolescencia de los conocimientos y a la aparición de nuevos valores, provocando continuas transformaciones en las estructuras económicas, sociales y culturales, e incidiendo en casi todos los aspectos de nuestra vida: el acceso al mercado, la gestión económica, el diseño industrial y artístico, el ocio, la comunicación, la información, la organización de las empresas e instituciones, sus métodos y actividades, la calidad de vida, la educación, la medicina. Su impacto en todas las esferas de la sociedad hace que sea cada vez más difícil poder actuar eficientemente prescindiendo de ellas. (1)

En Cuba, la informatización de la sociedad se define como el proceso de utilización ordenada y masiva de las TIC para satisfacer las necesidades de información y conocimiento de todas las personas y esferas de la sociedad. Este proceso busca lograr más eficacia y eficiencia, que permitan una mayor generación de riquezas y hagan sustentable el aumento sistemático de la calidad de vida de los cubanos. (2)

El sector geólogo-minero del país no se encuentra exento de este proceso de informatización, jugando un papel destacado la **Oficina Nacional de Recursos Minerales** (en lo adelante ONRM), entidad responsable de garantizar el aprovechamiento racional de los recursos minerales del país y ejercer con eficiencia, rigor técnico, y responsabilidad el control estatal sobre la geología, minería y petróleo en Cuba.

En la ONRM la mayoría de las actividades se realizan manualmente o con la ayuda de alguna herramienta informática como Microsoft Word, Microsoft Excel y Microsoft Access, debido al gran cúmulo de información se dificulta en gran medida la ejecución de sus actividades y como la mayor parte del trabajo se realiza manualmente la información se puede encontrar descentralizada, redundante y duplicada.

Además, muchos de los datos con los que se trabaja en dicha entidad se encuentran hoy en formato duro, propiciando esto, pérdida y deterioro, sin mencionar que no existen los mecanismos adecuados de control para verificar quién puede o no acceder a cierta información.

En el proceso para informatizar el sector geólogo-minero, se crea, en el año 2006, el Programa Nacional de Informatización del Conocimiento Geológico (en lo adelante PNICG). Como parte de este programa la ONRM y la Universidad de las Ciencias Informáticas (en lo adelante UCI), unen empeños con el objetivo de implementar herramientas informáticas robustas, debidamente protegidas y aseguradas, que sigan estándares informáticos, de calidad, legales y normativos de las geo-ciencias para la conservación y administración del conocimiento geológico.

Con el fin de lograr el cumplimiento del objetivo que se plantearon ambas entidades se crea en la facultad 9 de la UCI dentro del Centro de Desarrollo Geoinformática y Señales Digitales (en lo adelante GEySED), específicamente en el Departamento Geoinformática, el proyecto productivo "**Sistema de Gestión de Datos Geológicos**" (en lo adelante SGDG) con el propósito de desarrollar un sistema de gestión de tipo web para la ONRM que permita manipular de forma segura la información que se genera, manipula o transfiere en la mencionada entidad cubana. (3)

Dentro de cualquier proyecto que tenga planteado la realización de una aplicación informática el implementador juega un papel sumamente importante, porque es precisamente el *desarrollador o programador*[1] el encargado de definir y mantener el código fuente logrando hacer funcional el diseño planteado por el diseñador.

El implementador es responsable de los componentes de desarrollo y de prueba, de acuerdo con los estándares aprobados por el proyecto, para la integración en subsistemas más grandes. Cuando los componentes de prueba, como controladores o fragmentos para simulación, deben crearse para dar soporte a las pruebas, el implementador también es responsable del desarrollo y las pruebas de los componentes de prueba y los subsistemas correspondientes. (4)

[1] Nombres con lo que también se denomina al implementador de software

Actualmente el proyecto SGDG se encuentra en fase de construcción, siendo el principal objetivo obtener un producto listo para su utilización, que haya rebasado algunas pruebas y con un manual para el usuario del Sistema.

SGDG cuenta con los modelos de análisis, diseño y prototipos de una aplicación informática para llevar a cabo todos los procesos que tienen lugar en la ONRM, pero la misma no ha sido implementada hasta la fecha actual. Una vez constatada la problemática existente, se deriva entonces el siguiente **problema a resolver:** ¿Cómo lograr la funcionalidad del diseño propuesto para el módulo Registro Minero que forma parte del SGDG? Para dar solución al mismo, el **objeto de estudio** se define como el proceso de implementación de los sistemas de gestión de información, siendo el **campo de acción** la implementación del módulo Registro Minero que forma parte del SGDG.

Como **objetivo general** se pretende elaborar la documentación técnica de la implementación del diseño propuesto correspondiente al módulo Registro Minero que forma parte del SGDG.

Debido a lo anterior se expone 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 Registro Minero que forma parte del SGDG.

Para lograr el objetivo expuesto anteriormente se desarrollarán las siguientes tareas:

1. Desarrollar habilidades en el uso de la herramienta Case Visual Paradigm.
2. Desarrollar habilidades en el uso del lenguaje de programación PHP, del framework Symfony y del gestor de Bases de Datos PostgreSQL.
3. Argumentar el uso de PHP, PostgreSQL, Symfony.
4. Caracterizar el rol Implementador según la metodología seleccionada.
5. Caracterizar el diseño propuesto.
6. Seleccionar y aplicar un estándar de codificación adecuado.
7. Implementar los Casos de Uso propuestos por el analista.
8. Documentar los principales algoritmos codificados.
9. Validar la Solución Propuesta.

Una vez realizadas las tareas de investigación se espera el siguiente resultado:

- La documentación técnica producto del desarrollo del Rol de Implementador del Módulo Registro Minero que forma parte del SGDG.

Además, la ONRM tendrá a su disposición un sistema para el control legal del proceso de los concesionarios y sus concesiones, lo que consecuentemente propiciará:

- Un aumento en la capacidad de respuesta de la ONRM a la sociedad cubana.
- Mayor satisfacción en los usuarios de la ONRM.
- Centralización y organización de la información que se genera, manipula o transfiere en la ONRM, ofreciendo además un acceso limitado a esta información.

En toda investigación científica se utilizan métodos científicos de investigación debido a que estos son*: la forma de abordar la realidad, de estudiar la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia y sus relaciones*. (5) Estos métodos científicos se pueden clasificar fundamentalmente en:

1. **Métodos Empíricos:** Permiten la obtención y elaboración de los datos empíricos y el conocimiento de los hechos fundamentales que caracterizan los fenómenos. (5)
2. **Métodos Teóricos:** Se utilizan en la construcción y desarrollo de la teoría científica y en el enfoque general para abordar los problemas de las ciencias. (5)

Durante el transcurso y desarrollo de esta investigación científica, se utilizarán algunos **métodos teóricos** y dentro de estos métodos específicamente los que se detallan a continuación**:**

1. **Análisis y la síntesis:** Para descomponer el problema de investigación en partes, para un mejor entendimiento de la situación y luego poder sintetizarlos para la confección de la solución propuesta.
2. **Histórico y Lógico:** Para el estudio de los trabajos e investigaciones anteriores y tenerlo como base para esta investigación.

El presente trabajo está dividido en 3 capítulos:

**Capítulo 1:** Fundamentación Teórica. En este capítulo se aborda todo lo referido a las principales tecnologías existentes que se emplean para llevar a cabo la solución de esta investigación, así como a la descripción de las características de las plataformas y framework fundamentales que se utilizarán, unidos a las principales herramientas de desarrollo.

**Capítulo 2:** Descripción y análisis de la solución propuesta. En el presente capítulo se hace una valoración de la solución propuesta, así como una crítica al diseño propuesto por el analista del sistema.

**Capítulo 3:** Validación de la solución propuesta. En el mismo se analiza la solución dada, se corrobora que cumpla con los requerimientos.

# CAPÍTULO 1

## 1.1 Introducción

En este capítulo se abordarán y describirán todas las herramientas, tecnologías, así como los conceptos relacionados con estas, para lograr construir la solución propuesta. Se abordarán, además, los lenguajes de programación utilizados para la investigación y elementos referentes a los paradigmas de la programación.

## 1.2 Paradigmas de la Programación

Un paradigma de programación representa un enfoque o filosofía para la construcción de software. Es el criterio de muchos que uno no es mejor que otro sino que cada uno tiene ventajas y desventajas, además también hay situaciones en que un paradigma se ajusta mucho más que otro.

Los paradigmas de programación más comunes son los siguientes:

**Paradigmas procedimentales u operacionales,** es tal vez el más conocido y utilizado en el proceso de programación, donde los programas se desarrollan a través de procedimientos. Pascal C y BASIC son tres de los lenguajes imperativos más importantes. La palabra latina *imperare* significa "dar instrucciones". El paradigma se inició al principio del año 1950 cuando los diseñadores reconocieron que las variables y los comandos o instrucciones de asignación constituían una simple pero útil abstracción del acceso a memoria y actualización del conjunto de instrucciones máquina. Debido a la estrecha relación con la arquitectura de la máquina, los lenguajes de programación imperativa pueden ser implementados muy eficientemente, al menos en principio. (6)

**La programación funcional** se caracteriza por el uso de expresiones y funciones. Un programa dentro del paradigma funcional, es una función o un grupo de funciones compuestas por funciones más simples estableciéndose que una función puede llamar a otra, o el resultado de una función puede ser usado como argumento de otra función. El lenguaje por excelencia ubicado dentro de este paradigma es el LISP. (6)

**El paradigma declarativo o paradigma de programación lógica** se basa en el hecho que un programa implementa una relación antes que una correspondencia. Debido a que las relaciones son más generales que las correspondencias (identificador - dirección de memoria), la programación lógica es potencialmente de más alto nivel que la programación funcional o la imperativa. El lenguaje más popular enmarcado dentro de este paradigma es el lenguaje PROLOG. El auge del paradigma declarativo se debe a que el área de la lógica formal de las matemáticas ofrece un sencillo algoritmo de resolución de problemas adecuado para, usarse en un sistema de programación declarativo de propósito general. (6)

**El paradigma orientado a objetos**, se basa en los conceptos de objetos y clases de objetos. Un objeto es una variable equipada con un conjunto de operaciones que le pertenecen o están definidas para ellos. El paradigma orientado a objetos actualmente es el paradigma más popular. Una de las bondades importantes de los lenguajes orientados a objetos es que las definiciones de los objetos pueden usarse una y otra vez para construir múltiples objetos con las mismas propiedades o modificarse para construir nuevos objetos con propiedades similares pero no exactamente iguales. (6)

La Programación Orientada a Objetos (en lo adelante POO) no es un lenguaje de programación, puede aplicarse a cualquier lenguaje, y de hecho hoy en día está disponible en mayor o menor medida en todos los lenguajes tradicionales (C se ha convertido en C++, Pascal en Delphi, VB incorpora parte de la POO) y no aparece un lenguaje nuevo sin que incluya OOP (como es el caso de Java). (7)

## 1.3 Lenguajes de Programación.

Las máquinas en general, y las computadoras en particular, necesitan de un lenguaje propio para poder interpretar las instrucciones que se les dan y para poder controlar su comportamiento. Ese lenguaje que permite esta relación con las computadoras es el lenguaje de programación, algunos de ellos son Visual Basic, Java, PHP, Cobol, C++.

