UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 4

Módulo para el diseño gráfico en el Sistema Integral de Perforación de Pozos SIPP

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autores: Gloria Jean Guirola

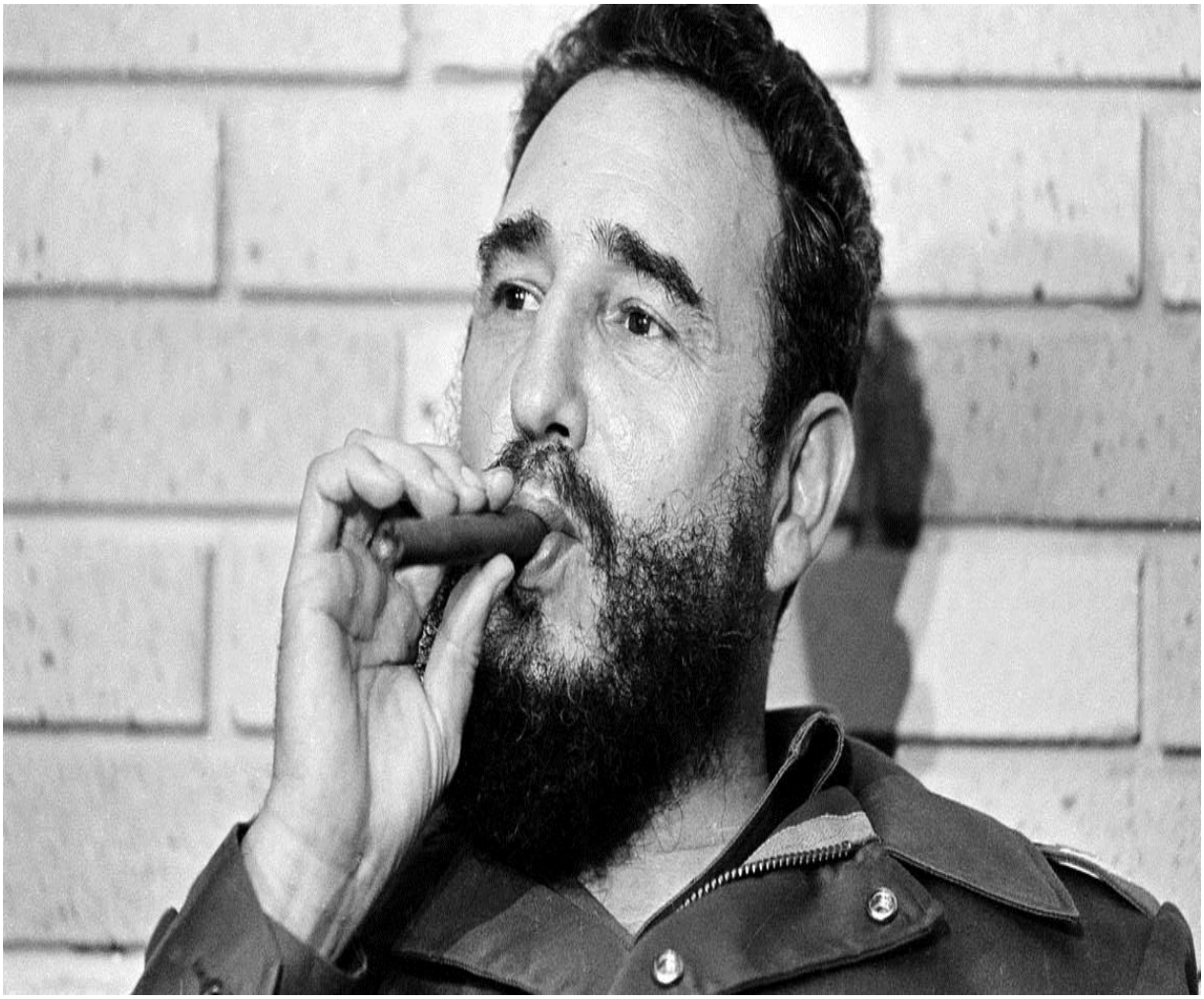
Alejandro Almirola Batista

Tutores: Ing. Carlos Heriberto Cordoví

Ing. José Antonio López

Ing. Dailyn García Domínguez

Curso docente: 2016-2017.



"El tema relativo al conocimiento y la tecnología es de especial relieve en nuestra agenda, porque en él abordamos los problemas que deciden, en buena medida, el futuro de nuestros países."

Fidel Castro Ruz

Declaración de Autoría

Declaramos ser los autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste se firma la presente a los _____ días del mes de _____ del año _____.

Firma del autor	Firma de los tutores
Gloria Jean Guirola	**Carlos Heriberto Cordoví**
Alejandro Almirola Batista	**José Antonio López** **Dailyn García Dominguez**

AGRADECIMIENTOS

A mi mamá por darme la vida, quererme mucho, creer en mí y porque siempre me apoyaste. Por haberme dado la oportunidad de tener una excelente educación en el transcurso de mi vida. Sobre todo, por ser un excelente ejemplo de vida a seguir. Mami gracias por darme una carrera para mi futuro, todo esto te lo debo a ti.

A mi papá por apoyarme en todo momento y por los valores que me has inculcado, esto también te lo debo a ti.

A mi madrina por quererme y apoyarme siempre, y por estar a mi lado en el momento más duro de mi carrera, te quiero muchísimo, aunque a veces no te lo demuestre.

A mi familia, en especial a mi prima Yusimí, por siempre estar a mi lado.

A mis amigos que más que amigos son mis hermanos Lilibeth, Yanelis, Jaime, Armando, Fernando, Lamis, Yasiel y Pepe, por aconsejarme durante estos cinco años de la carrera en mis momentos de tristeza a seguir adelante y no mirar hacia atrás. A ustedes de corazón muchísimas gracias por todo.

A todos mis amigos del aula y de la escuela, en especial a mi compañero de tesis que a pesar de que no estuvo desde el comienzo de la tesis a mi lado, al final logramos este título juntos.

A mis tutores por estar presente en todo momento, un millón de gracias, porque este título también es de ustedes, se lo merecen.

A mis profesores por la buena formación que me dieron y por todo su apoyo, en especial a Yirka por ser un buen ejemplo para mí, gracias, y a Sailyn por ser una buena jefa de año.

A Andrea y Yaima por escuchar en estos cinco años todas mis manifestaciones de rebeldía cuando quería dejar la escuela, y por siempre darme un buen consejo, a ambas muchísimas gracias por estar presente.

Y a esta escuela que sin ella hoy no me estuviera graduando de Ingeniera en Ciencias Informáticas, en especial a mi facultad 5, a Yadira, a Rigo, a Mayra y a todos los que contribuyeron a este resultado, millones de gracias

Gloria

Gracias a mi Familia, en especial a mi madre por ayudarme a nunca rendirme, por ser tan especial conmigo y hacer de mí la persona que soy hoy en día, a mi padre y mi hermano por haberme apoyado en todo durante los años de mi carrera y a lo largo de mi vida.

Gracias a aquellas amistades que siempre me han apoyado y han estado conmigo en momentos difíciles de mi vida que por ende también se han ganado estar en los momentos felices. Los quiero mucho.

Gracias a mi compañera de tesis y amiga por su apoyo, dedicación y confianza. Te quiero.

Gracias a los tutores por haberme guiado durante todo el proceso de elaboración y preparación de nuestro Trabajo de Diploma.

Gracias a todos mis profesores por los conocimientos compartidos y por los consejos que en alguna que otra ocasión me aportaron.

Alejandro

DEDICATORIAS

A mis padres por ser el pilar fundamental en todo lo que soy, en toda mi educación, tanto académica, como de la vida, por su incondicional apoyo perfectamente mantenido a través del tiempo.

Todo este trabajo ha sido posible gracias a ellos.

Gloria Jean Guirola

Con el más profundo de los respeto y admiración hoy dedico a mis padres y a mi hermano, quienes han sido el principal iniciador de mis grandes sueños y dándome apoyo incondicional. A ustedes hoy queridos darles mi sincero agradecimiento por ser mi sustento, por ser mi compañía y por siempre estar, deseándome continuamente lo mejor, gracias por cada momento vivido.

Alejandro Almirola
Batista

RESUMEN

En la Universidad de las Ciencias Informáticas, donde existe una red de centros productivos se encuentra el Centro de Informática Industrial. Dicho centro se encarga de desarrollar productos y servicios informáticos de automatización industrial. En este se desarrolla el Sistema Integral de Perforación de Pozos, el cual ha sido concertado entre la universidad y la División de Integración y Perforación de Pozos. El Sistema Integral de Perforación de los Pozos es capaz de exportar los documentos referentes a la construcción del pozo, pero los mismos carecen de representaciones gráficas, lo que representa una deficiencia en cuanto a la completitud y calidad del reporte. También genera un conjunto de reportes que requieren de la incorporación de varios componentes que representan el esquema de construcción del pozo por intervalos y secciones. El objetivo es desarrollar un módulo gráfico para el Sistema Integral de Perforación de Pozos que permita la representación gráfica de la construcción de los reportes que lo requieran. Con la implementación de dicho módulo se garantizará la representación de esquemas de construcción de pozos de manera intuitiva y amigable para el usuario final.

Palabras clave: diseño gráfico, esquemas, inspector de propiedades, perforación de los pozos, reportes.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El desarrollo de las actividades socioeconómicas está condicionado por la introducción y aplicación de las Tecnologías de la Información y las Comunicaciones (TICs) en los procesos económicos y productivos. Los nuevos adelantos científicos han impulsado la evolución de las TICs y su consecuente aplicación en la industria ha potenciado el florecimiento productivo y económico. La industria del software ha tomado auge en las últimas décadas, muchas empresas exitosas dependen de sistemas informáticos para la gestión de sus procesos sustantivos.

La Industria del Petróleo no queda ajena a este proceso; La Unión Cuba-Petróleo (CUPET) es la unión de empresas responsable del desarrollo y mantenimiento de la industria del petróleo en Cuba. La División de Integración y Perforación de Pozos (DIPP) es uno de los organismos encargado de la perforación de pozos de petróleo en el país. La DIPP gestiona toda la información referente al proceso de perforación. Por tanto, requiere hacer un levantamiento diario de las actividades del proceso de perforación. Con esta finalidad se hace necesario el uso de reportes que brinden una vista integrada de la información que cada pozo genera.

La industria del software también se introduce en la industria petrolera como herramienta para el análisis y la toma de decisiones. En la Universidad de las Ciencias Informáticas (UCI), donde existe una red de centros productivos que trabajan diversas líneas de investigación, se encuentra el Centro de Informática Industrial (CEDIN). El centro se encarga de desarrollar productos y servicios informáticos de automatización industrial, con un alto valor agregado y que cumplan las necesidades y expectativas de los clientes, potenciando la formación especializada y la investigación.

Uno de los proyectos que se desarrolla en dicho centro es el Sistema Integral de Perforación de Pozos (SIPP). Este sistema ha sido concertado entre la DIPP y la UCI, en aras de informatizar los procesos relacionados con la perforación de pozos de petróleo que supervisa la DIPP.

El sistema SIPP es capaz de exportar los documentos referentes a la construcción del pozo, pero los mismos carecen de representaciones gráficas de la construcción del pozo, lo que representa una deficiencia en cuanto a la completitud y calidad de los reportes que requieren estas representaciones. También genera un conjunto de reportes que requieren

de la incorporación de varios componentes que representan gráficamente la construcción del pozo por intervalos y secciones (para simplificar serán llamados esquemas).

Estos reportes se generan de acuerdo a modelos específicos basados en resoluciones dictadas al efecto y estas resoluciones demandan la inclusión de dichos esquemas en los reportes. Cada esquema debe estar asociado a un reporte en específico y deben persistir físicamente en base de datos, pues necesitan replicarse a la DIPP.

Sin embargo, los reportes generados actualmente por el sistema sólo muestran información relacionada con la construcción del pozo, excluyendo los esquemas que son necesarios en cada reporte fundamentalmente los de: Encamisado, Resumen de Construcción de Pozos y de Terminación y Ensayo (Esquema de completamiento del pozo).

Por otro lado, no es posible realizar estos esquemas en Excel e importar estos al sistema porque la biblioteca de algoritmos que utiliza SIPP para el trabajo con Excel: *phpExcel* no permite exportar esquemas en formato vectorial. Además, no resulta factible realizar estos esquemas desde otras herramientas e importarlos a los reportes. En su lugar es más eficiente construirlo sobre el mismo sistema.

A partir de la situación descrita anteriormente se define el siguiente **problema a resolver**

¿Cómo representar gráficamente la construcción de pozos desde el SIPP, de manera intuitiva y amigable para el usuario final, garantizando su inclusión en los reportes generados por el sistema?

El objeto de estudio para esta investigación son los sistemas de gestión para edición gráfica.

Para dar solución al problema planteado se propone como **objetivo general** el desarrollo de un módulo gráfico para el SIPP que permita representar de forma gráfica la construcción de pozos de petróleo para los reportes generados por el sistema.

El **campo de acción** se enmarca al módulo de edición gráfica para el diseño de pozos de petróleo en el SIPP.

Como **idea a defender** se tiene que, la implementación de un módulo gráfico para el SIPP permitirá representar gráficamente la construcción de pozos de petróleo de manera intuitiva y amigable para el usuario final.

Para darle cumplimiento a los objetivos trazados y proponer una solución al problema planteado se proponen las siguientes **tareas de investigación**:

1) Elaborar el marco teórico conceptual relacionado con los aspectos teóricos que sustentan la investigación.

2) Realizar el análisis y diseño de la herramienta.

3) Desarrollar una herramienta informática como soporte al modelo.

4) Realizar pruebas funcionales a la herramienta.

5) Validar la herramienta propuesta a través de los métodos definidos en la investigación.

Para el desarrollo de la propuesta de solución se utilizaron los siguientes métodos Científicos:

Métodos teóricos

Analítico-Sintético: El uso de este método facilitó realizar un estudio de la bibliografía existente referida al tema de los sistemas de gestión para la edición gráfica. A partir del análisis realizado, fue posible conocer las distintas herramientas y tecnologías utilizadas para el desarrollo de aplicaciones de este tipo, además de sistematizar los elementos comunes de acuerdo al análisis de varios editores gráficos convergen en los siguientes elementos: la paleta de componentes, el inspector de objetos y un inspector de propiedades.

Modelación: El uso de este método hizo posible realizar un modelo del módulo de diseño gráfico de la herramienta mediante la representación de las diferentes características y funcionalidades de la solución a la problemática planteada, de forma tal que representa cómo quedará estructurada la vista final para que el usuario pueda interactuar con cada uno de los esquemas que necesite.

Histórico- lógico: Se realizó un estudio lógico y cronológico de los sistemas que permiten la edición gráfica, esto permitió ver cómo han evolucionado para determinar las características y elementos fundamentales de los sistemas para la edición gráfica en la actualidad.

Métodos Empíricos

Observación: Con la utilización de este método se obtuvo información cualitativa y cuantitativa de la usabilidad acerca del módulo gráfico desarrollado durante el proceso de interacción de los usuarios finales con la propuesta de solución.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

El presente capítulo tiene como objetivo abordar los aspectos teóricos fundamentales que facilitan la comprensión y el desarrollo de la propuesta de solución. Se describen algunas herramientas que se utilizan en el desarrollo de aplicaciones web. Se realiza un estudio sobre los principales conceptos asociados al dominio de los sistemas de gestión para la edición gráfica y que son necesarios para comprender el desarrollo de la propuesta de solución. Se realiza además un análisis sobre las herramientas de diseño gráfico. Se presenta también un resumen sobre los lenguajes de modelado, herramientas y tecnologías utilizadas en la presente investigación.