El lenguaje de programación está conformado por una serie de reglas sintácticas y semánticas que son utilizadas por el programador y a través de las cuales creará un programa o subprograma. Todas estas instrucciones escritas por el desarrollador y que forman dicho programa es a lo que se denomina código fuente. (8)

Los lenguajes de programación pueden ser interpretados (que usan un intérprete) o compilados (que usan un compilador).

La ejecución de un programa con compilador requiere de dos etapas:

1. Traducir el programa simbólico a código máquina.
2. Ejecución y procesamiento de los datos.

Los lenguajes de programación interpretados utilizan un programa intérprete o traductor, el cual analiza directamente la descripción simbólica del programa fuente y realiza las instrucciones dadas.

La ventaja del proceso intérprete es que no necesita de dos fases para ejecutar el programa, sin embargo su inconveniente es que la velocidad de ejecución es más lenta ya que debe analizar e interpretar las instrucciones contenidas en el programa fuente.

### 1.3.1 HTML

HTML[1], lenguaje usado para escribir y publicar documentos en la *World Wide Web* (www). Es una aplicación de la ISO Standard 8879:1986 (SGML, *Standard Generalized Markup Language*). (9)

Una forma más informal de definirlo es diciendo que es el lenguaje con el que se escriben todas las páginas web en Internet. Este lenguaje fue originalmente concebido para el intercambio de documentos de corte científicos y técnicos.

HTML utiliza etiquetas, que consisten en breves instrucciones de comienzo y final, mediante las cuales se determina la forma en la que debe aparecer el texto en el navegador, así como también las imágenes y los demás elementos, en la pantalla del ordenador. Toda etiqueta se identifica porque está encerrada entre los signos menor que y mayor que (<,>), y algunas tienen atributos que pueden tomar algún valor.

[1] Hypertext Markup Language

HTML más que un lenguaje se ha convertido en un estándar aceptado en todo el mundo por lo que una página HTML se puede visualizar en una navegador de cualquier Sistema Operativo, sus normas las define el *World Wide Web Consortium* (en lo adelante W3C).

### 1.3.2 XML

XML[1] es un lenguaje extensible de etiquetas desarrollado por el W3C. XML no es realmente un lenguaje en particular, sino una forma de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, *Scalable Vector Graphics* (SVG) y MathML.

XML además de ser pensado para aplicaciones en INTERNET, se propone como un estándar para el intercambio de información entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil. XML ofrece además validación de los datos y realiza el recorrido de las estructuras, de manera que el implementador no tiene que preocuparse por estas cuestiones.

### 1.3.3 Javascript

Es un lenguaje interpretado, se utiliza, principalmente, para el desarrollo de interfaces mejoradas en aplicaciones web y la creación de páginas web dinámicas, permitiendo acceder a los elementos de una página incorporándoles efectos y animaciones. Todos los navegadores actuales interpretan el código Javascript integrado en las páginas web. Este lenguaje está equipado con una implementación del *Document Object Model* (en lo adelante DOM) lo que le facilita la interacción con los objetos en las páginas web.

El código Javascript se ejecuta en el cliente por lo que no se le hacen solicitudes innecesarias al servidor. Además es conveniente el uso de Javascript para la validación de los datos de los formularios del lado del

[1] Extensible Markup Language

cliente, y para la creación de interfaces de usuarios más complejas e interactivas. La sintaxis de Javascript es muy parecida a C++ y a Java, pero es mucho más fácil de aprender.

### 1.3.4 CSS

CSS[1] es un lenguaje para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML) (10). El W3C es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores. El principal objetivo del uso de CSS es separar la estructura del documento de su presentación y de esta forma aumentar la organización del código y el riesgo de pérdida de uno u otro.

Cuando se utiliza CSS, la etiqueta HTML no debería proporcionar información sobre cómo va a ser visualizada, solamente marca la estructura del documento. La información de estilo separada fichero CSS, específica cómo se ha de mostrar dicha etiqueta HTML. Algunas de las características que se le pueden definir a esta etiqueta son: color, fuente, alineación del texto, tamaño y posición.

La información de estilo puede ser adjuntada tanto como un fichero separado o en el mismo documento HTML. En este último caso podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo "style".

**Ventajas de utilizar CSS**

- Control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo.
- Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario.
- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño[2]

[1] Cascading Style Sheets.
[2] Siempre y cuando se utilice una hoja de estilos separada del documento HTML.

### 1.3.5 PHP

PHP es un acrónimo recursivo[1] de PHP: *HyperText Preprocessor*, aunque sus orígenes se remontan al nombre **P**ersonal **H**ome **P**age cuando fue creado por Rasmus Lerdof para conocer cuantas personas estaban leyendo su curriculum vitae en su página web.

PHP es un lenguaje de programación interpretado que va embebido (incrustado) en páginas HTML. Su sintaxis es similar a la utilizada en otros lenguajes de programación de alto nivel como C, Java y Perl, sólo con algunas diferencias. El principal objetivo del lenguaje es permitir a los desarrolladores web codificar rápidamente páginas que se generan dinámicamente. (11)

Es un lenguaje del lado del servidor lo que significa que el usuario abre la página HTML y hace una petición de interactuar, el servidor web, con PHP instalado, interpreta la petición y envía una respuesta al usuario, marcando una diferencia entre lenguajes como Javascript, que se ejecuta del lado del cliente.

Es un lenguaje código abierto, y cuenta con una comunidad que se dedica al continuo desarrollo y fortalecimiento del lenguaje.

**Ventajas:**

- PHP es un lenguaje multiplataforma.
- Soporta gran cantidad de gestores de base de datos. PHP implementa una librería para casi todos los Sistemas Gestores de Base de Datos conocidos.
- Es un lenguaje completamente expandible. Está compuesto de un sistema principal, un conjunto de módulos y una gran variedad de extensiones.
- Presenta interfaces distintas para cada tipo de servidor. PHP es soportado por los servidores web más conocidos.
- Gran rapidez. La velocidad aumenta cuando se ejecuta como módulo de Apache y al estar escrito en C se ejecuta rápidamente utilizando poca memoria.
- La versión 5 ya soporta Programación Orientada a Objetos

[1] Llamados así porque en el acrónimo se encuentra el acrónimo.

## 1.4 Sistemas Gestores de Bases de Datos

Un sistema gestor de base de datos (en lo adelante SGBD) es un conjunto de programas que permiten crear y mantener una base de datos asegurando su confidencialidad, integridad y seguridad. (12)

Entre las funciones indispensables de un SGBD se encuentran:

1. Definir una base de datos: especificar estructuras y restricciones de datos.
2. Construir la base de datos: salvar los datos en algún medio controlado por el SGBD.
3. Interactuar con la base de datos: realizar consultas, generar informes.

Entre los SGBD más prestigiosos se encuentran Oracle, SQL Server de Microsoft, MySQL, PostgreSQL.

Características deseables de un SGBD

- Control de redundancia.
- Restricción de los accesos.
- Cumplimiento de las restricciones de integridad.

### 1.4.1 PostgreSQL

PostgreSQL, originalmente se llamó Postgres, fue creado en la Universidad de California en Berkeley, por un profesor de Ciencia de la Computación llamado Michael Stonebraker. PostgreSQL es un potente gestor de base de datos relacional libre (liberado bajo licencia BSD[1]). Cuenta con más de 15 años de desarrollo activo y arquitectura probada que ha ganado mucha reputación por su confidencialidad e integridad en los datos. (13)

**Ventajas:**

**Gran escalabilidad:** Es capaz de ajustarse al número de CPUs y a la cantidad de memoria que posee el sistema de forma óptima, lo que posibilita atender un mayor número de peticiones concurrentes.

[1] Berkeley Software Distribution, licencia de software libre que permite el uso del código fuente en software no libre.

- **Extensible**: El código fuente está disponible para todos sin costo alguno, puede ser personalizado con un mínimo de esfuerzo y sin costos adicionales. (14)
- **Diseñado para ambientes de alto volumen:** PostgreSQL usa una estrategia de almacenamiento de filas llamada MVCC[1] para conseguir una mejor respuesta en ambientes de grandes volúmenes. (14)
- **Soporte:** Existe una comunidad de profesionales que contribuyen con su desarrollo.
- **Multi-plataforma:** Está disponible en casi cualquier UNIX (34 plataformas en la última versión estable) y con versión nativa para Windows. (14)

## 1.5 Plataformas, Herramientas de desarrollo y Frameworks

### 1.5.1 Servidor HTTP Apache

Es un servidor web HTTP de código abierto para plataformas Unix, Microsoft Windows, Macintosh, que implementa el protocolo HTTP 1.1. Este servidor web es uno de los más populares que existen, tanto así que llego a ser el servidor empleado en más del 70% de los sitios web del mundo.

**Ventajas de usar Apache**

- Altamente configurable, aunque no cuenta con una interfaz que facilite el trabajo.
- Arquitectura Modular.
- Código Abierto.
- Multi-plataforma.
- Extensible.

Las vulnerabilidades que se han descubierto y resuelto pueden ser solamente aprovechadas por usuarios locales y no remotamente. Su robustez y estabilidad lo han convertido en el servidor web por excelencia.

[1] Multi-Version Concurrency Control

### 1.5.2 AJAX

AJAX, acrónimo de *Asynchronous JavaScript And XML*. Es una técnica de desarrollo web para lograr aplicaciones interactivas mediante la combinación de tecnologías ya existentes:

1. HTML (o XHTML) y CSS para presentar la información.
2. DOM para interactuar dinámicamente con la presentación.
3. XML, XSLT[1] y JSON[2] para intercambiar y manipular datos de manera asíncrona con un servidor web.
4. XMLHttpRequest para el intercambio asíncrono de información.
5. Javascript para unir todas las tecnologías anteriores.

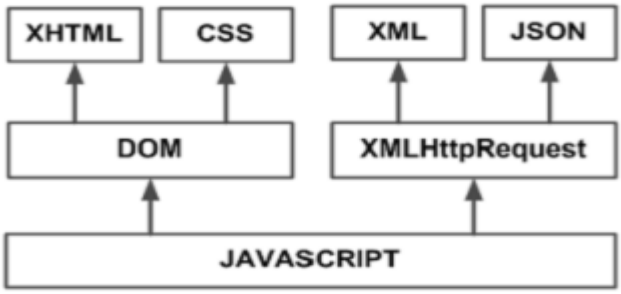


Figura 1: Tecnologías que forman AJAX

AJAX mejora enormemente la interacción del usuario con la aplicación web porque las peticiones que realiza al servidor las hace de forma asincrónica, sin recargar toda la página web sino solamente aquello que deba ser actualizado. Este intercambio de información es completamente trasparente para el usuario.

**Ventajas de usar AJAX**

- Está basado en estándares abiertos.
- Usabilidad.
- Válido en cualquier plataforma y navegador que soporte las tecnologías que lo conforman.

[1] Extensible Stylesheet Language Transformations.
[2] JavaScript Object Notation: Formato ligero para el intercambio de datos.

- Beneficia las aplicaciones web.
- Es independiente del tipo de tecnología que se utilice de servidor.
- Web 2.0, las interfaces de AJAX son una parte fundamental de muchas aplicaciones de la Web 2.0.
- Fácil de utilizar.

### 1.5.3 Frameworks de desarrollo

La palabra inglesa *framework* define un conjunto de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que son utilizados como referencia para enfrentar y resolver nuevos problemas de índole similar.