1.2 Diseño gráfico

(Baldasano, 1989) Plantea que hace tan sólo 47 años, casi todos los dibujos se ejecutaban utilizando lápiz y papel. Cuando se precisaba realizar cambios, era necesario borrar y volver a dibujar. La llegada de las computadoras facilitó la creación de herramientas de dibujo.

(Contreras, 2001) define el diseño gráfico como una forma comunicativa susceptible de transformarse en obra de arte en cuanto intervienen en ella ciertas virtudes excepcionales como la originalidad conceptual o la innovación. Entendido como una actividad comunicativa, industrial, proyectual y funcional.

Por otro lado, (Samara, 2015) especifica que el diseño gráfico y la publicidad comparten un objetivo común: informar al público de los bienes, servicios, acontecimientos o ideas que alguien cree que les pueden interesar. Pero el diseño gráfico se distingue de la publicidad en lo que respecta a su objetivo: la publicidad, una vez que ha informado al público de su producto o espectáculo, intenta engatusarlo para que gaste dinero. Sin embargo, el diseño gráfico busca únicamente aclarar el mensaje y transformarlo en una experiencia emocional.

De acuerdo a los objetivos de la propuesta de solución y tomando como referencia a (Contreras, 2001) y (Samara, 2015) se define al diseño gráfico como una actividad comunicativa, industrial, proyectual y funcional. En la actualidad los planos y prototipos

industriales se construyen en los sistemas de gestión haciendo uso de diseños automatizados utilizando la edición gráfica, de acuerdo a un propósito específico.

1.3 Sistema de gestión para el diseño gráfico

(Pfaff, 1983) define un sistema de gestión como herramientas de software cuya función es permitir la especificación, implementación y mantenimiento de una interfaz usuario-computadora que sea orientada al usuario y adaptable.

Por su parte (Ubiquitour, 2014) detalla que es una forma de software de aplicación para la edición, creación, manipulación y visualización de archivos gráficos. Estos programas permiten la visualización y modificación de fotografías y creación de medios animados o digitales. Los dos tipos principales de software de gráficos de función son: *raster graphics* o gráficos vectoriales.

Partiendo de estas dos definiciones y de acuerdo a los objetivos propuestos se asume que un sistema de gestión para el diseño gráfico es un conjunto de componentes gráficos directamente relacionados los cuales están integrados para satisfacer un propósito común. Los sistemas de gestión gráfica contienen módulos de diseño gráfico donde se realiza el prototipado y la edición de componentes gráficos.

- **Módulo gráfico**

En programación, un módulo es un software que agrupa un conjunto de subprogramas y estructuras de datos. Los módulos son unidades que pueden ser compiladas por separado y los hace reusables y permite que múltiples programadores trabajen en diferentes módulos en forma simultánea, produciendo ahorro en los tiempos de desarrollo (Alegsa,2010).
Los módulos promueven la modularidad y el encapsulamiento, pudiendo generar programas complejos de fácil comprensión (Alegsa, 2010).

Teniendo en cuenta la definición de módulo expuesta anteriormente se puede decir que un módulo gráfico está enfocado al diseño gráfico. La revisión de la literatura referente a este tema ha demostrado que los módulos gráficos poseen cuatro componentes fundamentales, los cuales son:

Canvas: Es una etiqueta html (sigla en inglés de Lenguaje de Marcas de HiperTexto) que revela el área de trabajo donde se realizan los componentes, predefinida por un ancho y un alto (**Ver Fig. 1**).

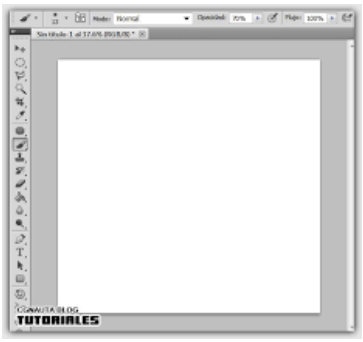


Fig.1: Canvas

Paleta de componentes: La paleta de componentes de un editor gráfico de aplicaciones compuestas, muestra componentes que pueden añadirse dinámicamente a una aplicación para resolver una necesidad empresarial. Puede controlar qué componentes gráficos se muestran y cómo están organizados (**Ver Fig. 2**).

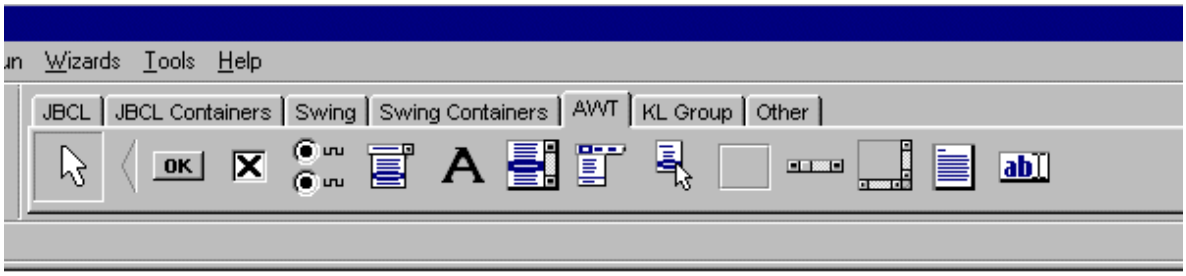


Fig.2: Paleta de componentes

Inspector de propiedades: Son los elementos del componente que configuran su aspecto y controlan su comportamiento (**Ver Fig. 3**). Las propiedades de los componentes pueden establecerse y modificarse en tiempo de diseño a componentes específicos.

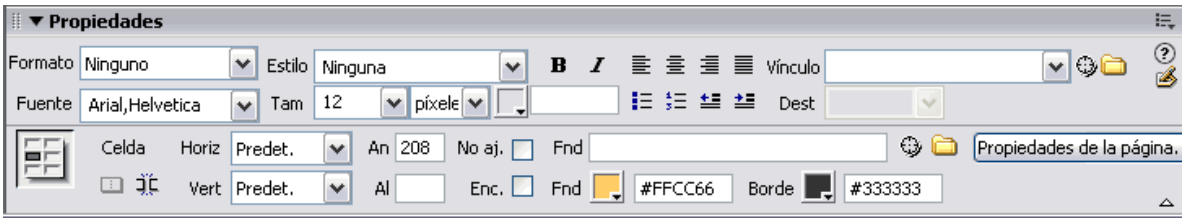


Fig.3: Inspector de propiedades

Inspector de objetos: Contiene varias características del objeto necesarias para conformar cada uno de los esquemas (**Ver Fig. 4**).

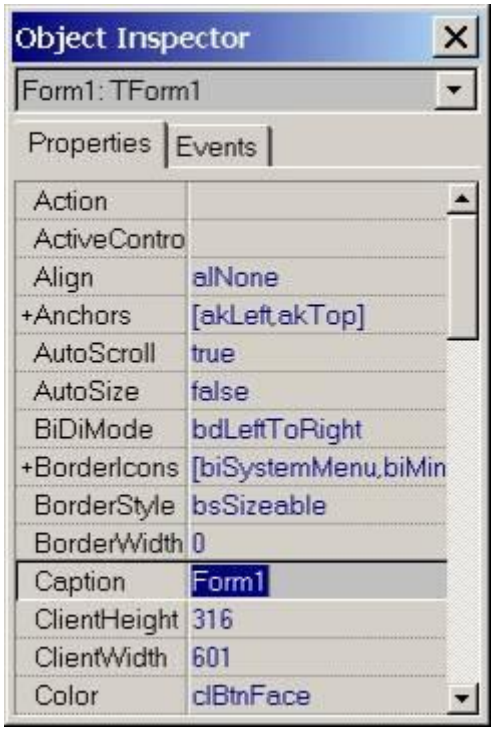


Fig.4: Inspector de objetos

Una vez especificados estos tres componentes fundamentales y el área de trabajo. El módulo gráfico va a permitir generar esquemas para su posterior incorporación en los reportes.

1.4 Reportes

El reporte contiene información obtenida sobre el evento elegido de forma organizada y relacionada de una manera que muestra aspectos de una materia específica. Algunos reportes del SIPP contienen información de cada uno de los esquemas de construcción del pozo de petróleo. Se especifican las variaciones y características que presentan los pozos en cada uno de sus intervalos y sesiones.

SIPP genera varios reportes para la gestión de los procesos en la DIPP y los pozos. De ellos específicamente tres requieren de la incorporación de varios esquemas para su completitud. Los reportes donde se deben incluir estos esquemas son:

- Reporte de resumen de construcción de pozos: Muestra dos características fundamentales: Plan y Real. El Plan modela cómo se planifica la construcción del pozo por intervalos y sesiones, mientras que Real especifica cómo queda estructurado realmente el pozo una vez construido.
- Reporte de Encamisado: Muestra todas las actividades que se realizan durante el proceso de encamisado. Además, muestra información sobre el zapato, la válvula, los torques mínimos y máximos, etc.
- Reporte de terminación y ensayo: Se realiza en la etapa de terminación, y muestra información a través de un esquema que muestra cómo queda el pozo una vez listo para su entrada en explotación.

Para garantizar la calidad los reportes, estos deben cumplir con parámetros como:

- Claridad.
- Coherencia.
- Precisión.
- Debe tener un orden lógico.
- Por último, deben tener un buen diseño y organización de todas sus propiedades.

A continuación, se enumeran las restricciones fundamentales:

- Escalado: Adaptación de un componente a una escala o marca.
- Pixelado: Es un efecto causado por visualizar un componente o una imagen a un tamaño en el que los píxeles individuales son visibles al ojo.
- Degradado: Un degradado es un rango de colores ordenados linealmente con la intención de dar visualmente una transición suave y progresiva entre dos o más colores (Eisenberg, 2002).

1.5 Fabric.js

Es una poderosa biblioteca de JavaScript que trabaja con lienzos HTML5. Es un proyecto de código abierto, con licencia bajo el MIT (sigla en inglés del Instituto Tecnológico de Massachusetts), con muchas contribuciones a lo largo de los años. Permite crear y rellenar objetos en lienzo (canvas), como formas geométricas y formas complejas. En Fabric, se opera sobre objetos, se instancian, se cambian sus propiedades y se añaden a lienzos. Facilita el uso del trazado libre, manipulación de grupos y objetos y la agrupación.

Proporciona un modelo de objeto sencillo pero potente sobre los métodos nativos. Se ocupa del estado del lienzo y la representación, y nos permite trabajar con "objetos"

directamente. Con Fabric, no es necesario borrar el contenido antes de intentar "modificar" cualquier contenido, simplemente cambiando sus propiedades, y luego volver a hacer el lienzo para obtener una "imagen nueva". Se encarga de la representación en el lienzo y de la gerencia del estado del objeto (Fabriq.js, s.f.).

1.6 Base tecnológica del proyecto SIPP

- **Visual Paradigm**

(Leiva Pereira, Yusemí; 2012) expresa en su tesis que Visual Paradigm es una herramienta de Ingeniería de Software Asistida por Computación (CASE). La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque orientado a objetos. Se integra con diferentes entornos de desarrollo como Eclipse y NetBeans y proporciona código y compatibilidad con diferentes lenguajes como C++, PHP, Java y Python. Dentro de sus características principales se encuentran:

- Disponibilidad en múltiples plataformas (Windows, Linux).
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Soporta aplicaciones Web.
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.

Visual Paradigm para UML (sigla en inglés de Lenguaje Unificado de Modelado) en su versión 5.0 es la herramienta seleccionada para modelar la solución ya que se puede ejecutar sobre plataforma Linux. Permite la ingeniería en ambos sentidos y es compatible con varios lenguajes como C++ y PHP. Además, presenta una interfaz de usuario de fácil uso para realizar los diagramas y artefactos que se generan durante el desarrollo del software.

- **Symfony como marco de desarrollo:**

El *framework* de desarrollo Symfony 2.7 permite optimizar 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 a los desarrolladores varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. También, automatiza las tareas más comunes, permitiendo a los desarrolladores dedicarse por completo a los aspectos específicos de cada aplicación. Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel, cuenta con la característica de ser multiplataforma (Francois Zaninotto, 2016).

- **PHP como lenguaje de programación**

PHP 5.5.11 (*Hypertext Preprocessor*), es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor. Ofrece una solución simple y universal para las páginas Web. Su diseño elegante lo hace perceptiblemente más fácil de mantener y ponerse al día en el código comparado con otros lenguajes. Debido a su amplia distribución PHP está perfectamente soportado por una gran comunidad de desarrolladores (Leiva Pereira, Yusemí; 2012). PHP es:

- Un lenguaje multiplataforma.
- Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- El código fuente escrito en PHP no visible al navegador ni al cliente.
- De gran capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad.
- De amplia documentación en su página oficial.
- Libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Un lenguaje que permite aplicar técnicas de programación orientada a objetos.
- Una biblioteca nativa de funciones sumamente amplia e incluida.

Es escogido PHP para la programación, ya que es el lenguaje en que está desarrollado completamente Symfony.

- **Jquery como marco de trabajo de JavaScript:**

JQuery es una biblioteca JavaScript destinada a hacer programación de JavaScript. Una biblioteca de JavaScript es un conjunto complejo de código JavaScript que simplifica las tareas difíciles y resuelve los problemas entre navegadores. En otras palabras, JQuery resuelve los dos principales dolores de cabeza de JavaScript: la complejidad y la naturaleza mezquina de los diferentes navegadores web. Con Jquery se puede realizar tareas en una sola línea de código que podría tener cientos de líneas de programación y muchas horas de pruebas de navegador para lograr con su propio código JavaScript. Permite agregar funciones avanzadas al sitio web con miles de plugins fáciles de usar. Por ejemplo, el complemento de la interfaz de usuario de JQuery le permite crear muchos elementos complejos de la interfaz de usuario como paneles con pestañas, menús desplegables, calendarios de selección de fecha emergentes, todos con una sola línea de programación (McFarland, 2014).

- **PostgreSQL**

PostgreSQL es uno de los Sistemas de Gestión de Bases de Datos Objeto Relacionales (ORDBMS) de código abierto más avanzado del mundo. PostgreSQL posee muchas características que normalmente sólo se encontraban en las bases de datos comerciales tales como DB2 u Oracle (Korry Douglas, 2005). Es un Sistema de gestión de bases de datos relacional orientado a objetos y libre, publicado bajo la licencia PostgreSQL (Group, 2015).

- **Metodología**

El Proceso Unificado Ágil (AUP) según sus siglas en inglés, establece un modelo más simple que el que aparece en el Proceso Unificado Racional (RUP) según sus siglas en inglés, por lo que reúne en una única disciplina las disciplinas de Modelado de Negocio, Requisitos y Análisis y Diseño. El resto de disciplinas (Implementación, Pruebas, Despliegue, Gestión de Configuración, Gestión y Entorno) coinciden con las restantes de RUP (Donatien, 2009).

1.7 Conclusiones Parciales

Se realizó un estudio sobre Fabrics.js y se decide emplearla para el diseño de la propuesta de solución, atendiendo a que es una poderosa biblioteca JavaScript de código abierto que trabaja con lienzos HTML5. Permite crear componentes específicos y figuras complejas en el CANVAS, elementos muy necesarios para la propuesta de solución. Además, el análisis de la literatura existente proyectó que en los sistemas de gestión gráfica son indispensables las siguientes utilidades: el inspector de objeto, la paleta de componentes y el inspector de propiedades.

CAPÍTULO II: Análisis y diseño de la propuesta de solución

2.1 Introducción

El presente capítulo hace referencia a los principales elementos que contribuyen a la construcción de la propuesta de solución. Se evidencian los conceptos del negocio mediante el mapa conceptual, se determinan las funcionalidades y las características del módulo a desarrollar puntualizando los requisitos funcionales y no funcionales, se muestran las distintas historias de usuario. Además, se proporciona un acercamiento más detallado al sistema mostrando la arquitectura propuesta. También se presenta el diagrama de clases con la descripción de cada una de ellas para comprender el módulo desarrollado, así como los patrones de diseño empleados en la propuesta.