En el desarrollo de software, un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, un framework proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, un framework facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. (15)

Un framework permite dividir la aplicación informática en capas, principalmente en tres:

- **Presentación**: Manejar las interacciones entre el usuario y el software.
- **Datos**: Permite el acceso a una fuente de almacenamiento persistente.
- **Dominio o Negocio**: Manipular los modelos de datos, en relación a los comandos recibidos desde la Presentación.

Los frameworks de desarrollo son diseñados con la intención de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de proveer un sistema funcional.

### 1.5.3.1 Symfony como framework de desarrollo para aplicaciones web

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web. (15)

**Características de Symfony**

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares) (15)
- Es compatible con casi todos los sistemas de bases de datos.
- Independiente del sistema gestor de bases de datos. (15)
- Incluye herramientas adicionales que ayudará a probar, depurar y documentar el proyecto.
- La capa de internacionalización que incluye Symfony permite la traducción de los datos y de la interfaz, así como la adaptación local de los contenidos. (15)
- Los helpers incluidos permiten minimizar el código utilizado en la presentación, ya que encapsulan grandes bloques de código en llamadas simples a funciones.
- El sistema de enrutamiento y las URL limpias permiten considerar a las direcciones de las páginas como parte de la interfaz, además de estar optimizadas para los buscadores. (15)
- Las interacciones con AJAX son muy fáciles de implementar mediante los helpers que permiten encapsular los efectos Javascript compatibles con todos los navegadores en una única línea de código. (15)
- Symfony está basado en un patrón clásico de arquitectura conocido como Modelo-Vista-Controlador (en lo adelante MVC).
- Se beneficia de una comunidad de desarrolladores en todo el mundo.
- La gestión de la caché reduce el ancho de banda utilizado y la carga del servidor. (15)

Symfony puede ser completamente personalizado para cumplir con los requisitos de las empresas que disponen de sus propias políticas y reglas para la gestión de proyectos y la programación de aplicaciones. (15) Para los desarrolladores que utilizan PHP y los patrones de diseño para crear aplicaciones de Internet utilizar Symfony resulta muy natural y fácil, tanto así que la curva de aprendizaje se reduce muchísimo.

### 1.5.3.2 JQuery como framework para Javascript

JQuery es un framework de Javascript que simplifica la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con AJAX a páginas web.

JQuery ofrece una serie de funcionalidades basadas en el lenguaje Javascript que de otra manera requerirían de mucho más código, logrando buenos resultados en menos tiempo y espacio. (16)

**Características de JQuery**

- Selección de elementos DOM.
- Interactividad y modificaciones del árbol DOM, incluyendo soporte para CSS 1-3
- Posee un mecanismo para la captura de eventos.
- Manipulación de la hoja de estilos CSS.
- Provee disímiles técnicas para añadir animaciones y efectos a una página web.
- Integra funcionalidades para interactuar con el servidor mediante AJAX.
- Soporta extensiones.
- Operar con Objetos y *Arrays*.

La característica principal de la biblioteca es que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX, además de la sencillez de su sintaxis y la poca extensión del código que se necesita escribir.

### 1.5.4 Visual Paradigm como herramienta CASE

Visual Paradigm es una herramienta CASE (Ingeniería de Software Asistida por Computador) que utiliza UML[1] como lenguaje de modelado. Está diseñada para una amplia gama de usuarios interesados en construir sistemas de software fiables con el uso del paradigma orientado a objetos, incluyendo actividades como ingeniería de software, análisis de sistemas y análisis de negocios.

Emplea las últimas notaciones de UML, ingeniería inversa, generación de código, importación de Rational Rose, exportación e importación XML. Soporta aplicaciones web, exporta en formato HTML, es tecnología libre y está disponible en varios idiomas, es fácil de instalar y fácil de actualizar.

De forma general esta herramienta ofrece:

- Creación de diagramas para UML 2.0.
- Diseño centrado en casos de usos y enfocado al negocio, generando un software de mayor calidad.
- Lenguaje estándar para el equipo de desarrollo, facilitando la comunicación.
- Disponibilidad en varias plataformas.

### 1.5.5 *Zend Studio for Eclipse* como IDE para PHP

Zend Studio es un ambiente de desarrollo (IDE por sus siglas en inglés), desarrollado por Zend Technologies basados en el plugin *PHP Development Tools*[2] (PDT) para la plataforma Eclipse. Se encuentra disponible para desarrolladores profesionales que agrupa una serie de componentes de desarrollo que posibilitan la creación de aplicaciones web usando PHP.

*Zend Studio for Eclipse* acelera los procesos de desarrollo y simplifica los proyectos complejos a través de herramientas de edición, depurado, optimización y base de datos. La herramienta está escrita en Java, soporta PHP 4 y PHP 5, posee un manual completo de PHP que puede ser consultado. Fue diseñado

[1] Unified Model Language
[2] Este proyecto está liderado por Zend Technologies

para usarse con el lenguaje PHP, pero ofrece soporte básico para otros lenguajes web como HTML, XML y Javascript.

## 1.6 Metodologías de Desarrollo

El proceso de desarrollo de software es, sin lugar a dudas, una tarea complicada. Para aminorar este problema, en la actualidad se usan metodologías, logrando que el desarrollo de software sea un proceso disciplinado, predecible y eficiente.

### 1.6.1 Rational Unified Process

El proceso unificado de desarrollo (en lo adelante RUP) es una metodología para la ingeniería de software que va más allá del simple análisis y diseño orientado a objetos para facilitar un grupo de técnicas que soporten el ciclo completo de desarrollo de software. El resultado es un proceso basado en componentes, dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental. Su principal objetivo es el desarrollo correcto de software.

**Características y Beneficios de RUP**

- **Las mejores prácticas más probadas de la industria** - Son las mejores prácticas de desarrollo adoptadas en cientos proyectos mundialmente y enseñadas como parte de la curricular en cientos de universidades, la metodología RUP se convirtió rápidamente en el estándar de facto para el proceso de desarrollo en la industria de software. (17)
- **Proceso hecho práctico** - Diferente que otras metodologías comerciales, la plataforma RUP hace que el proceso sea práctico con bases de conocimiento y guías para ayudar en el despegue de la planificación del proyecto, integrar rápidamente a los miembros del equipo y poner en acción el proceso personalizado. (17)
- **Se adapta a las necesidades de los proyectos** - Solo la plataforma RUP proporciona un framework de proceso configurable que permite seleccionar e implantar los componentes específicos de proceso necesarios para proporcionar un proceso consistente y personalizado para cada equipo y proyecto. (17)

RUP organiza los proyectos en términos de disciplinas y fases, consistiendo cada una en una o más iteraciones. Con esta aproximación iterativa, el énfasis de cada flujo de trabajo variará a través del ciclo de vida. La aproximación iterativa ayuda a mitigar los riesgos en forma temprana y continua, con un progreso demostrable y frecuentes versiones ejecutables. (17)

### 1.6.1.1 Caracterización del rol implementador según RUP

Este rol desarrolla los componentes de software y efectúa las pruebas de desarrollador para la integración en subsistemas más grandes, de acuerdo con los estándares adoptados en el proyecto. (18)

**Descripción Principal:**

El rol del implementador es responsable de los componentes de desarrollo y de prueba, de acuerdo con los estándares aprobados por el proyecto, para la integración en subsistemas más grandes. Cuando los componentes de prueba, como controladores o fragmentos para simulación, deben crearse para dar soporte a las pruebas, el implementador también es responsable del desarrollo y las pruebas de los componentes de prueba y los subsistemas correspondientes. (18)

**Habilidades:**

Las habilidades y conocimientos apropiados para el implementador incluyen:

- Conocimiento del sistema o aplicación que se somete a prueba
- Familiaridad con las herramientas de prueba y de automatización de prueba

**Propuestas de Asignación:**


**Figura 2: Propuesta de Asignación**

A un implementador se le puede asignar la responsabilidad de implementar una parte estructural del sistema (como un subsistema de implementación o de clases), o una parte funcional del sistema, como la ejecución de casos de uso o sus características. (18)

Es posible que dos personas actúen como implementador de un único componente del sistema, dividiendo las responsabilidades entre ellos o efectuando las tareas conjuntamente, como en un enfoque de programación en parejas. (18)

## 1.7 Conclusiones parciales

Luego de haber planteado las principales características de las herramientas y tecnologías que serán utilizadas en el desarrollo de esta investigación es posible afirmar que el uso de estas será de vital importancia en lograr un desarrollo óptimo de la solución propuesta minimizando tiempo y costos. Además se especifica claramente las tareas de un desarrollador según RUP, así como las principales habilidades que este debe desarrollar.

# CAPÍTULO 2

## 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 se muestra una descripción de las principales clases y funcionalidades.

## 2.2 Valoración Crítica del diseño propuesto por el analista

### El patrón Modelo-Vista-Controlador

Symfony está basado en un patrón clásico de arquitectura conocido como arquitectura Modelo-Vista-Controlador formado por tres niveles:

1. **El Modelo** representa la información con la que trabaja la aplicación, es decir, su lógica de negocio. (15)
2. **La Vista** transforma el modelo en una página web que permite al usuario interactuar con ella. (15)
3. **El Controlador** se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista. (15)

La arquitectura MVC separa la lógica de negocio (**el modelo**) y la presentación (**la vista**) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. (15)

El principio más importante de la arquitectura MVC es la separación del código del programa en tres capas, dependiendo de su naturaleza. La lógica relacionada con los datos se incluye en el modelo, el código de la presentación en la vista y la lógica de la aplicación en el controlador. (15)

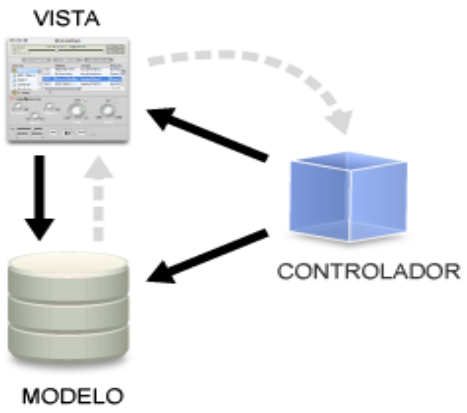


**Figura 3: Funcionamiento del patrón MVC**

En Symfony, la capa del controlador, que contiene el código que liga la lógica de negocio con la presentación, está dividida en varios componentes, utilizados para diversos propósitos:

- El controlador frontal es el único punto de entrada a la aplicación.
- Las acciones contienen la lógica de la aplicación.
- 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.
- Los filtros son trozos de código ejecutados para cada petición, antes o después de una acción.