2.2 Mapa Conceptual

Un mapa conceptual es una red de conceptos ordenados jerárquicamente, esto quiere decir que los conceptos de mayor generalidad ocuparán los espacios superiores. El mapa conceptual puede ser elaborado a partir de notas, para ordenar y representar los conocimientos que las personas tienen respecto a un tema o para representar conocimientos y teorías (Tamayo, 2006). El uso de los mapas conceptuales permite organizar y comprender ideas de manera significativa. Se representan como una red que contiene nodos los cuales se refieren a las entidades y las líneas representan las relaciones entre los conceptos. (**Ver Fig.5**).

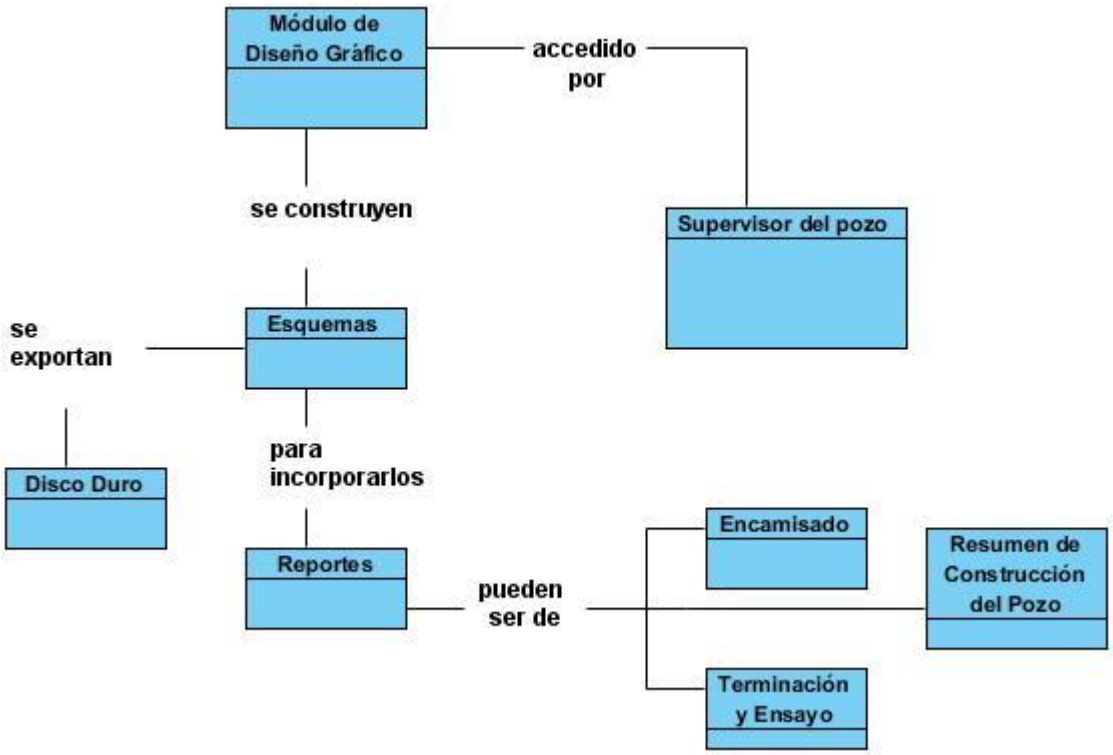


Fig.5: Mapa conceptual

2.2.1 Conceptos principales del sistema

Módulo: Es una parte del software del SIPP, su objetivo fundamental es permitir la gestión de esquemas de construcción del pozo para su inclusión en los reportes generados por el sistema teniendo en cuenta que dichos esquemas se pueden guardar en una dirección física del disco duro de la computadora (acción de exportar). Todo el proceso debe de ser de forma intuitiva y amigable para el usuario final.

Supervisor del Pozo: Persona que analiza, examina y supervisa diariamente los parámetros recibidos del pozo en perforación, confeccionando con estos datos una serie de reportes, los cuales se utilizan posteriormente en la DIPP.

Disco Duro: Se refiere a una dirección física dentro del disco duro de la computadora en la cual se pueden gestionar datos, en este caso para copiar los esquemas construidos.

2.3 Especificación de los Requisitos del Software.

La especificación de requisitos tiene como objetivo esclarecer de forma eficiente las funcionalidades y restricciones con que debe contar el sistema atendiendo a las necesidades de los clientes. Por lo que es de suma importancia contactar con los mismos

para almacenar qué es exactamente lo que desean que contenga la propuesta a desarrollar.

El resultado de una buena calidad en la definición de los requisitos eleva el nivel de automatización ya que da un mejor soporte a una gestión de cambios que pueda atravesar el sistema, reduce el riego de no conformidades dentro de la fase de pruebas, aporta a la toma de mejores decisiones de diseño y ayuda a reducir los problemas de mantenimiento del software. El levantamiento de los requisitos que se especifica dentro de la propuesta de solución se efectuó mediante reuniones con los distintos directivos que contiene el proyecto de desarrollo SIPP. Se muestran definidos en dos grupos:

2.3.1 Requisitos Funcionales

Son condiciones que el sistema debe cumplir, por lo que el sistema debe permitir:

TABLA I: REQUISITOS FUNCIONALES

Nº	Nombre	Descripción	Complejidad	Prioridad para el cliente	Referencias cruzadas
RF1	Integrar los componentes gráficos para construir los esquemas desde la paleta de componente.	El sistema debe permitir integrar los componentes en formato vectorial para ampliar la paleta de componentes a partir de la extensión de la biblioteca Fabrics.js con estos nuevos componentes.	Alta	Alta	HU #1

RF2	Generar el inspector de objeto.	El sistema debe permitir mostrar dinámicamente las propiedades que contiene el objeto que tiene el foco, es decir, el objeto que se encuentra activo en ese instante de tiempo.	Media	Alta	HU #2
RF3	Generar el inspector de propiedades.	El sistema debe permitir modificar las propiedades que contiene el objeto que se encuentre activo.	Alta	Alta	HU #3
RF4	Ajustar dinámicamente la escena de trabajo.	El sistema debe permitir que los componentes que conforman el esquema de construcción del pozo no se salgan del marco o del área de trabajo (objeto canvas), así como ajustarse a las dimensiones de la pantalla en la que se ejecuta el programa.	Alta	Alta	HU #4
RF5	Crear el esquema de construcción de pozos.	El sistema debe permitir insertar el esquema de construcción del pozo con los componentes que lo conforman. De manera que persista en base de datos para luego ser replicado a la DIPP.	Alta	Alta	HU #5

RF6	Mostrar el esquema de construcción de pozos.	El sistema debe permitir cargar el esquema de construcción del pozo con los componentes que lo conforman desde la base de datos en el objeto CANVAS.	Alta	Alta	HU #6
RF7	Modificar el esquema de construcción de pozos.	El sistema debe permitir modificar el esquema de construcción del pozo adicionándole nuevos componentes o eliminándole componentes que contiene, además de los cambios que se le puedan producir a los componentes mediante el inspector de propiedades. De manera que persista en base de datos los cambios efectuados.	Alta	Alta	HU #7
RF8	Eliminar el esquema de construcción de pozos.	El sistema debe permitir eliminar de la base de datos uno o varios esquemas específicos de construcción del pozo.	Alta	Alta	HU #8

RF9	Integrar el esquema de construcción del pozo en la plantilla de reporte Excel.	El sistema debe permitir insertar el esquema de construcción del pozo en el lugar que indica la plantilla Excel de cada reporte.	Alta	Alta	HU #9
RF10	Exportar la imagen del esquema de construcción del pozo	El sistema debe permitir exportar una imagen que contenga el esquema, en el formato que seleccione el usuario.	Media	Alta	HU #10

2.4 Requisitos no Funcionales

Son propiedades que el sistema debe tener, por lo que el módulo desarrollado debe contar con características como:

- Usabilidad
- Mantenibilidad
- Adecuación funcional

TABLA II: DESCRIPCIÓN DEL REQUISITO NO FUNCIONAL DE USABILIDAD

Atributo de Calidad	Usabilidad.
Sub-atributos/Sub-característica	Operabilidad.

Objetivo	Capacidad del producto que permite al usuario operarlo y controlarlo con facilidad.
Origen	Proveedor de requisitos.
Artefacto	Las computadoras que contengan el software elaborado por el SIPP.
Entorno	El software desplegado.
Estímulo **1. a. Número de pasos a ejecutar para llegar al módulo de diseño gráfico**	**Respuesta: Flujo de eventos (Escenarios)**
Cargar la interfaz principal de SIPP.	1. El sistema muestra la pantalla principal para poder observar los esquemas. 2. El usuario selecciona la opción deseada. 3. La aplicación verifica la opción seleccionada y responde según la opción deseada a crear, mostrar, modificar o eliminar los esquemas. 4. El usuario construye, modifica o elimina el esquema atendiendo a su necesidad para visualizar el mismo en el reporte luego de generarlo.
Medida de respuesta	
Visualizar las gráficas.	

TABLA III: TABLA DE REQUISITO NO FUNCIONAL DE USABILIDAD-AMIGABILIDAD

Atributo de Calidad	Usabilidad.

Sub-atributos/Sub-característica	Amigabilidad.
Objetivo	Capacidad del producto que permite al usuario interactuar con el módulo mediante interfaces agradables e intuitivas.
Origen	Proveedor de requisitos.
Artefacto	Las computadoras que contengan el software elaborado por el SIPP.
Entorno	El software desplegado.
Estímulo **1. a. Ambiente gráfico de interacción con los esquemas.**	**Respuesta: Flujo de eventos (Escenarios)**
Gestiona los esquemas con sus respectivos componentes mediante sus interfaces.	1. Selecciona las vistas correspondientes a las acciones sobre los esquemas. 2. Muestra las interfaces correspondientes para la interacción con el usuario.
Medida de respuesta	
Visualizar las gráficas.	

TABLA IV: DESCRIPCIÓN DEL REQUISITO NO FUNCIONAL DE MANTENIBILIDAD

Atributo de Calidad	Mantenibilidad.
Sub-atributos/Sub-característica	Cumplimiento de Mantenibilidad.

Objetivo	Capacidad del producto que permite que sea modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño.
Origen	Arquitecto de software.
Artefacto	El código fuente.
Entorno	El ambiente de desarrollo del módulo.
Estímulo **1. a. Capacidad de modificación del módulo**	**Respuesta: Flujo de eventos (Escenarios)**
La arquitectura del módulo está diseñada para brindar facilidades a la hora de visualizar la información brindada. Esto permite que se puedan introducir cambios o modificaciones, que no afecten el correcto desempeño del resto de las funcionalidades de la solución.	N/A
Medida de respuesta	
Introducir una modificación al módulo.	

TABLA V: DESCRIPCIÓN DEL REQUISITO NO FUNCIONAL DE ADECUACIÓN FUNCIONAL

Atributo de Calidad	Adecuación funcional.
Sub-atributos/Sub-característica	Integridad funcional.
Objetivo	Grado en el que el conjunto de funciones cubre todas las tareas y objetivos del usuario especificado.
Origen	Proveedor de requisitos.

Artefacto	El módulo.
Entorno	El módulo desplegado.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a. Grado en el que el módulo cubre todas las tareas y objetivos especificados	
El desarrollo del módulo está guiado por las necesidades expresadas por parte de los proveedores del sistema, dándole total cumplimiento a sus especificaciones declaradas e implícitas.	N/A
Medida de respuesta	
Analizar las funcionalidades del módulo.	

2.4.1 Descripción de los actores

El actor del sistema es quien interactúa o hace uso del sistema (el usuario).

TABLA VI: ACTOR DEL SISTEMA

Actor	Justificación
Especialista (Supervisor del Pozo y Directivo)	Es el encargado de seleccionar en el menú Generador el indicador referente a la construcción del esquema del pozo.

2.4.2 Historias de Usuarios del Sistema

Se utilizan para especificar los requisitos de los sistemas informáticos en las metodologías ágiles. Las historias de usuario son tarjetas que describen brevemente (con el fin de que sean dinámicas y flexibles) las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. Cada historia de usuario debe ser lo suficientemente comprensible y delimitada para que se pueda implementar en el mejor

tiempo posible. Las historias de usuario se descomponen en tareas de programación que los programadores implementan. (Suaza, 2014)

2.5 Descripción de las Historias de Usuarios del Sistema

TABLA VII: HU #1: INTEGRAR LOS COMPONENTES GRÁFICOS PARA CONSTRUIR LOS ESQUEMAS DESDE LA PALETA DE COMPONENTE.

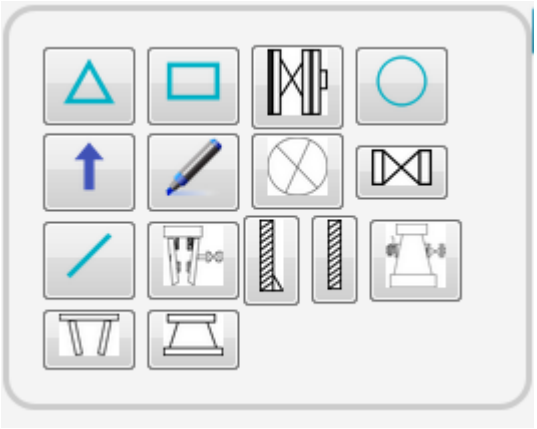
<table>
<tr><th colspan="3">Historia de Usuario</th></tr>
<tr><td>Número: RF1</td><td colspan="2">Nombre del requisito: Integrar los componentes gráficos para construir los esquemas desde la paleta de componentes.</td></tr>
<tr><td colspan="2">Programador: Gloria Jean Guirola y Alejandro Almirola Batista.</td><td>Iteración Asignada: 3</td></tr>
<tr><td colspan="2">Prioridad: Alta</td><td>Tiempo Estimado: 96 horas</td></tr>
<tr><td colspan="2">Riesgo en Desarrollo: Riesgo definido en el Gespro.</td><td>Tiempo Real: 94 horas</td></tr>
<tr><td colspan="3">Descripción: El sistema debe permitir integrar los componentes en formato vectorial para ampliar la paleta de componentes a partir de la extensión de la biblioteca Fabrics.js con estos nuevos componentes en formato de gráficos vectoriales (SVG).</td></tr>
<tr><td colspan="3">Observaciones:</td></tr>
<tr><td colspan="3">Prototipo de interfaz:
  </td></tr>
</table>

TABLA VIII: HU #2: GENERAR EL INSPECTOR DE OBJETO.

Historia de Usuario	
Número: RF2	**Nombre del requisito:** Generar el inspector de objetos.
Programador: Gloria Jean Guirola y Alejandro Almirola Batista.	**Iteración Asignada:** 3
Prioridad: Alta	**Tiempo Estimado:** 24 horas
Riesgo en Desarrollo: Riesgo definido en el Gespro.	**Tiempo Real:** 14 horas

Descripción: El sistema debe permitir mostrar dinámicamente las propiedades que contiene el o los objetos que se encuentran activos en ese instante de tiempo.

- Se debe construir un inspector de objeto que muestre al usuario las propiedades que contiene uno o varios objetos o componentes que conforman el esquema en el escenario de dibujo seleccionado, de forma tal que al desplazar el o los objetos o aplicarle alguna transformación muestre las propiedades actuales de los objetos activos en ese instante.

Observaciones:

Prototipo de interfaz:

TABLA IX: HU #3: GENERAR EL INSPECTOR DE PROPIEDADES.

Historia de Usuario	
Número: RF3	**Nombre del requisito:** Generar el inspector de propiedades.
Programador: Gloria Jean Guirola y Alejandro Almirola Batista.	**Iteración Asignada:** 3
Prioridad: Alta	**Tiempo Estimado:** 48 horas
Riesgo en Desarrollo: Riesgo definido en el Gespro.	**Tiempo Real:** 45 horas
Descripción: El sistema debe permitir modificar las propiedades de: color, color del borde, tamaño, gradiente, ancho y alto del objeto activo.	
Observaciones:	
Prototipo de interfaz: 	

TABLA X: HU #4: AJUSTAR DINÁMICAMENTE LA ESCENA DE TRABAJO.

Historia de Usuario	
Número: RF4	**Nombre del requisito:** Ajustar dinámicamente la escena de trabajo.

Programador: Gloria Jean Guirola y Alejandro Almirola Batista.	**Iteración Asignada:** 3
Prioridad: Alta	**Tiempo Estimado:** 24 horas
Riesgo en Desarrollo: Riesgo definido en el Gespro.	**Tiempo Real:** 5 horas

Descripción: El sistema debe permitir que los componentes que conforman el esquema de construcción del pozo no se salgan del marco o del área de trabajo (objeto CANVAS), así como ajustarse a las dimensiones de la pantalla en la que se ejecuta el programa.

Observaciones:

Prototipo de interfaz:

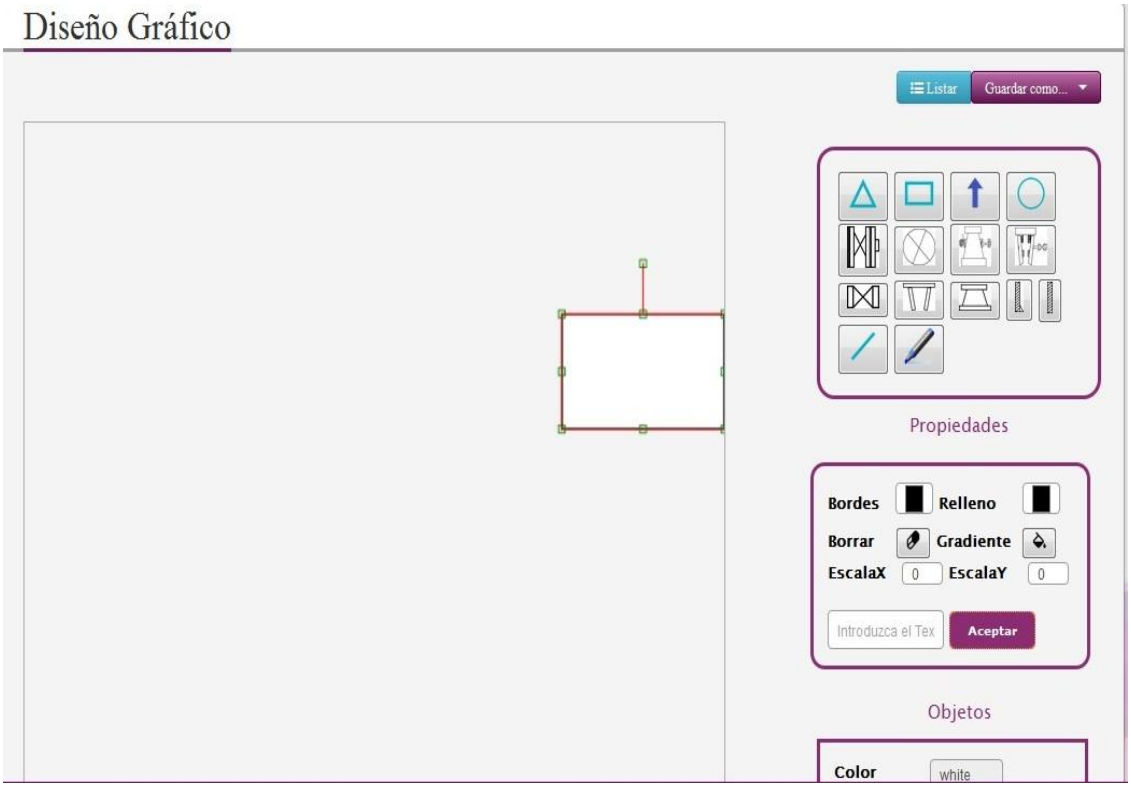


TABLA XI: HU #5: CREAR EL ESQUEMA DE CONSTRUCCIÓN DE POZOS.

Historia de Usuario	
Número: RF5	**Nombre del requisito:** Crear el esquema de construcción de pozos.
Programador: Gloria Jean Guirola y Alejandro Almirola Batista.	**Iteración Asignada:** 3
Prioridad: Alta	**Tiempo Estimado:** 24 horas
Riesgo en Desarrollo: Riesgo definido en el Gespro.	**Tiempo Real:** 11 horas
Descripción: El sistema debe permitir insertar el esquema de construcción del pozo con los componentes que lo conforman. De manera que persista en base de datos para luego ser replicado a la DIPP.	
Observaciones:	
Prototipo de interfaz:	

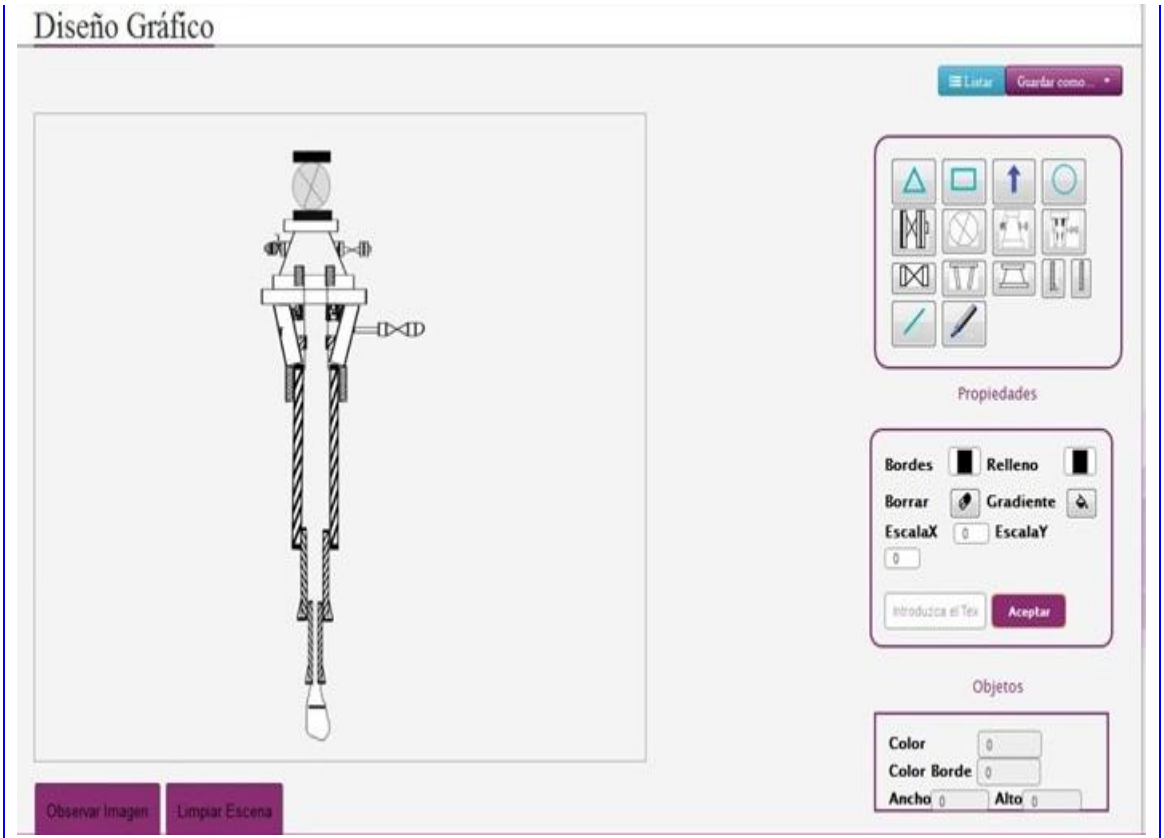


TABLA XII: HU #6: MOSTRAR EL ESQUEMA DE CONSTRUCCIÓN DE POZOS.

Historia de Usuario	
Número: RF6	**Nombre del requisito:** Mostrar el esquema de construcción de pozos.
Programador: Alejandro Almirola Batista y Gloria Jean Guirola	**Iteración Asignada:** 3
Prioridad: Alta	**Tiempo Estimado:** 24 horas
Riesgo en Desarrollo: Riesgo definido en el Gespro.	**Tiempo Real:** 23 horas
Descripción: El sistema debe permitir cargar el esquema de construcción del pozo con los componentes que lo conforman desde la base de datos en el objeto CANVAS.	

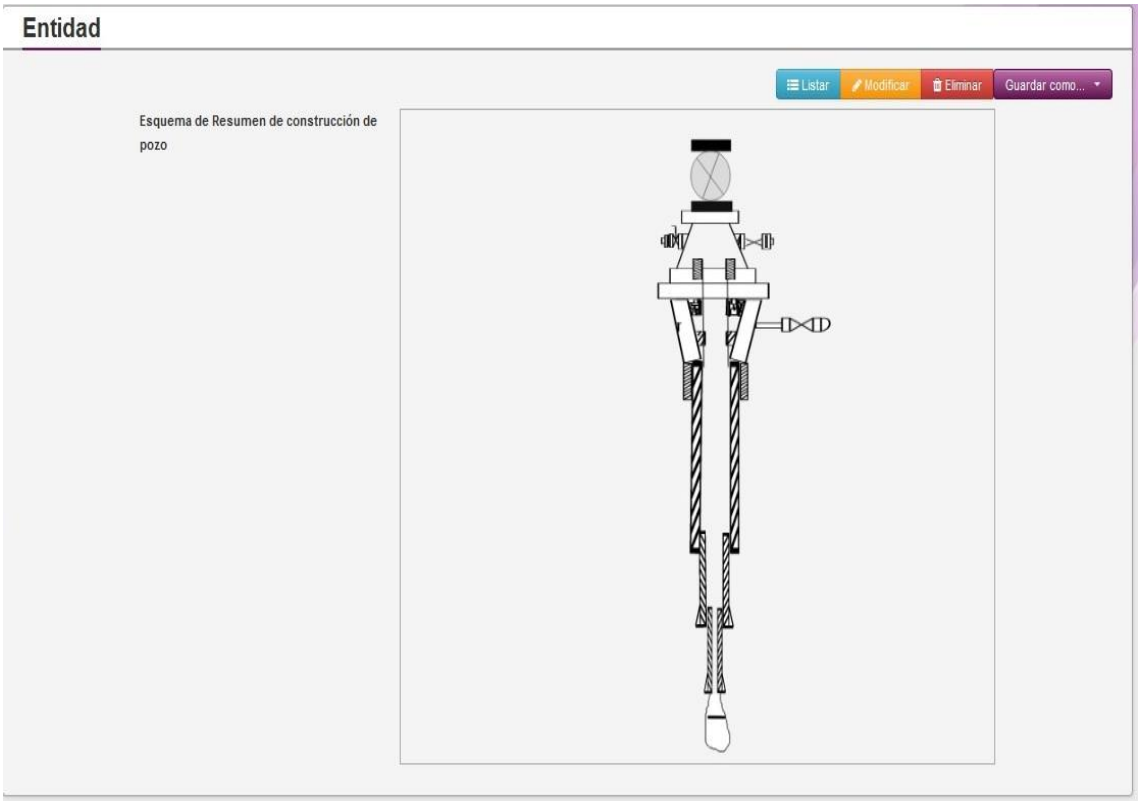
Observaciones:
Prototipo de interfaz: 

TABLA XIII: HU #7: MODIFICAR EL ESQUEMA DE CONSTRUCCIÓN DE POZOS.

Historia de Usuario	
Número: RF7	**Nombre del requisito:** Modificar el esquema de construcción de pozos.
Programador: Alejandro Almirola Batista y Gloria Jean Guirola	**Iteración Asignada:** 3
Prioridad: Alta	**Tiempo Estimado:** 48 horas
Riesgo en Desarrollo: Riesgo definido en el Gespro.	**Tiempo Real:** 47 horas

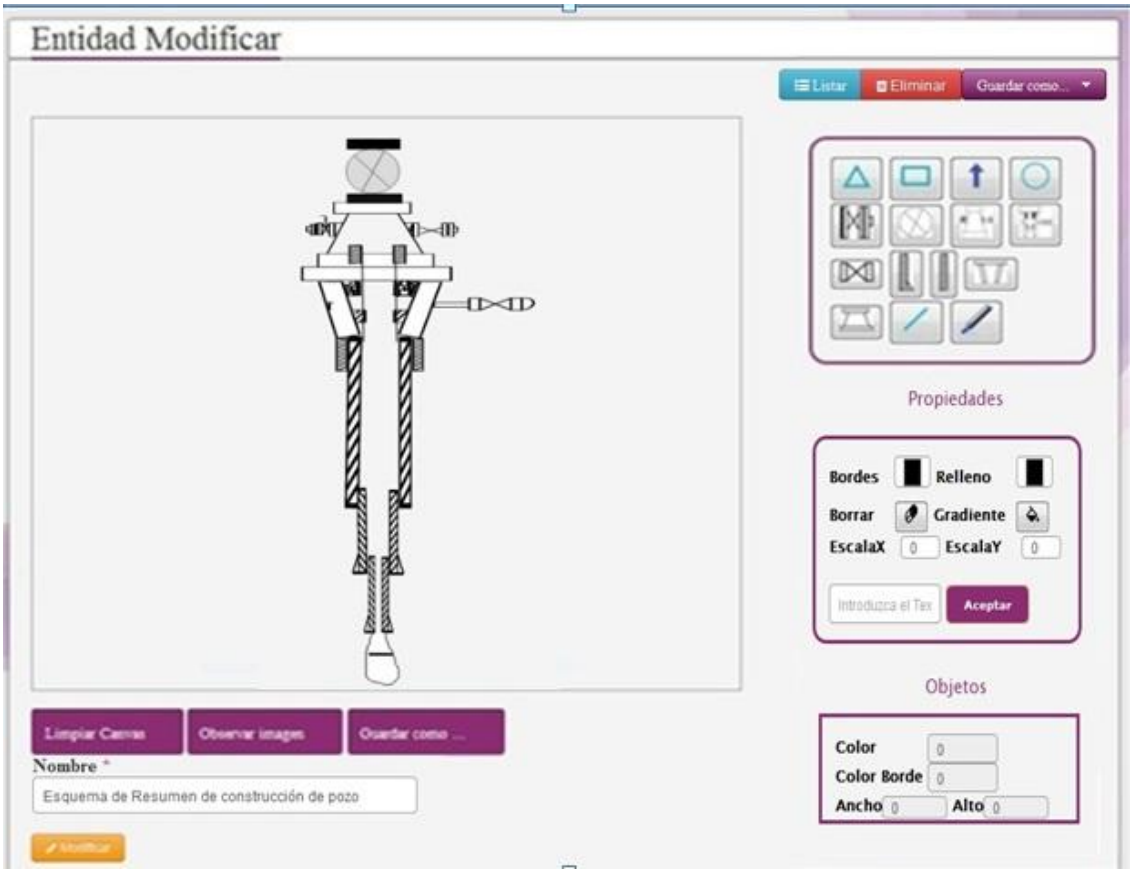
Descripción: El sistema debe permitir modificar el esquema de construcción del pozo adicionándole nuevos componentes o eliminándole componentes que contiene, además de los cambios que se le puedan producir a los componentes mediante el inspector de propiedades. De manera que persista en base de datos los cambios efectuados.
Observaciones:
Prototipo de interfaz: 

TABLA XIV: HU #8: ELIMINAR EL ESQUEMA DE CONSTRUCCIÓN DE POZOS.