La lógica de negocio de las aplicaciones web depende casi siempre en su modelo de datos. El componente que se encarga por defecto de gestionar el modelo en Symfony es una capa de tipo ORM (*object/relational mapping*) realizada mediante el proyecto Propel. En las aplicaciones Symfony, el acceso y la modificación de los datos se realiza mediante objetos, evitando, de esta forma, el acceso explícito a la base de datos. Esto permite un alto nivel de abstracción y fácil portabilidad. (15)

La vista se encarga de producir las páginas que se muestran como resultado de las acciones. La vista en Symfony está compuesta por diversas partes, estando cada una de ellas especialmente preparada para que pueda ser fácilmente modificable por la persona que normalmente trabaja con cada aspecto del diseño de las aplicaciones. (15)

A continuación se presenta un modelo de clases de diseño presentado por el analista para el caso de uso *Gestionar Medidas*

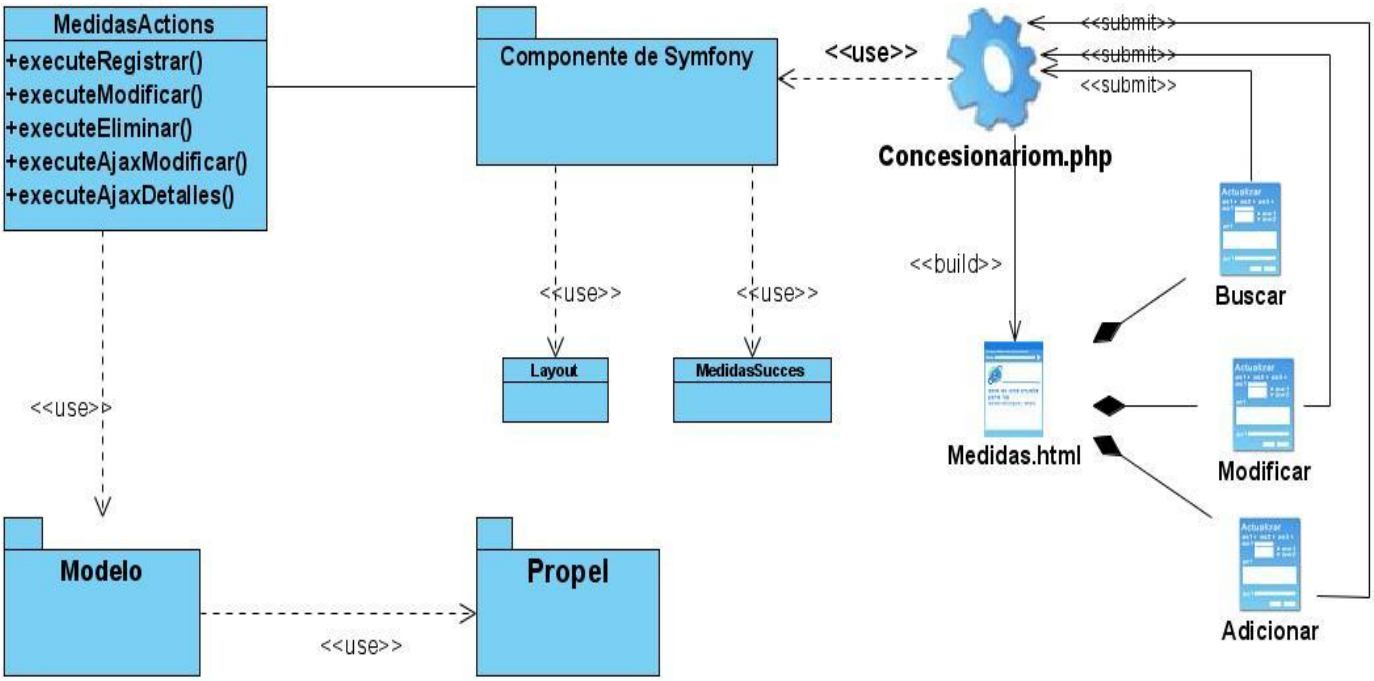


**Figura 4: Modelo de Diseño Caso de Uso Gestionar Medidas**

**Consideraciones del Diseño**

El diseño propuesto está basado en el patrón arquitectónico MVC, en Symfony, una de las partes del controlador son las acciones, estas son una parte vital de la aplicación, debido a que contienen toda la lógica de la aplicación. Las acciones 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. (15)

Las acciones son métodos con el nombre `executeNombreAccion` de una clase llamada `nombreModuloActions` que hereda de la clase `sfActions`. Estas clases se encuentran agrupadas por módulos. La clase que representa las acciones de un módulo se encuentra en el archivo `actions.class.php`, en el directorio `actions/` del módulo. (15)

Una sintaxis alternativa para las acciones sería distribuir las acciones en archivos separados, es decir una acción por archivo. En este caso se tendría una clase `nombreAccionAction` y esta extendería de la clase `sfAction`[1]. El nombre del método es simplemente `execute`. El nombre del fichero es el mismo que el de la clase. (15)

En el diseño propuesto se utiliza la primera opción, es decir una sola clase donde las acciones son funciones de ella. Esta alternativa permite mayor organización en el código debido a que todo se encuentra en un solo fichero, además de ser consistente con otros patrones de diseño como Alta Cohesión y Bajo Acoplamiento.

Por otra parte, en el diseño propuesto por el analista, se utiliza un solo controlador frontal para la aplicación que sería *concesionariom.php*, evidenciándose el patrón de diseño creacional *Singleton*, que plantea que una clase será instanciada una sola vez durante la ejecución del software.

Las plantillas es una de las partes más importantes que utiliza Symfony para implementar la capa de la Vista en el modelo MVC y su contenido está compuesto por código HTML, y algo de PHP embebido y sencillo, además de llamadas a variables creadas en la acción (15). Symfony incluye la lógica dentro de las acciones lo que permite disponer de varias plantillas para una sola acción sin tener que duplicar el código (15). Sin embargo en el diseño planteado se utiliza una sola plantilla o página cliente para cada página servidora logrando organización y unificación en la tarea a realizar.

El modelo, en Symfony, cuando es generado se realiza un mapeo de las tablas de la Base de Datos, generando por cada tabla 4 clases, de ellas 2 relacionadas directamente con el acceso a datos y otras dos para implementar la lógica del negocio. Por otra parte es común que la lógica del negocio se lleve a cabo en las acciones del controlador. En el diseño propuesto se eligió la primera alternativa, donde la lógica del negocio se llevará a cabo en las clases del modelo, manifestándose nuevamente los patrones de diseño Bajo Acoplamiento, puesto que cada clase realiza las funcionalidades de la tabla que le corresponda en la Base de Datos y el patrón Alta Cohesión, donde las clases no son sobrecargadas de funcionalidades, las acciones en este caso solo serán responsables de interrelacionar la vista con el modelo.

[1] En vez de heredar de sfActions

## 2.3 Análisis de posibles implementaciones, componentes o módulos ya existentes.

La reutilización de código se refiere a las técnicas que garantizan que una parte o la totalidad de un programa informático existente se puedan emplear en la construcción de otro programa, aprovechándose el trabajo anterior, economizando tiempo y eliminando la redundancia.

En el desarrollo de la presente implementación se utilizan los beneficios de la reutilización de código, debido a que se usan módulos y componentes implementados con anterioridad, específicamente se puede mencionar el Módulo de Seguridad que proporciona todos los elementos para lograr la seguridad del módulo en desarrollo. Este módulo de seguridad es el encargado de la asignación de roles a los usuarios de la aplicación y de manipular los permisos de estos cuando se encuentren interactuando con la aplicación.

El lenguaje PHP permite además incluir código mediante la instrucción *include(),* recibiendo como parámetro la dirección física del fichero donde se encuentra el código que se pretende incluir. Sin embargo, esta forma de trabajar con fragmentos de código no es muy limpia, sobre todo porque puede que los nombres de las variables utilizadas no coincidan en el fragmento de código y en las distintas plantillas. Además, el sistema de cache de Symfony no puede detectar el uso de *include(),* por lo que no se puede incluir en la cache el código del fragmento de forma independiente al de las plantillas. Symfony define 3 alternativas al uso de la instrucción *include()* y que permiten manejar de forma inteligente los fragmentos de código:

- Si el fragmento contiene poca lógica, se puede utilizar un archivo de plantilla al que se le pasan algunas variables. En este caso, se utilizan los elementos parciales (*partial*).
- Si la lógica es compleja (por ejemplo se debe acceder a los datos del modelo o se debe variar los contenidos en función de la sesión) es preferible separar la presentación de la lógica. En este caso, se utilizan componentes (*component*).
- Si el fragmento va a reemplazar una zona específica del layout, para la que puede que exista un contenido por defecto, se utiliza un (*slot)*.

En el desarrollo del módulo Registro Minero, se utiliza, principalmente la alternativa parcial, que al decir de sus creadores, no es más que un trozo de código de plantilla reutilizable. Estos elementos se encuentran en el directorio `templates/` de un proyecto Symfony y contienen código HTML y PHP. El nombre de un elemento parcial siempre comienza con un guión bajo (_), diferenciándose de esta manera de las plantillas.

El paradigma de programación POO, posibilita de manera innata la reutilización de código, debido a la utilización de clases para definir objetos, y de procedimientos o métodos para definir el comportamiento de estos objetos. En la implementación de este módulo se aprovecha esto utilizando tres clases de verdadera importancia debido a que estas son las responsables de la generación y/o exportación de la información que se manipula en el Módulo Registro Minero a los formatos Excel, Word y PDF.

## 2.4 Descripción de las nuevas clases u operaciones fundamentales.

Durante la implementación del módulo se han construido nuevas clases y funciones con el objetivo de llevar a cabo los principales procesos que propician el funcionamiento correcto del Registro Minero. A continuación se explican las principales clases con sus funcionalidades.

| **Nombre:** solicitudActions | |
|---|---|
| **Tipo de la clase:** Controladora | |
| **Para cada responsabilidad** | |
| **Nombre:** | **Descripción:** |
| `executeIndex()` | Función principal para poder acceder al módulo de la gestión de las solicitudes de concesión. |
| `executeRegistrar ()` | Función encargada de recoger los datos de la vista y enviarlos al modelo para que sean insertados en la base de datos. |
| `executeModificar ()` | Función que carga los datos de una solicitud de concesión que va a ser modificada. |
| `executeModificarSolicitud()` | Función que recoge los datos de la vista y los |

| | |
|---|---|
| | envía al modelo para que la solicitud sea modificada con estos nuevos datos. |
| executeEliminar() | Función encargada eliminar una solicitud de concesión, dado el id de esta, mediante el método eliminar de la clase del modelo TsolicitudPrimariaPeer. |
| executeBuscarSolicitud () | Es la función encargada de llamar al método Buscar del modelo y además obtiene los resultados devueltos por el anterior y le da formato para que sean mostrados en la tabla paginada. |
| executeAjaxCompletarMunicipios () | Función utilizada para completar un listado de municipios en la vista, usando AJAX. |
| executeAjaxCompletarMinerales() | Función utilizada para completar un listado de minerales en la vista, usando AJAX. |
| executeAjaxCompletarUsos() | Función utilizada para completar un listado de usos de los minerales en la vista, usando AJAX. |
| executeAjaxDetalles( ) | Función encargada de devolver a la vista, utilizando AJAX, todos los datos de una solicitud de concesión. |
| executeAceptar () | Función que acepta una solicitud de concesión, registrándose, en el modelo, un nuevo derecho minero. |
| executeDenegar() | Función que deniega una solicitud de concesión. |

| **Nombre:** TmedidaPeer |
|---|
| **Tipo de la clase:** Controladora |
| **Para cada responsabilidad** |

| Nombre: | Descripción: |
|---|---|
| registrarMedida () | Función que lleva a cabo la inserción en la base de datos de una nueva medida |
| eliminarMedida () | Función encargada de eliminar una medida de la base de datos. |
| modificarMedida() | Función que modifica los datos de una medida en la base de datos. |
| getMedidas () | Función que devuelve todas las medidas que existen en la base datos. |
| obtenerMedidasTexto () | Función que ejecuta un procedimiento almacenado en el gestor de base de datos. |
| CantidadMedidasTexto () | Función que ejecuta un procedimiento almacenado en el gestor de base de datos. |
| buscarDM () | Función utilizada para completar un listado de municipios en la vista, usando AJAX. |

| **Nombre:** Tdictamen | |
|---|---|
| **Tipo de la clase:** Entidad | |
| **Para cada responsabilidad** | |
| **Nombre:** | **Descripción:** |
| getIddictamen () | Obtener el id correspondiente a un dictamen. |
| getFechaDevuelto () | Obtener la fecha de devolución de un dictamen. |
| getFechaEntrega () | Obtener la fecha de entrega de un dictamen. |
| getDescripcion () | Obtener la descripción del dictamen. |
| getIdsolicitud () | Obtener el id de la solicitud a la que se le realiza |

| | |
|---|---|
| | dictamen. |
| setIddictamen () | Asigna un nuevo valor al id del dictamen. |
| setFechaDevuelto () | Asigna un nuevo valor a la fecha de devolución del dictamen. |
| setFechaEntrega () | Asigna un nuevo valor a la fecha de entrega del dictamen. |
| setDescripcion () | Asigna un nuevo valor a la descripción del dictamen. |
| setIdsolicitud() | Asigna un nuevo valor al id de la solicitud a la que se le realiza dictamen. |
| save () | Salva los datos del dictamen en la base de datos. |
| getTsolicitudPrimaria () | Devuelve un objeto de la clase TsolicitudPrimaria. |

## 2.5 Estilos de Programación.

El código de un programa lo leen muchas personas, bien para corregirlo, para ampliarlo o simplemente para evaluarlo. Para estas personas es fundamental que el código se encuentre bien redactado para que su significado sea claro y nítido. Sin embargo la legibilidad de un programa no depende sólo de las características del lenguaje sino que depende en gran medida del estilo de codificación en que está escrito. A diferencia de la sintaxis de un programa que son reglas estáticas que obligatoriamente hay que seguir, un estilo de programación está constituido por directrices que ayudan a obtener programas más legibles. Si bien no existen estilos de programación correctos o incorrectos es aconsejable la adopción de un conjunto de normas para la escritura de programas. Estas normas, básicamente, se refieren a la forma en que se colocan las llaves, a como se identa el código y como se ubican los paréntesis.

Existen diversos criterios para un buen estilo, algunos de ellos son:

- **Nombres significativos:** Los nombres de funciones y variables definidos por el programador deben ser significativos y auto-explicativos con respecto a su función. **Ejemplo**: Si se debe designar una variable que almacenará la edad de una persona una forma correcta de hacerlo sería: `$edad = $persona->getEdad();` y no `$x = $a->getEdad();`
- **Indentación y espacios en el código:** Muestra las líneas que se encuentran subordinadas a otras líneas. **Ejemplo:**

```
if($flag == true)
{
	$cantidad++;
	$flag = !$flag;
}
```

Nótese como todas las instrucciones que se ejecutan dentro del **if** se encuentran dentro del mismo.

- **Documentar el código:** Las complicadas e inusuales secciones de código, así como las funciones deben ser comentadas para facilitar su comprensión.

Existen estilos de programación que por su fácil estructura y organización se han convertido en los más conocidos y utilizados por los desarrolladores pudiéndose mencionar dentro de ellos a los siguientes:

**Estilo K&R y BSD KNF:** Estilo muy usado en el lenguaje C y PHP. Se trata de abrir la llave en la misma línea de declaración de la orden, identando los siguientes pasos al mismo nivel que la llave y cerrando la llave en el mismo nivel que la declaración. Normalmente las tabulaciones en Windows son de 4 espacios, cuando las tabulaciones tienen 8 espacios se trata del estilo BSD o KNF, muy usado en Linux.

```
function Ejemplo($val){
	if($val == 1){
		echo $foo;
	}else {
		echo $bar;
	}
}
```

La ventaja de usar este estilo es que las llaves iniciales no necesitan ninguna línea extra para ellas solas, por lo que se ahorra espacio vertical de lectura. La desventaja de este estilo es que las llaves de cierre si necesitan una línea exclusiva para ellas. Resulta difícil identificar el comienzo de las llaves de un bloque. Sin embargo es fácil identificar el comienzo de un bloque debido a la identación a la derecha. (19)

**Estilo Allman:** Se trata de crear una nueva línea para las llaves, e identar el código debajo de ellas. La llave de cierre tiene el mismo identado que la de inicio.

```
function Ejemplo($val)
{
    if($val % 2 == 0)
    {
        echo “El valor es par”;
    }
    else
    {
        echo “El valor es impar”;
    }
}
```

Este estilo mantiene un código limpio y claro. Las llaves de inicio y fin coinciden en la misma columna, haciendo más fácil la identificación de cada bloque. Además, es más difícil olvidarse cerrar una llave cuando se identifica claramente la llave inicial. (19)

La desventaja de este estilo, es que las llaves ocupan enteramente una línea, ocupando más espacio vertical de lectura. En monitores con resoluciones bajas (24 líneas) resulta difícil leer código fuente con este estilo aunque esto poco afecta en monitores con resoluciones más grandes. (19)

**Estilo GNU:** Parecido al estilo Allman, se trata de poner las llaves en una nueva línea. La diferencia es que las llaves deben estar identadas por 2 espacios y el código dentro de ellas debe estarlo por 2 espacios más. (19)

Para la implementación del Módulo Registro Minero se ha utilizado el estilo Allman, aunque con algunas modificaciones para adaptarlo a las condiciones y necesidades propias del SGDG.

## 2.6 Estándares de Codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, debe establecerse un estándar de codificación para asegurar que todos los programadores del proyecto trabajen de forma coordinada. Cuando el proyecto de software incorpore código fuente previo, o bien cuando realice el mantenimiento de un sistema de software creado anteriormente, el estándar de codificación debería establecer cómo operar con la base de código existente. (20)

La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. La mantenibilidad del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores o mejorar el rendimiento. Aunque la legibilidad y la mantenibilidad son el resultado de muchos factores, una faceta del desarrollo de software en la que todos los programadores influyen especialmente es en la técnica de codificación. El mejor método para asegurarse de que un equipo de programadores mantenga un código de calidad es establecer un estándar de codificación sobre el que se efectuarán luego revisiones del código de rutinas. (20)

Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento. Además, si se aplica de forma continuada un estándar de codificación bien definido, se utilizan técnicas de programación apropiadas, y, posteriormente, se efectúan revisiones del código de rutinas, caben muchas posibilidades de que un proyecto de software se convierta en un sistema de software fácil de comprender y de mantener. (20)

Aunque el propósito principal para llevar a cabo revisiones del código a lo largo de todo el desarrollo es localizar defectos en el mismo, las revisiones también pueden afianzar los estándares de codificación de manera uniforme. La adopción de un estándar de codificación sólo es viable si se sigue desde el principio

hasta el final del proyecto de software. No es práctico, ni prudente, imponer un estándar de codificación una vez iniciado el trabajo. (20)

En la implementación del Módulo Registro Minero se utilizará un estándar de codificación que pone en práctica dos notaciones muy conocidas por los desarrolladores:

**Notación PascalCasing:** Los nombres de variables y funciones con una o más palabras se escribirán cada palabra con letra inicial mayúscula. Ejemplo: `$CantidadTuplasDevueltas = 0;`

**Notación CamelICasing:** Los nombres de variables y funciones con una o más palabras se escribirán la primera palabra con letra inicial minúscula y las demás con letra inicial mayúscula. Ejemplo: `$cantidadTuplasDevueltas = 0;`

Además contiene otros elementos importantes que se detallan a continuación con los debidos ejemplos que lo evidencien:

**Descripción de las clases**

Todas las clases deberán tener una breve descripción dentro de un comentario multilínea con el siguiente formato:

```
/**
 * class: MiClase
 * Esta clase es para ...
 *
 * @package    sgdg
 * @subpackage login
 * @author     Nombre del Desarrollador
 * @version    Sistema de Gestión de Datos Geológicos versión 1.0
 */
```

**Descripción de las funciones**

Todas las funciones deberán tener una breve descripción dentro de un comentario multilínea con el siguiente formato:

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

**Ejemplo de cómo quedarían las clases con sus funciones:**

```
/**
 * class: solicitudActions.
 *
 * Esta clase contiene las acciones del módulo Registro
 * Minero, caso de uso Gestionar
 * Solicitud Concesión
 * @package    sgdg
 * @subpackage solicitud
 * @author     Joel Macías Roque
 * @version    Sistema de Gestión de Datos Geológicos 1.0
 */
```

```
class solicitudActions extends sfActions
{
 /**
  * function: executeRegistrar
```

```
 * Esta función permite registrar una solicitud de concesión y lo redirecciona hacia la página
 * principal de la aplicación.
 *
 * @author Joel Macías Roque.
 */
 public function executeRegistrar()
 {
     if($this->getRequest()->getMethod() != sfRequest::POST)
     {
           $this->redirect('solicitud/index');
     }
     else
     {
           $nombreSolicitud = $this->getRequestParameter('txtNombreSolicitud');
           $nombreSolicitante = $this->getRequestParameter('txtSolicitante');
           $ministerio = $this->getRequestParameter('txtMinisterio');
           $fechaPresentacion = $this->getRequestParameter('txtFechaPresentacion');
           $municipios = $this->getRequestParameter('cbxMunicipios');
           $hectareas = $this->getRequestParameter('txtHectareas');
           $termino = $this->getRequestParameter('txtTermino');
           $objetivos = $this->getRequestParameter('txtObjetivos');
           $minerales = $this->getRequestParameter('cbxMinerales');
           $cantidadMinerales = $this->getRequestParameter('hddCantidadMinerales');
           $arregloUsos = array();

           for($i = 1; $i <= $cantidadMinerales; $i++)
           {
                 $arregloUsos[] = $this->getRequestParameter('cbxUsos'.$i);
           }

           $tipoSolicitud = $this->getRequestParameter('cbxTipoSolicitud');
           TsolicitudPeer::registrarSolicitud($nombreSolicitud,   $nombreSolicitante,
$ministerio,  $objetivos,  $fechaPresentacion,  $municipios,  $hectareas,  $termino,
$tipoSolicitud, $minerales, $arregloUsos);

           $this->redirect('solicitud/index');
     }
  }
```

**Comentarios de las variables**

Los comentarios de las variables deberán ser cortos y precisos, y estarán ubicados de la línea donde fue declarada la variable y se hará usando comentarios simples. Ejemplo:

```
public function EstoEsUnEjemplo()
{
      $foo = 1; // evidencia lo expuesto anteriormente
      $bar = false; //otro ejemplo
}
```

**Programación en las clases de las acciones y en las de acceso a datos**

En las clases de las acciones las funciones no deben tener código que realice consultas a la base de datos, estas consultas serán hechas en funciones de las clases del modelo de datos y serán accedidas desde las acciones.

Ejemplo:

```
public function executeIndex()
{
    $this->solicitantes = TsolicitantePeer::LlenarSolicitante();
    $this->entidades = TempresaPeer::LlenarEntidad();
}
```

El estándar seleccionado plantea además un grupo de requerimientos que deben ser respetados cuando se vaya a implementar el Módulo Registro Minero, se mostrarán los fundamentales:

**Organización Visual**

- Deben incluirse espacios entre todos los operadores binarios.
- Los operadores unarios se deberán colocar sin espacio alguno junto a sus operandos.
- La idetanción tendrá una longitud de 4 espacios.

**Características de un buen programa**

- Ejecución correcta: funciona de acuerdo a las especificaciones.
- Fácil de depurar.
- Fácil de mantener.