Historia de Usuario	
Número: RF8	**Nombre del requisito:** Eliminar el esquema de construcción de pozos.
Programador: Alejandro Almirola Batista y Gloria Jean Guirola	**Iteración Asignada:** 3

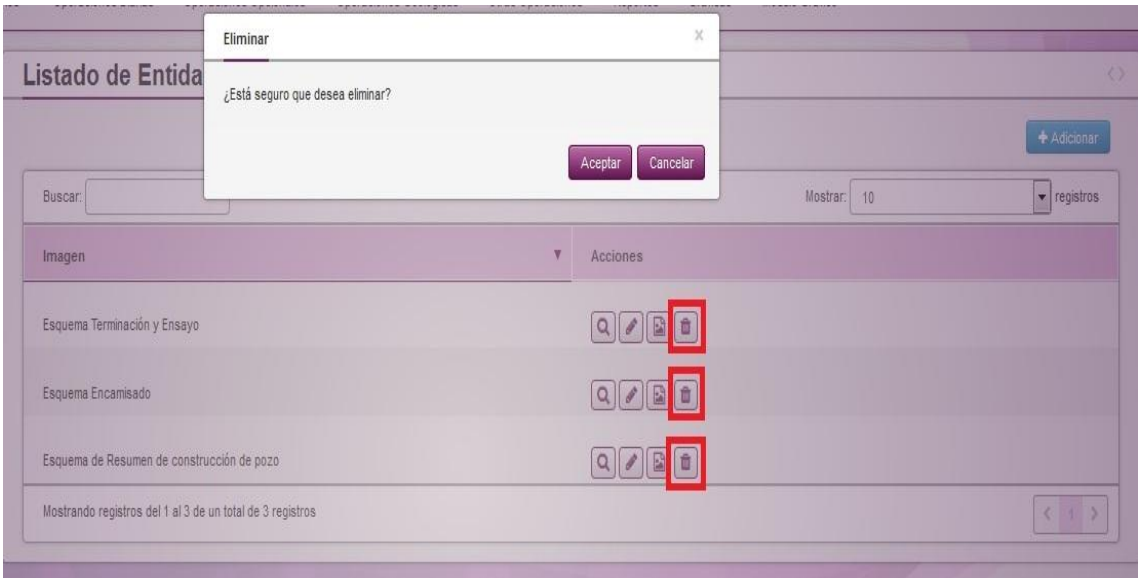
Prioridad: Alta	**Tiempo Estimado:** 24 horas
Riesgo en Desarrollo: Riesgo definido en el Gespro.	**Tiempo Real:** 14 horas
Descripción: El sistema debe permitir eliminar de la base de datos uno o varios esquemas específicos de construcción del pozo. Antes debe mostrar un mensaje de aseguramiento.	
Observaciones:	
Prototipo de interfaz: 	

TABLA XV: HU #9: INTEGRAR EL ESQUEMA DE CONSTRUCCIÓN DEL POZO EN LA PLANTILLA DE REPORTE EXCEL.

Historia de Usuario	
Número: RF9	**Nombre del requisito:** Integrar el esquema de construcción del pozo en la plantilla de reporte Excel.
Programador: Alejandro Almirola Batista y Gloria Jean Guirola	**Iteración Asignada:** 3
Prioridad: Alta	**Tiempo Estimado:** 64 horas

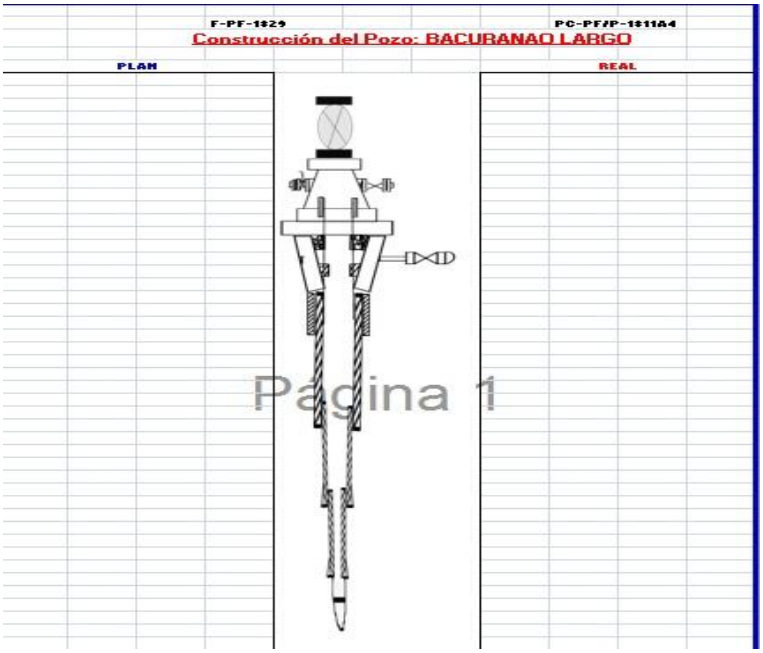
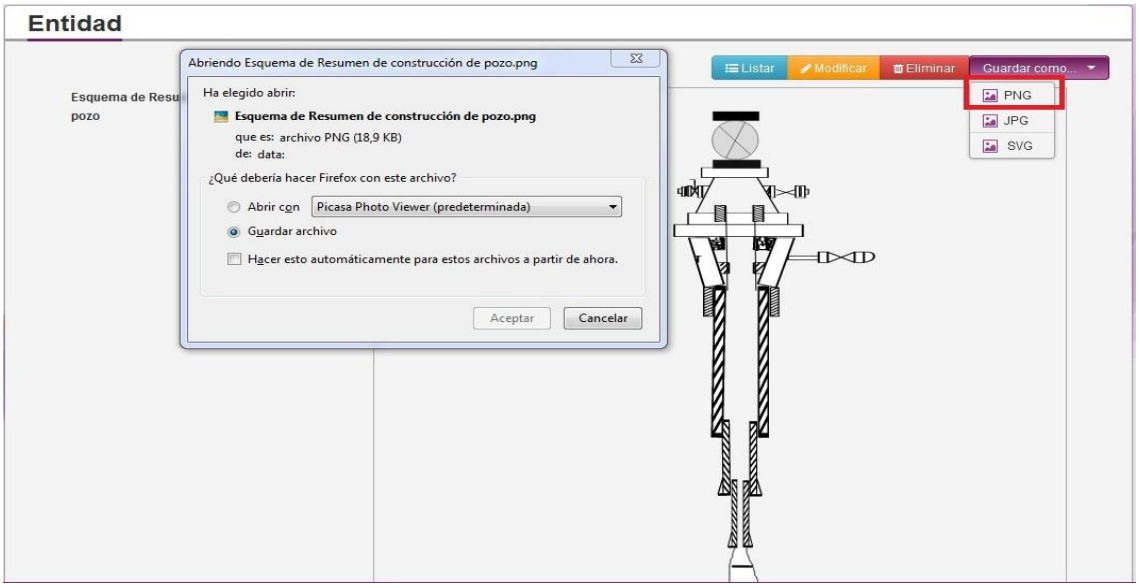
Riesgo en Desarrollo: Riesgo definido en el Gespro.	**Tiempo Real:** 58 horas
Descripción: El sistema debe permitir insertar el esquema de construcción del pozo en el lugar que indica la plantilla Excel de cada reporte.	
Observaciones:	
Prototipo de interfaz: 	

TABLA XVI: HU #10: EXPORTAR LA IMAGEN DEL ESQUEMA DE CONSTRUCCIÓN DEL POZO.

Historia de Usuario	
Número: RF10	**Nombre del requisito:** Exportar la imagen del esquema de construcción del pozo.
Programador: Alejandro Almirola Batista y Gloria Jean Guirola	**Iteración Asignada:** 2

Prioridad: Alta	**Tiempo Estimado:** 24 horas

Riesgo en Desarrollo: Riesgo definido en el Gespro.	**Tiempo Real:** 22 horas
Descripción: El sistema debe permitir exportar una imagen que contenga el esquema, en el formato que seleccione el usuario (JPG, PNG, SVG) a una dirección física del disco duro de la computadora.	
Observaciones:	
Prototipo de interfaz: 	

2.6 Representación de los Diagramas de Clases

Symfony está basado en el patrón arquitectónico Modelo Vista Controlador (MVC). Por lo que los diagramas de clases del diseño fueron realizados según su patrón, obteniendo así una visión general del diseño de la aplicación **(Ver Fig.6)**.

Diagrama de Clases del Gestionar Esquema (CRUD).

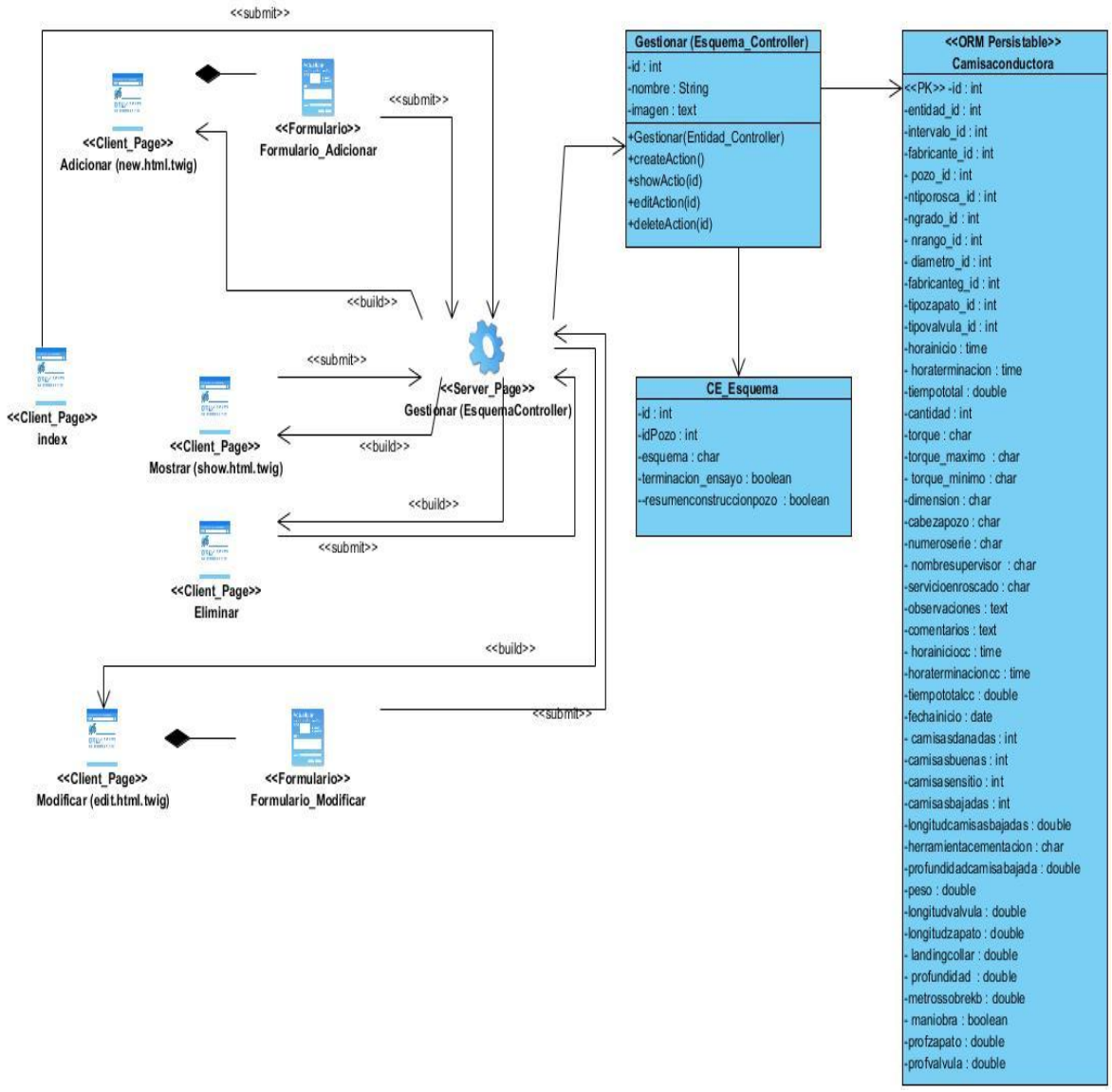


Fig.6: Diagrama de clases del Gestionar Esquema (CRUD).

2.7 Patrón arquitectónico

Los patrones arquitectónicos se utilizan para expresar una estructura de organización base o esquema para un software. Proporcionando un conjunto de sub-sistemas predefinidos, especificando sus responsabilidades, reglas, directrices que determinan la organización, comunicación, interacción y relaciones entre ellos.

Los patrones arquitectónicos heredan mucha de la terminología y conceptos de patrones de diseño, pero se centran en proporcionar modelos y métodos re-utilizables

específicamente para la arquitectura general de los sistemas de información (Gómez A. , 2013).

2.7.1 Patrón Arquitectónico Modelo-Vista-Controlador (MVC)

El patrón MVC dota a la aplicación de su estructura “más visible”, y ayuda al programador de aplicaciones web con Symfony a colocar cada cosa en su sitio y a construir aplicaciones altamente modulares, extensibles, portables, mantenibles y, sobre todo, “vivas” durante mucho tiempo (Rodríguez, 2012).

Dicho patrón divide la aplicación en tres tipos de elementos:

El modelo: Este permite realizar una representación de la lógica del negocio y en él se encuentra la clase esquema como entidad, clase que contiene toda la información que requiere el módulo.

La vista: Es la encargada de notificar al controlador que una solicitud ha sido hecha por parte del usuario y este a su vez le envía la información obtenida a otras clases las cuales corresponden a las vistas del módulo desarrollado.

El controlador: En este va a estar la clase EsquemaController.php la cual va a servir de mediador entre las vistas y el modelo, además de permitir manejar las funcionalidades del sistema descritas en cada Historia de Usuario.

El patrón MVC establece una relación entre las clases que manejan la información del negocio (entidades) y las clases correspondientes a la interfaz de usuario (vistas), las cuales mantienen un flujo a través del controlador, que permite mostrar al usuario los resultados de su solicitud (**Ver Fig.7**).

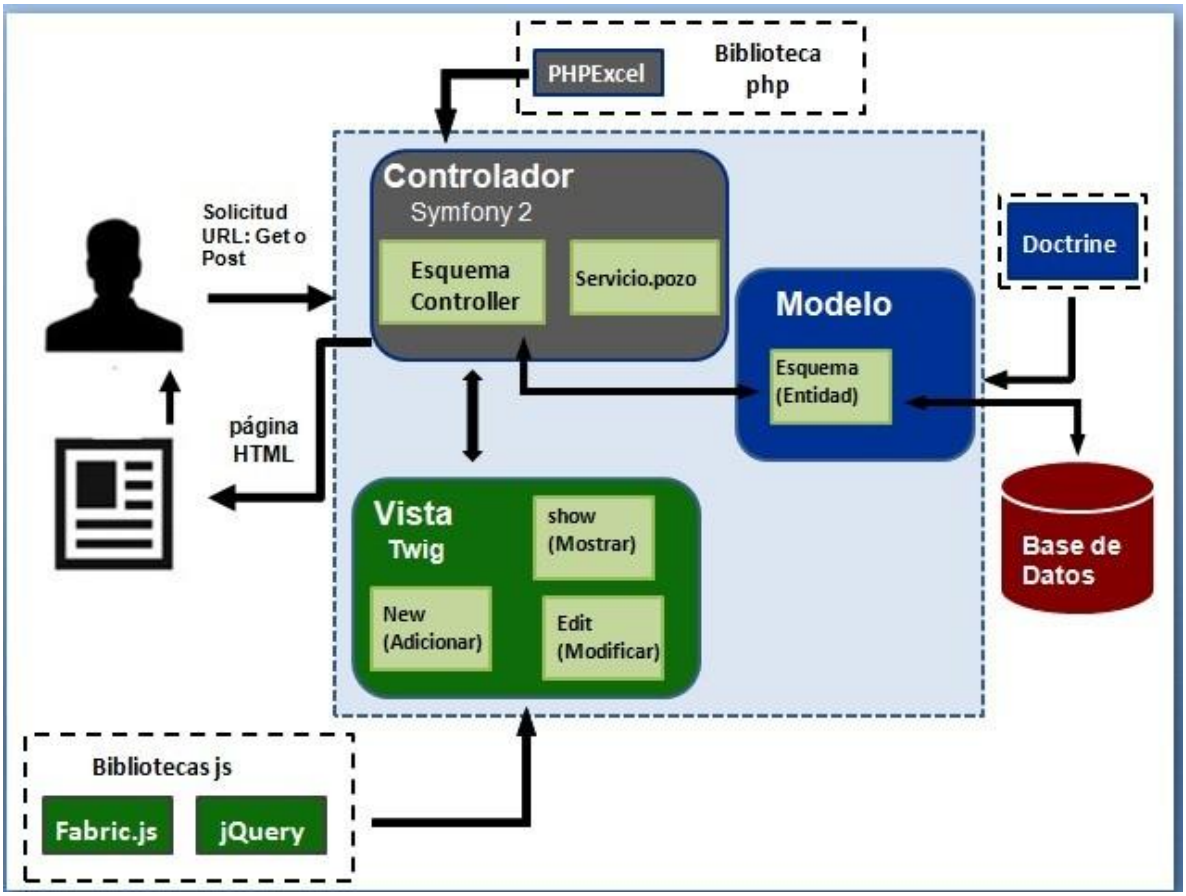


Fig.7: Patrón Arquitectónico (MVC).