- Ejecución eficiente: uso mínimo de recursos de memoria y tiempo.

En el

*Anexo # 2 Estándar de Codificación del* SGDG de este documento se explica detalladamente el estándar de codificación adoptado para la implementación del Módulo Registro Minero.

## 2.7 Conclusiones parciales

Una vez realizada la valoración del diseño propuesto, así como la descripción de las implementaciones ya existentes se puede concluir que el diseño que se propone está bien estructurado facilitando en gran medida las bases para la implementación. Otro elemento importante tratado en este capítulo fue el estándar de codificación que se usará en la implementación del Módulo Registro Minero, elemento este importante a la hora de documentar y dar mantenimiento al sistema.

# CAPÍTULO 3

## 3.1 Introducción

En este capítulo se pretende validar la solución propuesta, mediante la construcción de casos de pruebas de caja negra orientados a comprobar los requisitos funcionales del módulo Registro Minero, sin llegar a probar directamente el código. El objetivo en este capítulo es la construcción de los casos de pruebas para las principales funcionalidades del módulo, cumpliendo así con una de las tareas que plantea RUP para el rol de implementador.

## 3.2 Descripción general de las pruebas

Una vez que se ha generado el código, comienzan las pruebas del programa. El proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales; es decir, realizar las pruebas para la detección de errores y asegurar que la entrada definida produce resultados reales de acuerdo con los resultados requeridos (21).

Las pruebas del software es un elemento crítico para la garantía de calidad del software, para las primeras pruebas al módulo Registro Minero, se escogió la técnica conocida como caja negra.

Las pruebas de caja negra se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación. Por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro. (22)

Las pruebas de caja negra se apoyan en la especificación de requisitos del módulo. De hecho, se habla de "cobertura de especificación" para dar una medida del número de requisitos que se han probado.

A la vista de los requisitos de un módulo, se sigue una técnica algebraica conocida como "clases de equivalencia". Esta técnica trata cada parámetro como un modelo algebraico donde unos datos son equivalentes a otros. Si logramos partir un rango excesivamente amplio de posibles valores reales a un conjunto reducido de clases de equivalencia, entonces es suficiente probar un caso de cada clase, pues los demás datos de la misma clase son equivalentes. (22)

## 3.3 Diseño de las pruebas que permitan validar la solución propuesta

### 3.3.1 Gestionar Obligaciones

**Descripción General:** Gestionar Obligaciones permite adicionar, modificar, eliminar o mostrar los detalles de una obligación referente a un derecho minero.

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

**Escenarios a probar en el Caso de Uso**

| Nombre de la sección | Escenarios de la sección | Descripción de la funcionalidad |
|---|---|---|
| SC1: Buscar Obligaciones | EC 1.1: Se marca la opción buscar | El sistema muestra una lista con todas las obligaciones y sus datos. |
| | EC 1.2: Se introduce el texto por el cual se desea buscar una obligación. | El sistema muestra un listado con todas las obligaciones y sus datos que tienen en cualquiera de sus campos el texto entrado. |
| SC2: Insertar Obligaciones | EC 2.1: Se adiciona una obligación correctamente | El sistema muestra un formulario para insertar datos. Primero se selecciona el nombre del derecho |

| | | |
|---|---|---|
| | | minero, y a partir de ahí se muestran los datos a insertar:<br><br>Tipo de obligación, fecha de presentación y de cumplimiento, y las observaciones. |
| | EC 2.2: : Los datos entrados son incorrectos | El sistema muestra un mensaje especificando el error, y no se realiza el adicionar una obligación. |
| | EC 2.3: Se cancela la inserción. | Después que inserte los datos puedo cancelar, y el sistema no inserta ningún dato y cierra la interfaz de insertar. |
| SC3: Modificar Obligación | EC 3.1: Se modifica una obligación correctamente.. | El sistema muestra un formulario para modificar con los siguientes datos: Tipo de obligación, fecha de presentación y de cumplimiento, y las observaciones |
| | EC 3.2: : Los datos no son correctos | El sistema manda un mensaje especificando el error, y no se modifican los datos. |
| | EC 3.3: Se cancela la modificación. | Luego de haber modificado los datos, se puede cancelar dicha modificación, y el sistema no modifica y cierra la interfaz de modificar |
| SC4: Eliminar una obligación | EC4.1: Se elimina una obligación. | El sistema da la opción de eliminar, se selecciona en la lista la obligación que se desea eliminar y se elimina. |

| | | |
|---|---|---|
| | EC4.2: No se elimina | Si no se desea eliminar luego de haber apretado el botón eliminar, este tiene dos opciones aceptar (y se elimina la obligación), o de cancelar, y no se realiza dicha eliminación. |

**Descripción de las variables**

| No | Nombre de campo | Clasificación | Valor Nulo | Descripción |
|---|---|---|---|---|
| 1 | Nombre del derecho minero | Lista desplegable | NO | Letras y números. |
| 2 | Tipo de obligación | Lista desplegable | NO | Letras |
| 3 | Fecha de presentación. | Campo de selección | NO | Solo son números. |
| 4 | Fecha de cumplimiento | Campo de selección | NO | Solo son números. |
| 5 | Frecuencia de la obligación | Campo de texto | NO | Solo son números. |
| 6 | Observaciones | Campo de texto | SI | Se admiten letras y números. |
| 7 | Cancelar | Botón | SI | Botón para cancelar |

**SC1: Buscar Obligaciones**

| Escenario | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Respuesta del Sistema | Resultado de la Prueba |
|---|---|---|---|---|---|---|---|---|---|
| EC 2.1: Se adiciona una obligación correctamente | V | V | V | V | V | NA | I | El sistema inserta los datos de la Obligación. | |
| EC 2.2: Los datos entrados son incorrectos | I | V | V | V | V | NA | I | El sistema muestra un mensaje especificando el error,y no se realiza la inserción de los datos. | |
| | V | I | V | V | V | NA | I | El sistema muestra un mensaje especificando el error y no se realiza la inserción de los datos. | |
| | V | V | I | V | V | NA | I | El sistema muestra un mensaje especificando el error y no se realiza la inserción de los datos. | |
| | V | V | V | I | V | NA | I | El sistema muestra un mensaje especificando el error y no se realiza la inserción de los datos. | |
| | V | V | V | V | I | NA | I | El sistema muestra un mensaje especificando el error y no se realiza la inserción de los datos. | |
| | I | I | I | I | I | NA | I | El sistema muestra un mensaje con todos los errores y no se realiza la inserción de los datos. | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| EC 2.3: Se cancela la inserción. | NA | NA | NA | NA | NA | NA | V | El sistema cancela la Inserción de una nueva obligación. | |

**SC: 2 Insertar Obligaciones**

| Escenario | Buscar | Respuesta del Sistema | Resultado de la Prueba |
|---|---|---|---|
| EC 1.1: Se marca la opción buscar | V | El sistema muestra el listado con todos los obligaciones existente en el sistema. | |
| EC 1.2: Se introduce el texto por el cual se desea buscar una obligación. | V | El sistema debe de mostrar una lista con todas las obligaciones que tienen en alguno de sus campos el texto igual al entrado. | |

**SC: 3 Modificar Obligaciones**

| Escenario | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Respuesta del Sistema | Resultado de la Prueba |
|---|---|---|---|---|---|---|---|---|---|
| EC 3.1: Se modifica una obligación correctamente | V | V | V | V | V | NA | I | El sistema modifica los datos de la Obligación seleccionada. | |
| EC 3.2 : Los datos entrados son | I | V | V | V | V | NA | I | El sistema muestra un mensaje especificando el error y no se realiza la modificación de los | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| incorrectos | | | | | | | | datos. | |
| | V | I | V | V | V | NA | I | El sistema muestra un mensaje especificando el error y no se realiza la modificación de los datos. | |

**SC: 4 Eliminar Obligaciones**

| Escenario | Aceptar | Cancelar | Respuesta del Sistema | Resultado de la Prueba |
|---|---|---|---|---|
| EC 4.1: Se elimina una obligación | V | I | Se elimina la obligación satisfactoriamente | |
| EC 4.2: No se elimina | I | V | No se llega a eliminar, ha sido cancelada | |

### 3.3.2 Gestionar Medidas

**Descripción General:** Gestionar Medidas permite adicionar, modificar, eliminar o mostrar los detalles de una medida aplicada a un concesionario por el incumplimiento de alguna de sus obligaciones.

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

**Escenarios a probar en el Caso de Uso**

| Nombre de la sección | Escenarios de la sección | Descripción de la funcionalidad |
|---|---|---|
| SC1: Buscar Medidas | EC 1.1: Se marca la opción buscar | El sistema muestra una lista con todas las medidas y sus datos. |
| | EC 1.2: Se introduce el texto por el cual se desea buscar una medida. | El sistema muestra un listado con todas las medidas y sus datos que tienen en cualquiera de sus campos el texto entrado. |
| SC2: Insertar Medida | EC 2.1: Se adiciona una medida correctamente | El sistema muestra un formulario para insertar datos. Primero se selecciona el nombre del derecho minero, y a partir de ahí se muestran los datos a insertar:<br>Fecha de imposición de la medida, fecha limite |
| | EC 2.2: : Los datos entrados son incorrectos | El sistema muestra un mensaje especificando el error, y no se realiza el adicionar una medida. |
| | EC 2.3: Se cancela la inserción. | Después que inserte los datos puedo cancelar, y el sistema no inserta ningún dato y cierra la interfaz de insertar. |
| SC3: Modificar Medidas | EC 3.1: Se modifica una medida correctamente. | El sistema muestra un formulario para modificar con los siguientes datos: nombre del derecho minero, medida aplicada, fecha de imposición, fecha límite, fecha en que se cumple la medida, y observaciones. |

| | | |
|---|---|---|
| | EC 3.2: : Los datos no son correctos | El sistema manda un mensaje especificando el error y no se modifican los datos. |
| | EC 3.3: Se cancela la modificación. | Luego de haber modificado los datos, se puede cancelar dicha modificación y el sistema no modifica y cierra la interfaz de modificar. |
| SC4: Eliminar una Medida | EC4.1: Se elimina una medida. | El sistema da la opción de eliminar, se selecciona en la lista la medida que se desea eliminar y se elimina. |
| | EC4.2: No se elimina | Si no se desea eliminar luego de haber apretado el botón eliminar, este tiene dos opciones aceptar (y se elimina la obligación), o de cancelar y no se realiza dicha eliminación. |

**Descripción de las variables**

| No | Nombre de campo | Clasificación | Valor Nulo | Descripción |
|---|---|---|---|---|
| 1 | Nombre del derecho minero | Lista desplegable | NO | Letras y números. |
| 2 | Medida aplicada | Campo de texto | NO | Solo letras |
| 3 | Fecha de imposición. | Campo de selección | NO | Solo son números. |
| 4 | Fecha límite | Campo de selección | NO | Solo son números. |

| | | | | |
|---|---|---|---|---|
| 5 | Fecha de cumplimiento de la medida | Campo de selección | NO | Solo son números. |
| 6 | Observaciones | Campo de texto | SI | Se admiten letras y números. |
| 7 | Cancelar | Botón | SI | Botón para cancelar. |