2.8 Diagrama de paquetes

El diagrama de paquetes representa la interacción de los subpaquetes y las clases del módulo, este contiene las relaciones entre ellas, así como los atributos y métodos que componen cada una (Barrera, 2013).

A continuación, se muestra una visión general del diseño de la aplicación siguiendo la estructura del patrón arquitectónico Modelo Vista Controlador (MVC) en el que está basado Symfony. El diagrama modela cómo están distribuidas las clases con estereotipos web, la relación que existe entre ellas y los paquetes en que se encuentran. Además de reflejar cómo interactúa el controlador con los datos del modelo para actualizar las vistas atendiendo a las peticiones de estas. **(Ver Fig.8)**.

Diagrama de Paquetes

Diagrama de Paquetes con clases correspondientes al módulo "Diseño Gráfico"

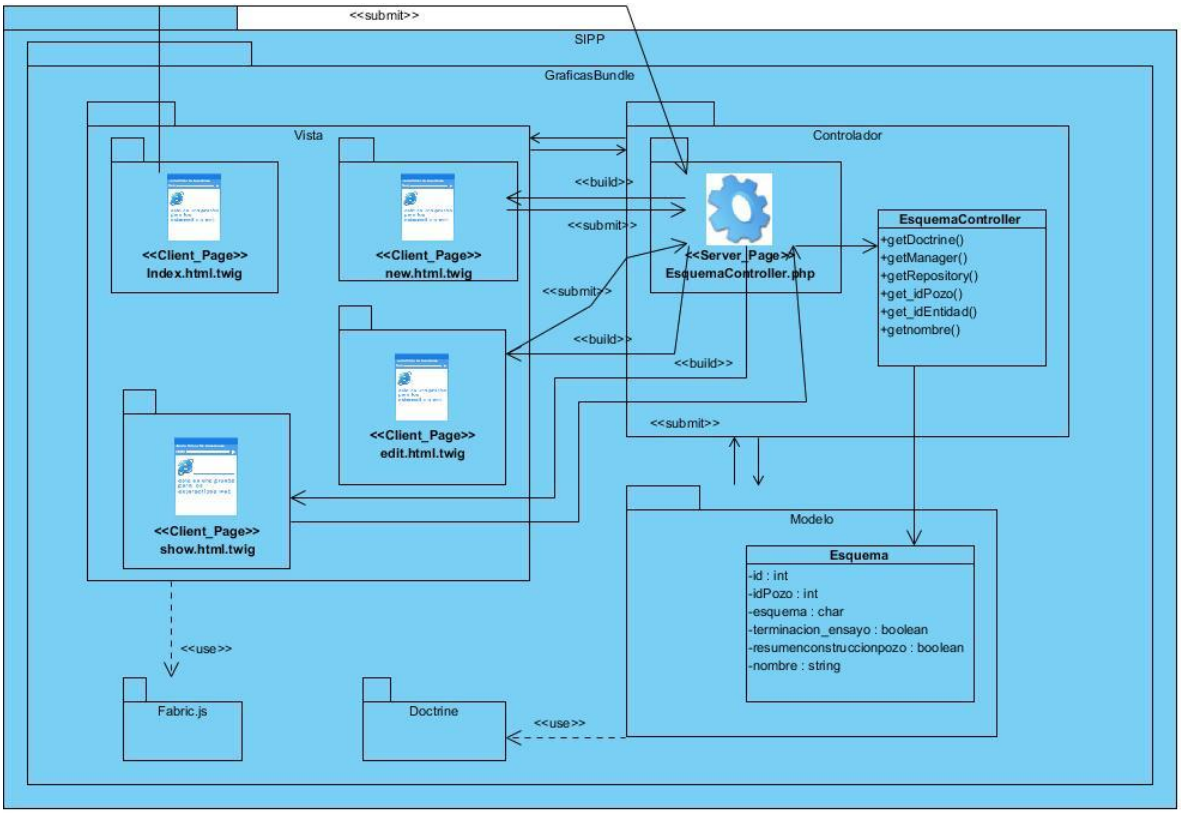


Fig.8: Diagrama de Paquetes con clases correspondientes al módulo ¨Diseño Gráfico¨.

2.9 Modelo de Datos

El modelo de datos es el conjunto de reglas y conceptos que permiten describir y manipular los datos que se desean almacenar en la base de datos. Es una colección de conceptos bien definidos matemáticamente que ayudan a expresar las propiedades estáticas y dinámicas de una aplicación con un uso de datos intensivo (Fabre, 2013). Partiendo del diagrama de clases persistentes (Ver Anexo 1) se muestra la relación que existe entre las clases necesarias para la persistencia de los datos en las mismas, dando lugar al diagrama Entidad Relación de la Base de Datos **(Ver Fig.9)**.

Diagrama de Entidad Relación de la Base de Datos

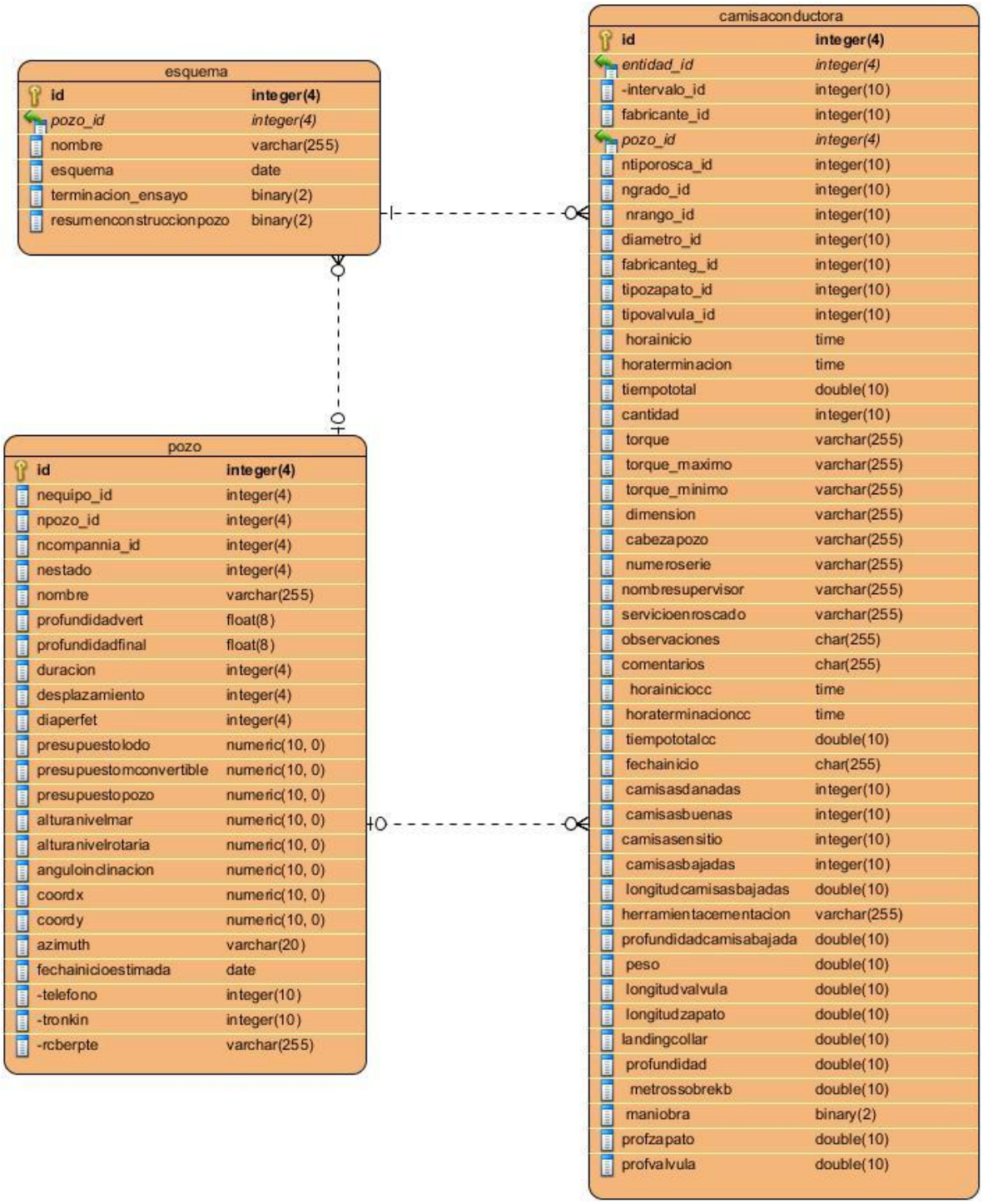


Fig.9: Diagrama de Entidad Relación de la Base de Datos

2.10 Conclusiones Parciales

Después de haber realizado el análisis y diseño de la propuesta de solución se concluye que:

- A través del mapa conceptual se concibió el sistema a desarrollar mediante la representación de los principales conceptos que intervienen en el módulo.
- El diseño de la relación entre los datos facilitó que los programadores obtuvieran los datos de las tablas para el desarrollo del módulo, e incluyeran una nueva entidad en la Base de Datos del SIPP.

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBAS

3.1 Introducción

En este capítulo se exponen todas las particularidades de la implementación del módulo, en aras de obtener un producto final que esté en correspondencia con los requisitos definidos por el cliente. Se mostrarán aspectos que definen de forma clara y sencilla el estilo que debe tener el código implementado en la aplicación desarrollada. Se validará aplicando diferentes tipos de prueba que propone la metodología de desarrollo de software utilizada.

3.2 Implementación

El Modelo de Implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes se pueden encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos (Hernández, Leovigilda,2013).

3.2.1 Diagrama de componentes

Según (Hernández, Leovigilda,2013) un componente es una parte física de un sistema (módulo, base de datos, programa ejecutable, etc). Se puede decir que un componente es la materialización de una o más clases, porque una abstracción con atributos y métodos pueden ser implementados en los componentes.

El diagrama de componentes muestra cómo están divididos los componentes dentro del sistema y las dependencias que existen entre ellos.

- Ayuda a los desarrolladores a visualizar el camino de la implementación.
- Provee una vista arquitectónica de alto nivel del sistema.
- Permite tomar decisiones respecto a las tareas de implementación.

A continuación, se muestra el diagrama de componentes correspondiente a la propuesta de solución (**Ver Fig.10**).

Diagrama de Componentes

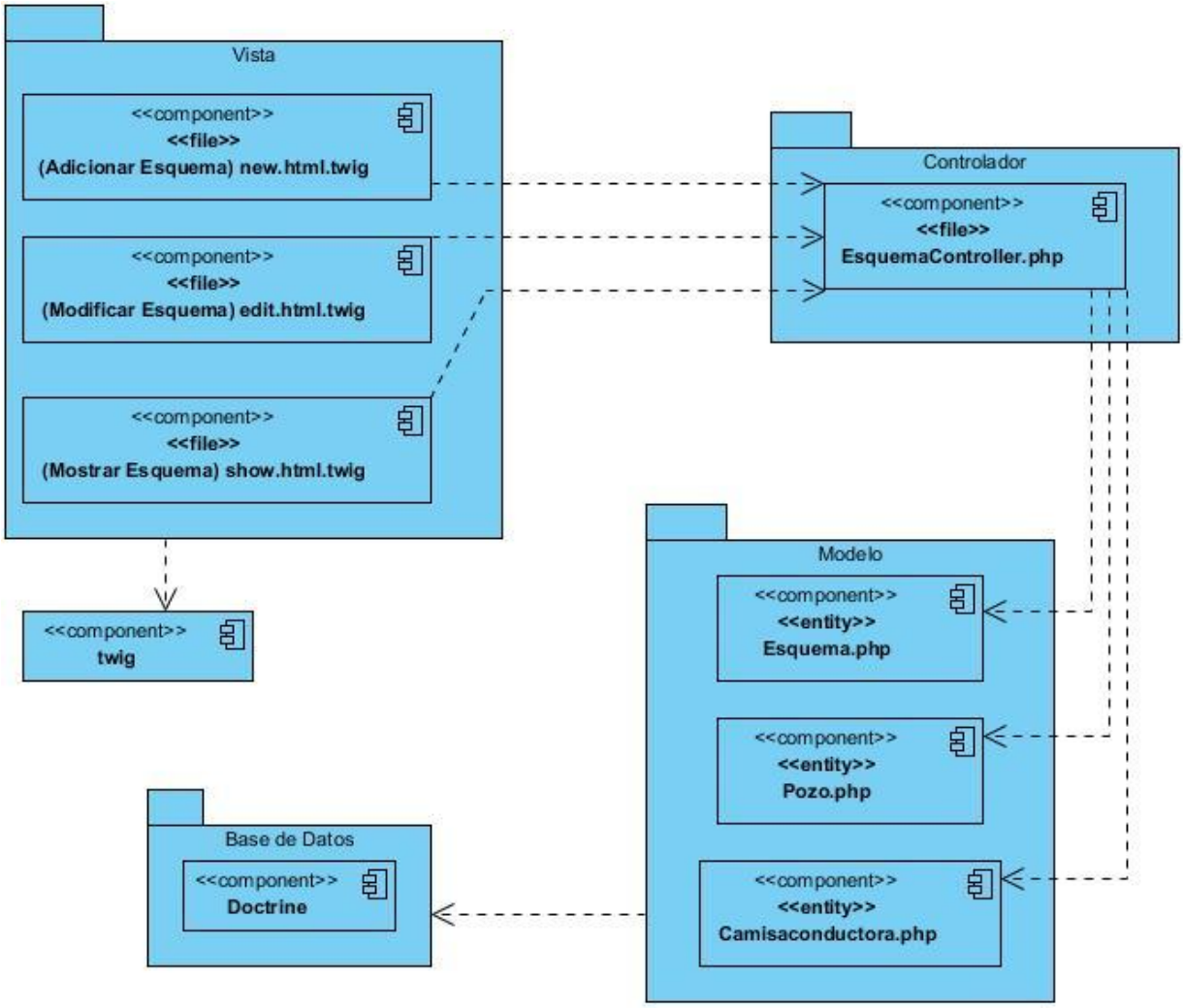


Fig.10: Diagrama de Componentes.