**SC1 Buscar Medidas**

| Escenario | Buscar | Respuesta del Sistema | Resultado de la Prueba |
|---|---|---|---|
| EC 1.1: Se marca la opción buscar | V | El sistema muestra el listado con todos los medidas existente en el sistema. | |
| EC 1.2: Se introduce el texto por el cual se desea buscar una obligación. | V | El sistema debe de mostrar una lista con todas las medidas que tienen en alguno de sus campos el texto igual al entrado. | |

**SC2 Insertar Medidas**

| Escenario | 1 | 2 | 3 | 4 | 5 | 6 | Respuesta del Sistema | Resultado de la Prueba |
|---|---|---|---|---|---|---|---|---|
| EC 2.1: Se adiciona una medida correctamente | V | V | V | V | NA | I | El sistema inserta los datos de la Medida. | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| EC 2.2: Los datos entrados son incorrectos | I | V | V | V | NA | I | El sistema muestra un mensaje especificando el error, y no se realiza la inserción de los datos. | |
| | V | I | V | V | NA | I | El sistema muestra un mensaje especificando el error, y no se realiza la inserción de los datos. | |
| | V | V | I | V | NA | I | El sistema muestra un mensaje especificando el error, y no se realiza la inserción de los datos. | |
| | V | V | V | I | NA | I | El sistema muestra un mensaje especificando el error, y no se realiza la inserción de los datos. | |
| | I | I | I | I | NA | I | El sistema muestra un mensaje con todos los errores y no se realiza la inserción de los datos. | |
| EC 2.3: Se cancela la inserción. | NA | NA | NA | NA | NA | V | El sistema cancela la Inserción de una nueva medida. | |

**SC3 Modificar Medidas**

| Escenario | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Respuesta del Sistema | Resultado de la Prueba |
|---|---|---|---|---|---|---|---|---|---|
| EC 3.1: Se modifica una medida correctamente | V | V | V | V | V | NA | I | El sistema modifica los datos de la medida seleccionada. | |
| EC 3.2 : Los datos entrados son incorrectos | I | V | V | V | V | NA | I | El sistema muestra un mensaje especificando el error, y no se realiza la modificación de los datos. | |
| | V | I | V | V | V | NA | I | El sistema muestra un mensaje especificando el error, y no se realiza la modificación de los datos. | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | V | V | I | V | V | NA | I | El sistema muestra un mensaje especificando el error, y no se realiza la modificación de los datos. | |
| | V | V | V | I | V | NA | I | El sistema muestra un mensaje especificando el error, y no se realiza la modificación de los datos. | |
| | V | V | V | V | I | NA | I | El sistema muestra un mensaje especificando el error, y no se realiza la modificación de los datos. | |
| | I | I | I | I | I | NA | I | El sistema muestra un mensaje con todos los errores y no se realiza la modificación de los datos. | |
| EC 3.3: Se cancela la modificación. | NA | NA | NA | NA | NA | NA | V | El sistema cancela la modificación de la medida que se había seleccionado | |

**SC4 Eliminar Medidas**

| Escenario | Aceptar | Cancelar | Respuesta del Sistema |
|---|---|---|---|

| | | | |
|---|---|---|---|
| EC 4.1: Se elimina una medida | V | I | Se elimina la medida satisfactoriamente |
| EC 4.2: No se elimina | I | V | No se llega a eliminar, ha sido cancelada |

## 3.4 Conclusiones Parciales

Los diseños de casos de pruebas presentados en el capítulo pertenecen a algunas de las funcionalidades del módulo Registro Minero, escogidas estas para mayor entendimiento de los casos de pruebas, debido a que no son muy complejas, aunque brindan cumplimiento a algunos de los requisitos más importantes señalados por el analista.

# CONCLUSIONES

Una vez concluida la investigación es importante destacar una serie de aspectos que se enumeran a continuación:

1. El diseño propuesto por el analista es claro, simple, orientado al implementador y se corresponde con los patrones de diseño que implementa el framework Symfony.
2. La utilización de implementaciones, componentes y módulos previamente construidos permitió disminuir el tiemplo empleado en la codificación y estandarizar la propuesta con el resto de los módulos que componen el proyecto SGDG.
3. La aplicación desarrollada se rige por normas de diseño y codificación establecidas en el proyecto SGDG.
4. Las herramientas, tecnologías, lenguajes y gestor de base de datos utilizados se corresponden con la premisa de soberanía tecnológica impulsada por la Universidad.
5. Se implementaron todos los requisitos funcionales previstos para la primera versión de este sistema.
6. Los Casos de Prueba diseñados permiten detectar errores tempranos, agilizar y simplificar el flujo de trabajo de pruebas.

Finalmente,

Se obtuvo la documentación técnica apropiada, generada por el implementador en el desarrollo de su rol y se logró una aplicación que luego de ser probada, corregidos posibles errores y finalmente liberada se pondrá a disposición de la ONRM para el control legal de las concesiones mineras y sus concesionarios.

# RECOMENDACIONES

- Continuar con el flujo de trabajo de pruebas con el objetivo de completar la fase de construcción y comenzar la transición en el menor tiempo posible.
- Comenzar con la Fase 2 del SGDG y definir nuevas funcionalidades a implementar.
- Utilizar otras herramientas de trabajo que proporcionen mayor rapidez en etapa de implementación, específicamente usar el IDE de desarrollo Netbeans 6.8 debido a que ya viene integrado con el framework Symfony, facilitando la implementación.

# REFERENCIAS BIBLIOGRÁFICAS

1. **Graells, Dr. Pere Marquès.** [En línea] [Citado el: 2009 de Diciembre de 7.] http://www.pangea.org/peremarques/tic.htm.

2. Ministerio de Relaciones Exteriores de Cuba. *Ministerio de Relaciones Exteriores de Cuba.* [En línea] [Citado el: 2 de Diciembre de 2009.] http://www.cubaminrex.cu/Sociedad_Informacion/Informacion_Gral.htm.

3. **Martell, Ing. Vladimir.** *Proyecto Técnico.* 2008.

4. **Jacobson, Ivar.** *Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000. ISBN:9788478290369.

5. **Rolando Alfredo Hernández León, Sayda Coello González.** *El Paradigma Cuantitativo de la Investigación Científica.* Ciudad de la Habana : EDUNIV Editorial Universitaria, 2002. ISBN: 959-16-0343-6.

6. Portal del sistema de bibliotecas de la UNMSM. [En línea] Universidad Nacional Mayor de San Marcos. http://sisbib.unmsm.edu.pe/bibVirtual/Publicaciones/indata/v04_n1/lenguajes.htm.

7. **Morero, Francisco.** *Introduccion a la OOP.* s.l. : Grupo EIDOS, 1999-2000.

8. **Lanzillotta, Analía.** [En línea] http://www.mastermagazine.info/termino/5560.php .

9. World Wide Web Consortium. [En línea] http://www.w3.org/TR/xhtml1/#xhtml.

10. World Wide Web Consortium. [En línea] http://www.w3.org/Style/CSS/.

11. PHP: Hypertext Preprocessor. [En línea] http://www.php.net/manual/en/faq.general.php.

12. **Ortíz, Antonio.** Error500. [En línea] http://www.error500.net/garbagecollector/archives/categorias/bases_de_datos/sistema_gestor_de_base_de_datos_sgbd.php.

13. PostgreSQL. [En línea] http://www.postgresql.org/about/.

14. Links Global Services. [En línea] http://www.lgs.com.ve/pres/PresentacionES_PSQL.pdf.

15. **Fabien Potencier, François Zaninotto.** *Symfony 1.1, la guía definitiva.*

16. JQUERY. [En línea] http://www.jquery.com.

17. Soluciones y propuestas RATIONAL. [En línea] http://www.rational.com.ar/herramientas/rup.html.

18. **Corporation, IBM.** *Ayuda Rational Unified Process.* 2006.

19. Coders. [En línea] http://www.coders.me/php/codigo-mas-bonito.

20. MSDN Home Page. [En línea] http://msdn.microsoft.com/es-es/library/aa291591%28VS.71%29.aspx.

21. **Pressman, R.** *Software Engineering. A Practitioner's Approach. .* USA : s.n., 1999.

22. Departamento de Ingeniería y Sistemas Telemáticos, Universidad Politécnica de Madrid. *Departamento de Ingeniería y Sistemas Telemáticos, Universidad Politécnica de Madrid.* [En línea] http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm#s1.

# BIBLIOGRAFÍA

1. **Graells, Dr. Pere Marquès.** [En línea] [Citado el: 2009 de Diciembre de 7.] http://www.pangea.org/peremarques/tic.htm.

2. Ministerio de Relaciones Exteriores de Cuba. *Ministerio de Relaciones Exteriores de Cuba.* [En línea] [Citado el: 2 de Diciembre de 2009.] http://www.cubaminrex.cu/Sociedad_Informacion/Informacion_Gral.htm.

3. **Martel, Ing. Vladimir.** *Proyecto Técnico.* 2008.

4. **Jacobson, Ivar.** *Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000. ISBN:9788478290369.

5. **Rolando Alfredo Hernández León, Sayda Coello González.** *El Paradigma Cuantitativo de la Investigación Científica.* Ciudad de la Habana : EDUNIV Editorial Universitaria, 2002. ISBN: 959-16-0343-6.

6. Portal del sistema de bibliotecas de la UNMSM. [En línea] Universidad Nacional Mayor de San Marcos. http://sisbib.unmsm.edu.pe/bibVirtual/Publicaciones/indata/v04_n1/lenguajes.htm.

7. **Morero, Francisco.** *Introduccion a la OOP.* s.l. : Grupo EIDOS, 1999-2000.

8. **Lanzillotta, Analía.** [En línea] http://www.mastermagazine.info/termino/5560.php .

9. World Wide Web Consortium. [En línea] http://www.w3.org/TR/xhtml1/#xhtml.

10. World Wide Web Consortium. [En línea] http://www.w3.org/Style/CSS/.

11. PHP: Hypertext Preprocessor. [En línea] http://www.php.net/manual/en/faq.general.php.

12. **Ortíz, Antonio.** Error500. [En línea] http://www.error500.net/garbagecollector/archives/categorias/bases_de_datos/sistema_gestor_de_base_de_datos_sgbd.php.

13. PostgreSQL. [En línea] http://www.postgresql.org/about/.

14. Links Global Services. [En línea] http://www.lgs.com.ve/pres/PresentacionES_PSQL.pdf.

15. **Fabien Potencier, François Zaninotto.** *Symfony 1.1, la guía definitiva.*

16. JQUERY. [En línea] http://www.jquery.com.

17. Soluciones y propuestas RATIONAL. [En línea] http://www.rational.com.ar/herramientas/rup.html.

18. **Corporation, IBM.** *Ayuda Rational Unified Process.* 2006.

19. Coders. [En línea] http://www.coders.me/php/codigo-mas-bonito.

20. MSDN Home Page. [En línea] http://msdn.microsoft.com/es-es/library/aa291591%28VS.71%29.aspx.

21. **Pressman, R.** *Software Engineering. A Practitioner's Approach. .* USA : s.n., 1999.

22. **León, Lic. Rodrigo Ronda.** *Productos electrónicos: principios y pautas.* 2005.

23. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : Prentice Hall, 1999. ISBN: 970-17-0261-1.

24. WopSolutions. *WopSolutions.* [En línea] [Citado el: 10 de Noviembre de 2009.] http://www.wopsolutions.com/ventajas-de-las-aplicaciones-web.html.