Twig: Es un motor de plantillas rápido y eficiente para PHP. Se recomienda mediante el uso de Symfony2 el empleo de Twig para crear todas las plantillas de la aplicación ya que uno de los objetivos de Twig es conseguir que los diseñadores sean capaces de crear todas las plantillas de la aplicación de forma autónoma, sin ayuda de los programadores. De esta forma se acelera el desarrollo de las aplicaciones y se mejora la productividad.

Doctrine: Se encarga del mapeo en la base de datos, accede a los datos de las tablas (entidades) para consultarlos. Realiza acciones sobre los datos tales como: insertar, borrar y actualizar.

3.2.2 Estándares de codificación

Un estándar de codificación permite tener un código entendible y organizado. Son muy empleados para asegurar la unificación en el código y tratar que todos sigan reglas durante el desarrollo. Son pautas que no están enfocadas a la lógica del programa sino a la apariencia y estructura del código (Fabre, 2013).

Seguidamente se definen las pautas o estándares de codificación utilizados en la implementación de la propuesta de solución:

Estructura:

- Sólo se añade un espacio después de cada delimitador coma.
- Sólo se añade un espacio alrededor de los operadores.
- Se añade una coma después de cada elemento del arreglo.
- Se añade una línea en blanco antes de las declaraciones return.
- Se declara las propiedades de las clases antes que los métodos.
- Se declara explícitamente la visibilidad de los métodos utilizando las palabras clave: private, protected o public.
- Se usan llaves para indicar la estructura del cuerpo de control, independientemente del número de declaraciones que contenga.

Convenciones de nomenclatura:

- El nombre de las clases siempre comienza con letra mayúscula, si es un nombre compuesto por más de una palabra, cada una debe comenzar con mayúscula y sin espacios. Ejemplo: EntidadController.
- El nombre de las variables siempre comienza con el carácter especial “$”, sin espacio y escrito en minúsculas. En caso de ser un nombre compuesto por más de una palabra, cada una debe escribirse en minúscula, sin espacio y sin guiones. Ejemplo: $id y $resumenconstruccionpozo.
- El nombre de los métodos o funciones comienza con letra minúscula, si es un nombre compuesto por más de una palabra cada una debe escribirse con mayúscula y sin guiones. Con la excepción del método ajax_request (). Ejemplo: editAction ().

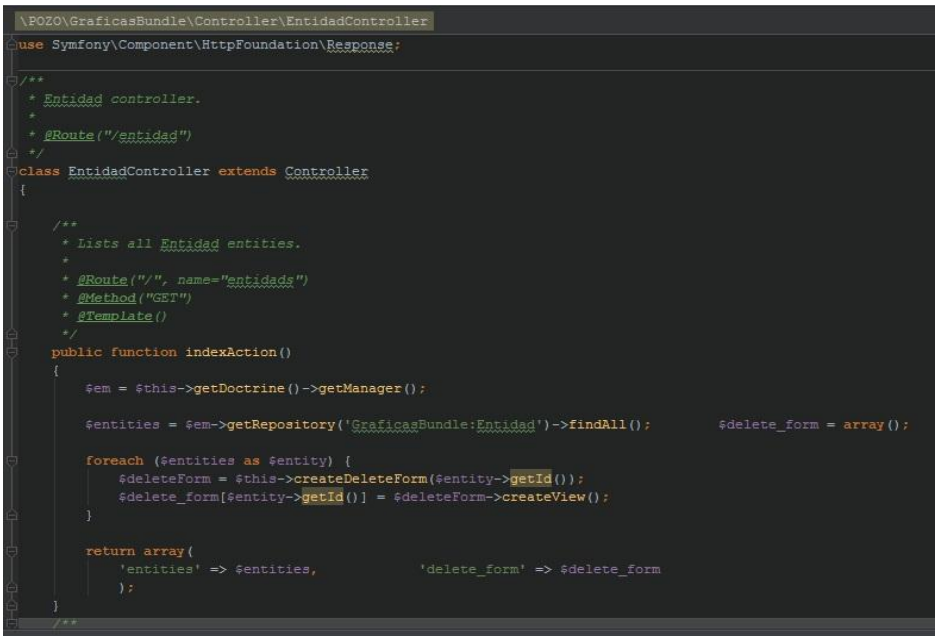
3.3 Patrones de diseños

Los patrones de diseño son soluciones para problemas típicos y recurrentes que no se pueden encontrar a la hora de desarrollar una aplicación. Estas soluciones ayudan a desarrollar aplicaciones robustas y fáciles de mantener, además ayudan a estandarizar el código, haciendo que el diseño sea más comprensible para otros programadores y están probadas y documentadas por una gran multitud de especialistas de la rama (Rubenfa, 2014).

En el desarrollo del sistema se utilizaron los siguientes patrones de diseño:

Patrones GRASP (Patrones Generales de Software para Asignar Responsabilidades):

- **Alta Cohesión:** Symfony aprueba asignar responsabilidades con una alta cohesión, por ejemplo, la clase EntidadController.php tiene como trabajo definir las acciones para las plantillas y colabora con otras para realizar distintas operaciones, instanciar objetos y acceder a las propiedades, es decir, está formada por diferentes funcionalidades que se hallan estrechamente relacionadas facilitando que el software sea flexible frente a grandes cambios (**Ver Fig.11**).



```
\POZO\GraficasBundle\Controller\EntidadController
use Symfony\Component\HttpFoundation\Response;

/**
 * Entidad controller.
 *
 * @Route("/entidad")
 */
class EntidadController extends Controller
{

    /**
     * Lists all Entidad entities.
     *
     * @Route("/", name="entidads")
     * @Method("GET")
     * @Template()
     */
    public function indexAction()
    {
        $em = $this->getDoctrine()->getManager();

        $entities = $em->getRepository('GraficasBundle:Entidad')->findAll();        $delete_form = array();

        foreach ($entities as $entity) {
            $deleteForm = $this->createDeleteForm($entity->getId());
            $delete_form[$entity->getId()] = $deleteForm->createView();
        }

        return array(
            'entities' => $entities,            'delete_form' => $delete_form
            );
    }
    /**
```

Fig.11: Código que muestra el patrón Alta Cohesión.

- **Bajo Acoplamiento:** Este patrón se manifiesta en cada uno de los módulos del sistema, por ejemplo, la clase EntidadController.php hereda solamente de Controller para lograr un bajo acoplamiento de clases (**Ver Fig.12**).

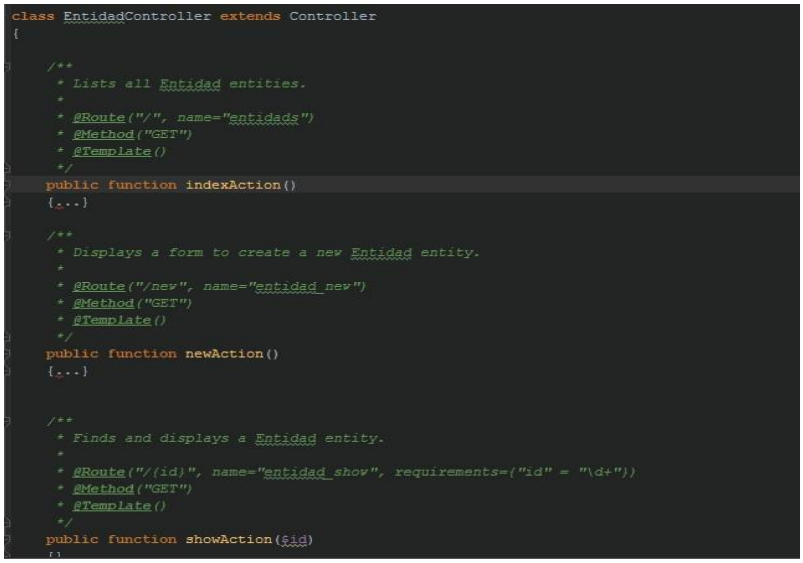


Fig.12: Código que muestra el patrón Bajo Acoplamiento.

- **Creador:** En la clase EntidadController.php del módulo se encuentra definida las acciones del sistema. En las acciones se crean los objetos de las clases que representan las entidades, evidenciando de este modo que la clase ControladorController.php es "creadora" de la entidad Gráfica (**Ver Fig.13**).

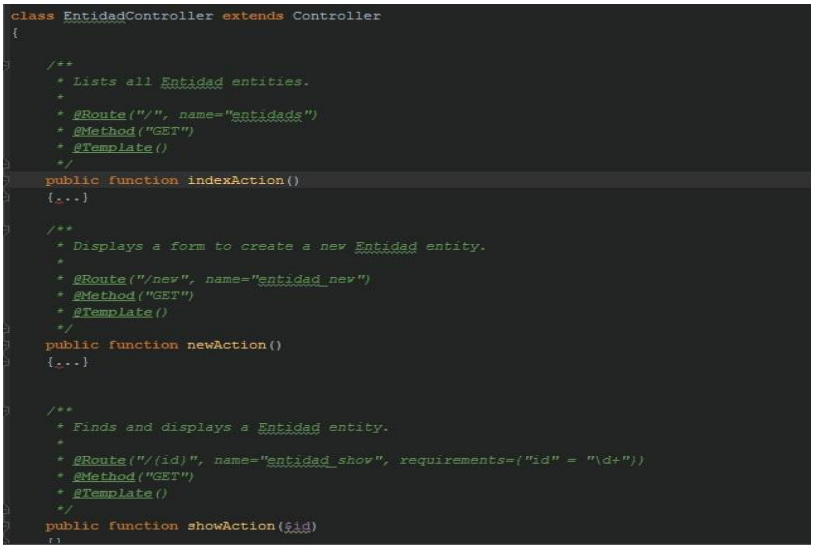


Fig.13: Código que muestra el patrón Creador.

- **Controlador:** Todas las peticiones son manejadas por un solo controlador frontal (app_dev.php), que es el punto de entrada único de toda la aplicación en un entorno determinado. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario. Este patrón se puede observar en la clase EntidadController.php ya que la misma se encarga de gestionar todo el proceso de diseño de esquema de pozo (**Ver Fig.14**).

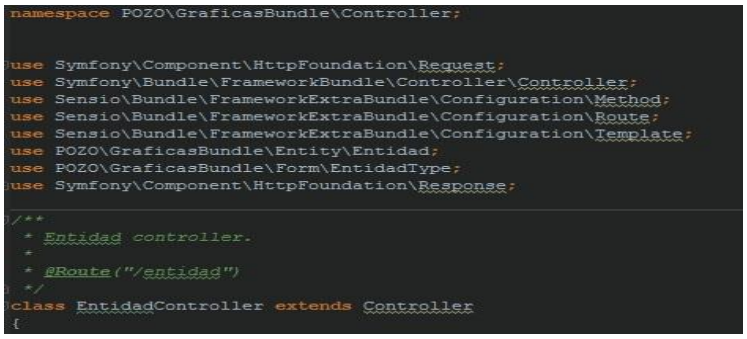


Fig.14: Código que muestra el patrón Controlador.

- **Experto:** Es un principio básico que suele utilizarse en el diseño orientado a objetos. Este propone la asignación de responsabilidad a la clase que cuenta con la información necesaria para crear una instancia o implementar un método. En la solución que se propone este patrón se evidencia ya que cada clase es responsable de manejar la información que posee (**Ver Fig.15**).

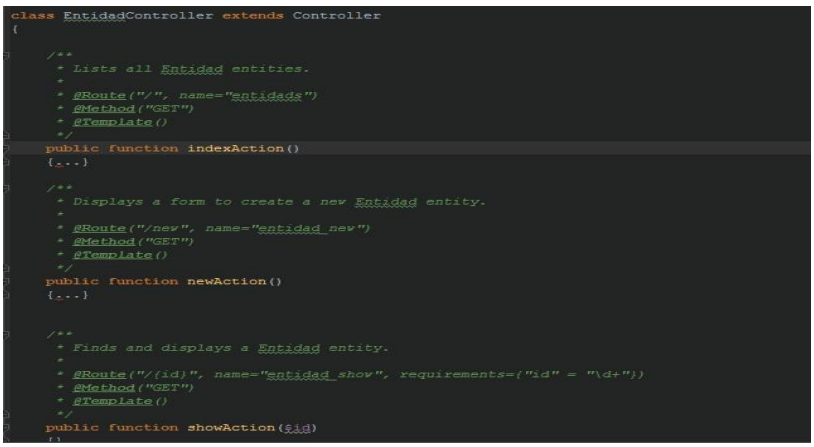


Fig.15: Código que muestra el patrón Experto.

Patrones GOF (Banda de los Cuatro):

- **Decorador** (Envoltorio): Añade funcionalidad a una clase, dinámicamente. El archivofrontend.html. twig, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en la plantilla global, si se mira desde otro punto de vista, la plantilla global decora la plantilla (**Ver Fig.16**).

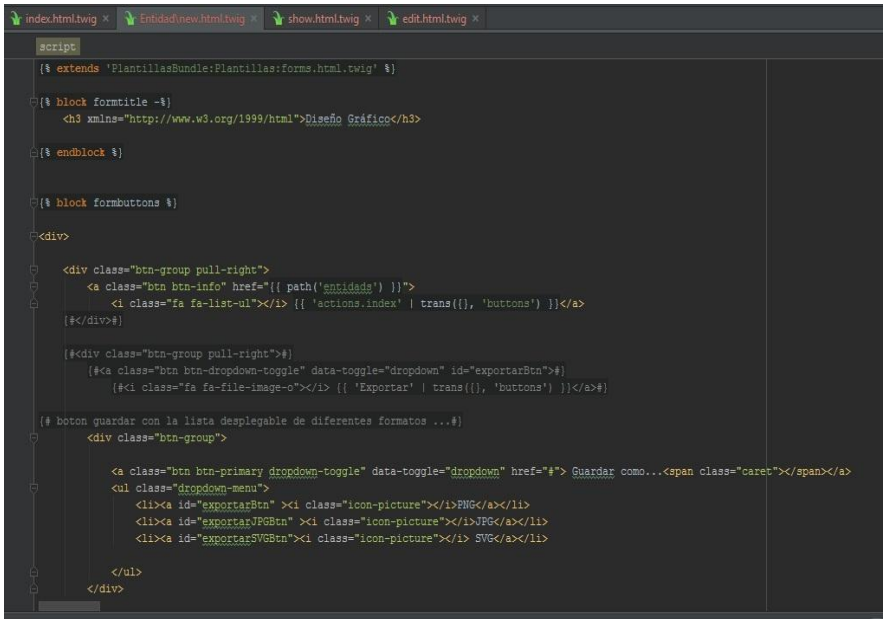


Fig.16: Código que muestra el patrón Decorador.

3.4 Pruebas de software

Una prueba es un conjunto de actividades que pueden planearse por adelantado y realizarse de manera sistemática. Por esta razón, durante el proceso de software, debe definirse una plantilla para la prueba del software: un conjunto de pasos que incluyen métodos de prueba y técnicas de diseño de casos de prueba específicos (Tencio, 2015).

Métodos de Prueba

Según Pressman “Las pruebas del software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación”. El software debe probarse desde dos perspectivas diferentes: la lógica interna del programa, se comprueba utilizando técnicas de diseño de casos de prueba

(caja blanca) y los requisitos del software, se comprueban utilizando técnicas de diseño de casos de prueba (caja negra).

3.4.1 Caja Blanca

Según, (Bárbara,2012) afirma que las pruebas de caja blanca, que sería mejor llamarlas de caja transparente, se establecen por medio del diseño de casos que usan como base las estructuras de control del flujo. Estas pruebas se pueden clasificar en tres tipos según sus objetivos:

- Que se ejecute por lo menos una vez cada instrucción del programa.
- Garantizar que todas las condiciones se comprueban como verdaderas y falsas.
- Que se ejecuten los bucles, probando el caso general y los casos extremos.

Al sistema desarrollado se le aplicó la prueba de caja blanca, haciendo uso de la técnica del camino básico, con el objetivo de evaluar la complejidad lógica de un diseño procedimental.