25. Universidad Carlos III de Madrid. *Universidad Carlos III de Madrid.* [En línea] [Citado el: 25 de Noviembre de 2009.] http://www.uc3m.es/portalHelp2/ohw/state/content/locale.es/vtTopicFile.welchelp_hs_es%7Cwelcport~htm/navId.3/navSetId._/.

26. PHP Oficial Home Page. *PHP Oficial Home Page.* [En línea] [Citado el: 7 de Octubre de 2009.] http://www.php.net/manual/en/faq.general.php.

27. Oficina Nacional de Recursos Minerales. *Oficina Nacional de Recursos Minerales.* [En línea] [Citado el: 6 de Diciembre de 2009.] http://www.onrm.minbas.cu/?q=es/node/91.

28. Lenguajes de Programación. *Lenguajes de Programación.* [En línea] [Citado el: 20 de Octubre de 2009.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

29. Instituto Geográfico Agustín Codazzi. [En línea] http://www.igac.gov.co:8080/igac_web/UserFiles/File/ciaf/TutorialSIG_2005_26_02/paginas/ctr_sistemasdegestiondebasededatos.htm.

30. Departamento de Informática de la Universidad de Valladolid. *Departamento de Informática de la Universidad de Valladolid.* [En línea] [Citado el: 3 de Noviembre de 2009.] http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node11.html.

31. CAVSI. [En línea] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

32. Aplicaciones Web. *Aplicaciones Web.* [En línea] [Citado el: 25 de Octubre de 2009.] http://www.aplicacionesweb.net/.

33. Departamento de Ingeniería y Sistemas Telemáticos, Universidad Politécnica de Madrid. *Departamento de Ingeniería y Sistemas Telemáticos, Universidad Politécnica de Madrid.* [En línea] http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm#s1.

# GLOSARIO DE TÉRMINOS

**APLICACION WEB:** En la ingeniería de software se denomina aplicación web a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web en la que se confía la ejecución al navegador. Las aplicaciones web son populares debido a lo práctico del navegador web como cliente ligero, así como a la facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software a miles de usuarios potenciales.

**CASE:** Las herramientas CASE (**C**omputer **A**ided **S**oftware **E**ngineering, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, calculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

**HELPERS:** Son funciones escritas en lenguaje PHP y que son utilizadas por el framework Symfony en su versión 1.1.4 para hacer mas entendible el código de la página web, además que facilita la codificación debido a que pueden traer encapsulado código Javascript, logrando resultados agradables con solo la llamada a una función, ahorrando tiempo y esfuerzo al programador.

**SOFTWARE:** Conjunto de instrucciones y datos codificados para ser leídas e interpretadas por una computadora. Estas instrucciones y datos fueron concebidos para el procesamiento electrónico de datos.

Series de instrucciones codificadas que sirven para que la computadora realice una tarea. Son los programas de la computadora.

**PÁGINA WEB:** Documento o archivo codificado en un lenguaje de marcado, usualmente HTML, que suele contener textos con hipervínculos, imágenes, sonidos, etc., que en la mayoría de los casos reside en un sitio o servidor web, y puede ser consultado empleando un navegador web.

**PATRÓN ARQUITECTÓNICO:** Define la estructura de un sistema software, el cual a su vez se componen de subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema

**PROGRAMA:** Un programa es un conjunto de instrucciones u órdenes basadas en un lenguaje de programación que una computadora interpreta para resolver un problema o una función específica.

# ANEXOS

## Anexo #1 Diferencia entre aplicaciones web con AJAX y clásicas


**Figura 5: Modelo Clásico de aplicaciones web**


**Figura 6: Modelo AJAX de aplicaciones web**

## Anexo # 2 Estándar de Codificación del SGDG

| **Identación**<br><br>**Objetivo:** Lograr una estructura uniforme para los bloques de código así como para los diferentes niveles de anidamiento. | | |
|---|---|---|
| **Inicio y fin de bloque** | | Se recomienda dejar dos espacios en blanco desde la instrucción anterior para el inicio y fin de bloque {}.Lo mismo sucede para el caso de las instrucciones if, else, for, while, switch, foreach. |
| **Aspectos Generales** | El identando debe ser de dos espacios por bloques de código. No se debe usar el tabulador, ya que este puede variar según la PC o la configuración de dicha tecla. | |

| | | |
|---|---|---|
| | Los inicios (**{**) y cierre (**}**) de ámbito deben estar alineados debajo de la declaración a la que pertenecen y deben evitarse si hay solo una instrucción.<br><br>Nunca colocar en la línea de un código cualquiera, esto requiere una línea propia. | |
| **Comentarios, separadores, líneas, espacios en blanco y márgenes.**<br><br>Objetivo: Establecer un modo común para comentar el código de forma tal que sea comprensible con solo leerlo una sola vez. | | |
| **Ubicación de comentarios** | Al inicio de cada clase o función y al final de cada bloque de código. | Se recomienda comentar al inicio de la clase o función especificando el objetivo de la misma así como los parámetros que usa (especificar tipos de datos, y objetivo del parámetro). |
| **Líneas en blanco** | Se emplean antes y después de métodos, clases y estructuras. | Se recomienda dejar una línea en blanco antes y después de las declaraciones de una clase o de una estructura y de la implementación de una función. |
| **Espacios en blanco** | Entre operadores lógicos y aritméticos. | Se recomienda usar espacios en blanco entre estos operadores para lograr una mayor legibilidad en el código. Ejemplo:<br><br>producto = nomproducto |
| **Aspectos Generales** | Sobre el comentario | Se debe evitar comentar cada línea. Cuando el comentario se aplica a un grupo de instrucciones debe estar seguido de una línea en blanco. En caso de que se necesite comentar una sola instrucción se suprime la línea en blanco o se escribe a continuación de la instrucción. |
| | Sobre los espacios en blanco | No se debe usar espacios en blanco. |

| | | |
|---|---|---|
| | | Después del corchete abierto y antes del cerrado de un arreglo.<br><br>Después del paréntesis abierto y antes del cerrado.<br><br>Antes de un punto y coma. |
| **Variables y constantes** | | |
| **Apariencia de constantes** | Todas sus letras en mayúsculas. | Se debe declara todas las constantes con sus letras en mayúsculas. |
| **Aspectos Generales** | Nombres de las variables y las constantes. | El nombre empleado debe permitir que con solo leerlo se conozca el propósito de la misma. |
| **Clases y Objeto**<br><br>**Objetivo:** Nombrar las clases e instancias de forma estándar para todas las aplicaciones. | | |
| **Apariencia de las clases** | Si las clases son generadas por el Framework utilizado se dejan como mismo están. | Los nombres de las clases declaradas por los programadores deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará la notación PascalCasing*.<br><br>Ejemplo: MiClase (). |
| **Apariencia de atributos** | Todas las letras del nombre de los atributos deben ser con minúsculas. | El nombre del atributo debe ser en minúscula y tener una concordancia del valor que debe tener la variable.<br><br>Ejemplo en PHP:<br><br>$contador = 0; |
| **Apariencia de las funciones** | Primera letra en mayúscula para las funciones declaradas con la excepción de las funciones para | Para nombrar las funciones se debe tratar de utilizar verbos que denoten la acción que hace la función. Se empleará |

| | | |
|---|---|---|
| | las acciones y las generadas por el framework. | la notación. PascalCasing*.Ejemplo: function BuscarUnidad ().<br><br>Excepciones en la Notación:<br><br>Si son funciones que obtienen un dato se emplea el prefijo **get** y si fijan algún valor se empleará el prefijo **set**.<br><br>Si son funciones de las clases sfActions se emplea el prefijo **execute**. |
| **Aspectos generales.** | Sobre las clases, los objetos, los atributos y las funciones. | El nombre empleado para las clases, objetos, atributos y funciones debe permitir que con solo leerlo se conozca el propósito de los mismos. |
| **Bases de Datos, Tablas, Esquemas y Campos.** | | |
| **Apariencia de las vistas.** | Las 2 primeras letras representan el tipo. Todas las letras en minúscula. | El nombre a emplear para las vistas deben comenzar con el prefijo **vt** seguido de guión bajo y el nombre debe escribirse con todas las letras en minúscula para evitar problemas con el Case Sensitive del gestor. Ejemplo:<br><br>create view “vt_finanzas”; |
| **Apariencia de las tablas.** | La primera letra representa el tipo. Todas las letras en minúscula. | El nombre a emplear para las tablas debe comenzar con el prefijo **t** seguido por el nombre, en caso de que sea un nombre compuesto se utilizara guión bajo para separarlo. Ejemplo: tproducto o tproducto_comercial. |
| **Tablas que representan Relaciones.** | Las 2 primeras letras representan el tipo. Todas las letras en minúscula. | El nombre a emplear para estas tablas de relación deben comenzar con el prefijo **tr** seguido del nombre que será la concatenación del nombre de las dos |

| | | |
|---|---|---|
| | | tablas que la generaron separados por guión bajo todo en minúscula.<br><br>Ejemplo: “trpaciente_enfermedad” |
| **Tablas que representan nomencladores.** | Las 2 primeras letras representan el tipo. Todas las letras en minúscula. | El nombre a emplear para estas tablas debe comenzar con el prefijo **tn** seguido del nombre en caso que sea un nombre compuesto se utilizara guión bajo para separarlo, descriptivo y todo en minúscula. |
| **Apariencia de los identificadores.** | Las 2 primeras letras indican el identificador. Todo con minúscula. | El nombre del identificador comienza con el prefijo **id** seguido del nombre, en caso de que sea un nombre compuesto se usara guión bajo para separarlo. |
| **Apariencia de los procedimientos almacenados.** | Las 2 primeras letras representan el tipo. Todas las letras en minúscula. | El nombre a emplear para los procedimientos debe comenzar con un prefijo **pa** seguido de guión bajo y luego el nombre con todas las letras en minúscula en caso que sea un nombre compuesto se utilizara guión bajo para separarlo.<br><br>Ejemplo: pa_paciente_especialidad |
| **Sentencias SQL.** | Todas las letras en mayúsculas. | Las palabras correspondientes a las sentencias SQL y sus parámetros deben ir con mayúscula. |
| **Aspectos generales.** | Sobre las vistas, tablas, atributos y procedimientos. | El nombre empleado para las vistas, tablas, los campos y los procedimientos almacenados, deben permitir que con solo leerlos se conozca el propósito de los mismos. |
| **Controles** | | |

| | | |
|---|---|---|
| **Apariencia de los controles.** | Los controles tendrán un prefijo para el tipo de datos en minúscula. | El nombre que se le va a dar a los controles deben comenzar con las primeras letras en minúscula , las cuales identificaran el tipo de datos al que se refiere (Ver tabla 1.1) , en caso de que sea un nombre compuesto se empleará la notación CamellCasing**.<br><br>Ejemplo: btnAceptar. |

**Tabla 1: Estándar de Codificación del SGDG**

| Control | Prefijo | Ejemplo |
|---|---|---|
| **Botón** | btn | btnAceptar |
| **Etiqueta** | lbl | lblNombre |
| **Lista/Menú** | mn | mnPrincipal |
| **Campo de texto** | txt | txtFecha |
| **Botón de opción** | bpt | bptSexo |
| **Checkbox** | chx | chxFuncion |
| **Casilla de selección** | cbx | cbxSexo |
| **Form** | frm | frmIngreso |
| **Submit** | cmb | cmbEnviar |
| **Radio Button** | rad | radOpcion |

**Tabla 2: Componentes visuales**