A continuación, se analizan y enumeran las sentencias de código de la funcionalidad *ShowAction* (**Ver Fig.17**).

```
public function showAction($id)
{
    $em = $this->getDoctrine()->getManager(); //1

    $entity = $em->getRepository('GraficasBundle:Entidad')->find($id); //2

    if (!$entity) { //3
        throw $this->createNotFoundException('Unable to find Entidad entity.'); //4
    }

    $deleteForm = $this->createDeleteForm($id); //5

    return array( //6
        'entity' => $entity,
        'delete_form' => $deleteForm->createView(),
    );
}
```

Fig.17: Función que muestra los detalles de la imagen

Ahora es necesario representar el grafo de flujo asociado al código presentado anteriormente a través de nodos, aristas y regiones (**Ver Fig.18**).

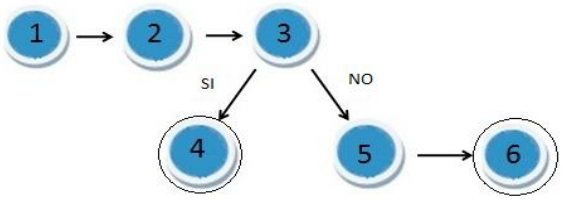


Fig.18: Grafo de flujo asociado al método showAction

Cálculo de la complejidad ciclomática para el grafo de flujo de la **(Ver Fig.18*)***.

$$V(g) = (a - n)(I)$$

$$V(g) = (p + 1)(II)$$

$$V(g) = r\ (III)$$

Siendo $V(g)$ el valor de la complejidad ciclomática, “a” la cantidad total de aristas, “n” la cantidad total de nodos, “p” la cantidad total de nodos predicados (nodos de los cuales parten 2 o más aristas) y “r” la cantidad total de regiones, se incluye el área exterior del grafo como una región más.

$$V(g) = (5 - 6) = -1 \qquad V(g) = (1 + 1) = 2 \qquad V(g) = 2$$

La evaluación de las formulas anteriores arroja un valor de complejidad ciclomática igual a 2, de manera que existen 2 posibles caminos por donde el flujo puede circular. Dicho valor representa el número mínimo de casos de prueba para el procedimiento tratado.

A continuación, se especifica los caminos básicos que puede tomar el algoritmo durante su ejecución.

Camino básico #1: 1-2-3-4

Camino básico #2: 1-2-3-5-6

Por tanto, se procede a ejecutar los casos de prueba para cada uno de los caminos básicos determinados en el grafo de flujo. Para definirlos es necesario tener en cuenta:

Descripción: En este se describen los casos de prueba y se especifican los aspectos fundamentales de los datos de entrada.

Condición de ejecución: Se verifican que los parámetros cumplan las condiciones de ejecución.

Entrada: Se muestran los parámetros de entrada del procedimiento.

Resultados esperados: Se especifica el resultado que debe arrojar el procedimiento.

TABLA XVII: CASO DE PRUEBA PARA EL CAMINO BÁSICO #1

Caso de prueba Camino básico #1	
Descripción	Los datos de entrada cumplirán los siguientes requisitos: Se realiza en primera instancia una consulta en DQL, (*Doctrine Query Language*), para luego mediante el id solicitar todo lo que contiene la entidad, si dicha entidad no existe se crea la siguiente excepción: imposible encontrar la entidad.
Condición de ejecución	La entidad no existe.
Entrada	Se tiene como dato de entrada el identificador del esquema **(id),** el cual se define como un número entero **(*integer*).**
Resultados esperados	El sistema muestra un mensaje enunciando que es imposible encontrar la entidad.

Resultado: El sistema lanza una excepción mostrando que es imposible encontrar la entidad.

TABLA XVIII: CASO DE PRUEBA PARA EL CAMINO BÁSICO #2

Caso de prueba Camino básico #2	
Descripción	Los datos de entrada cumplirán los siguientes requisitos: Se realiza en primera instancia una consulta en DQL, (*Doctrine Query Language*) para

	luego con doctrine mediante el id solicitar todo lo que contiene la entidad, si dicha entidad existe se crea un formulario *delete* (permite que una vez presionado el botón eliminar el sistema responda a la acción y retorne un arreglo con la entidad y el formulario *delete*).
Condición de ejecución	Existe la entidad.

Entrada	Se tiene como dato de entrada el identificador del esquema **(id),** el cual se define como un número entero **(*integer*).**
Resultados esperados	Se retorna un arreglo con la entidad y el formulario *delete*.

Resultado: Se retorna un arreglo con la entidad y el formulario *delete.*

3.4.2 Resultados

Se realizaron un total de 50 pruebas de caja blanca (pruebas unitarias), de estas 8 resultaron no satisfactorios, lo que representa un 13% del total de casos de prueba de caja blanca realizados **(Ver Fig.19).** Los errores detectados por los casos de pruebas no satisfactorios fueron eliminados luego de realizarse 3 iteraciones de prueba.

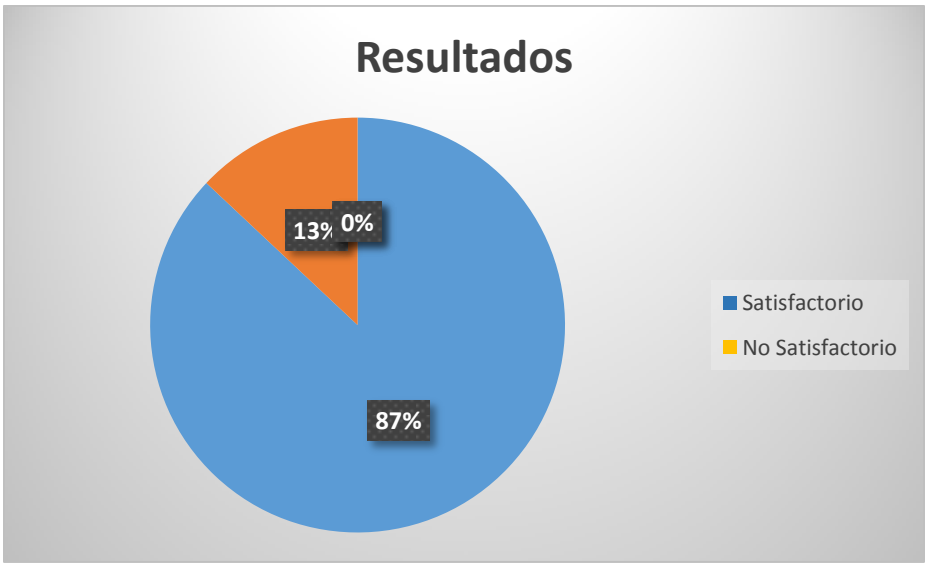


Fig.19: Resultados de las pruebas de caja blanca

3.5 Caja Negra

La técnica de caja negra es empleada en la ejecución de las pruebas de usabilidad, las cuales son realizadas una vez se compruebe que el sistema se encuentra funcionando como un todo. Para llevar a cabo la validación de la propuesta de solución el equipo de desarrollo determinó el empleo de las técnicas de partición de equivalencia. Estas facilitan comprobar que las funcionalidades registradas en las historias de usuarios se realicen satisfactoriamente.

Para su realización se generan un conjunto de casos de prueba en los que se registran las acciones a probar, los posibles valores introducidos en el sistema y los resultados que debe arrojar el mismo. De esta manera se podrá validar si la propuesta de solución satisface o no las necesidades planteadas por el cliente.

A continuación, se muestra el caso de prueba correspondiente a la historia de usuario Generar el inspector de propiedades-02.

TABLA XIX: CASO DE PRUEBA DE USABILIDAD

Caso de prueba de usabilidad			
Código: 1		**Historia de usuario: RF2**	Generar el inspector de

			propiedades-02.
Nombre: Generar el inspector de propiedades.			
Descripción: El sistema debe permitir insertar, modificar y eliminar las propiedades del componente seleccionado.			
Acción a probar		**Resultados esperados**	
Insertar un componente.		El componente escogido quedará representado dentro del área de trabajo CANVAS.	
Eliminar componente.		Se eliminará el componente especificado.	
Modificar componente.		Se muestra cada una de las propiedades con las cuales el usuario puede transformar el componente seleccionado, ya sea el tamaño, color del borde como el color de fondo de ésta.	
Evaluación de la prueba: Satisfactoria.			

3.5.1 Resultados

Se realizó un total de 30 casos de prueba de caja negra (pruebas de usabilidad), de estas 23 resultaron no satisfactorias, lo cual representa el 78% del total de pruebas de caja negras realizadas **(Ver Fig.20).** Mientras los 7 casos de prueba restantes resultaron satisfactorios para un 22% del total. Los errores detectados por los casos de pruebas no satisfactorias fueron eliminados luego de realizarse 3 iteraciones de prueba.

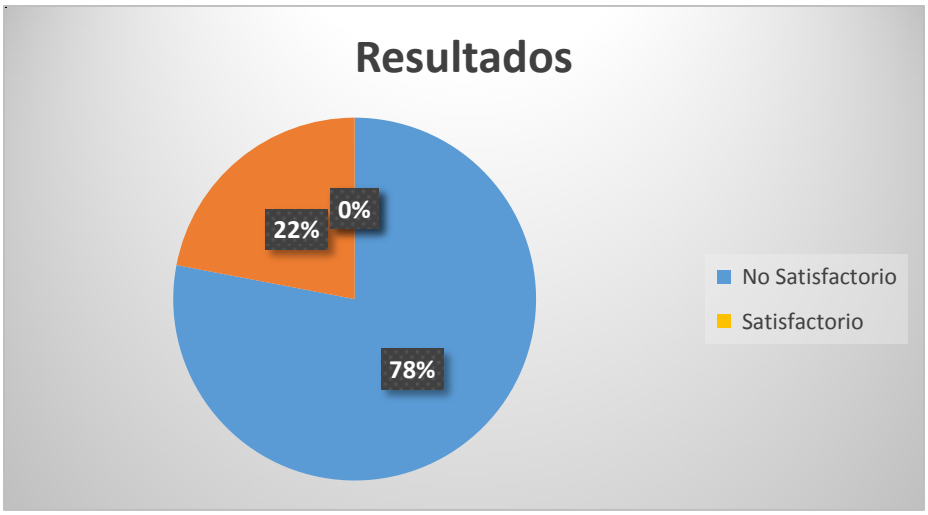


Fig.20: Resultados de las pruebas de caja negra.

3.6 Integración

El módulo de diseño gráfico como resultado de la presente investigación, culmina con su integración en el sistema SIPP para el cual ha sido propuesto. Dicha integración se realizó de forma satisfactoria sin afectar el funcionamiento de otros módulos del sistema y lográndose obtener los reportes de: Construcción del pozo, Encamisado y Terminación y ensayo con sus respectivos esquemas realizados desde el módulo de diseño gráfico. Como aval de este resultado se cuenta con una Carta de aceptación (Ver Anexo 2) realizada por el líder del proyecto SIPP como cliente inmediato del módulo desarrollado. En la Fig.21 se muestra la vista principal del módulo gráfico integrado en el sistema SIPP.

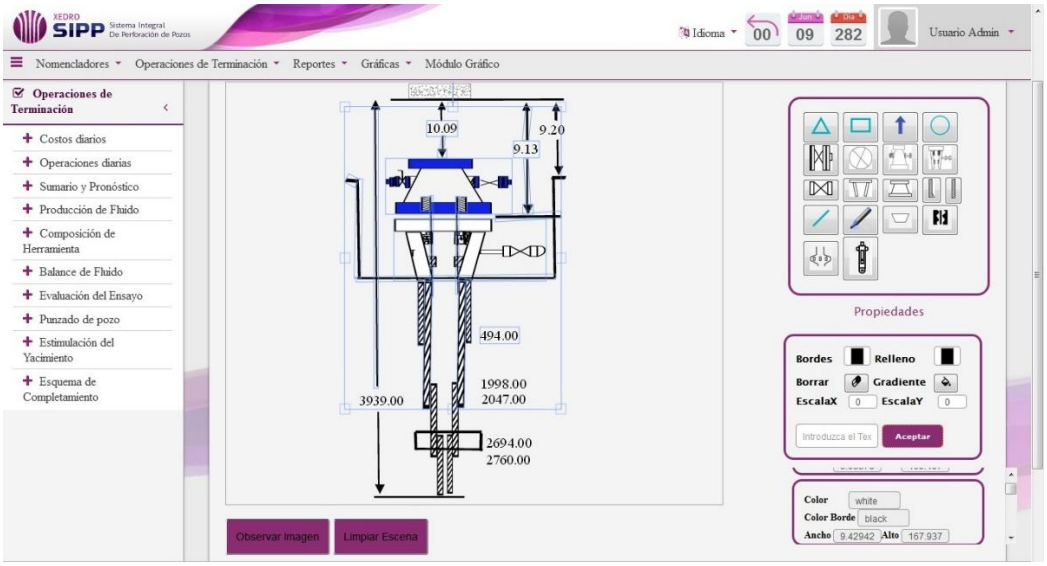


Fig.21 Integración del módulo gráfico en el sistema SIPP.

3.7 Conclusiones Parciales

Después de haber realizado la implementación y haber llevado a cabo las pruebas al módulo de diseño gráfico, se concluye que:

- Los patrones de diseño facilitaron la asignación de responsabilidades mostrando que el desarrollo del software sirve de apoyo en la implementación de nuevas herramientas.
- Las pruebas de caja blanca y caja negra realizadas al módulo arrojaron resultados satisfactorios luego de tres iteraciones. Además, se realizó la integración del módulo con el sistema SIPP de forma satisfactoria de modo que el líder del proyecto SIPP emitió una carta de aceptación hacia el módulo de diseño gráfico (Ver Anexo 2).

CONCLUSIONES GENERALES

El análisis de los conceptos asociados al dominio del problema contribuyó al conocimiento de los principales términos empleados en la investigación.

La descripción, el diseño y la modelación, permitieron crear las bases para la implementación del módulo gráfico, garantizando la creación de un sistema completamente funcional.

Se obtuvo un módulo de diseño gráfico para SIPP que permite la creación de esquemas de construcción de pozos mediante la incorporación de componentes gráficos personalizados en cada uno de los reportes.

RECOMENDACIONES

Se recomienda extender las funcionalidades de la propuesta de solución para incorporar automáticamente nuevos componentes gráficos a la paleta de componentes. Así como extender las propiedades que maneja el inspector de propiedades.

Referencias bibliográficas

"MPIu+a, G. A. (2004). *Una metodologia que integra la ingeniería del software, la interacción persona-ordenador y la accesibilidad en el contexto de equipos de desarrollo multidisciplinares" [Tesis Doctoral].* Lleida: Universitat de Lleida.

@whatisdotcom. (10 de 02 de 2017). *SearchDataCenter en Español.* Recuperado el 11 de 03 de 2017, de Modelado de datos: http://searchdatacenter.techtarget.com/es/definicion/Modelado-de-datos

Alegsa, L. (2010). *Diccionario de Informática y Tecnología.* Argentina.

Alicia Bárbara Expósito Santana. (23/04/2012). Pruebas de caja blanca. https://abesdih.files.wordpress.com/2012/04/cajablancaabes_1.pdf

Baldasano, J. (1989). *Influencia de la Informática en el Proceso de Proyectar.* Valencia.

Barrera, Y. M. (2013). *Módulo para monitorizar la integridad de los archivos.* La Habana.

Contreras, F. y. (2001). *Diseño gráfico, creatividad y comunicación.* Madrid: Blur ediciones.

Donatien, A. R. (2011). *Descripción de la Metodología de Desarrollo de Software.* La Habana.

Donatien, A. R. (2009). *Sistema para la Identificación de Aguas en Pozos Petroleros (SIAPP).* La Habana, Universidad de las Ciencias Informática.

Eguiluz, J. (2010). *Symfony.* version 4.8. Obtenido de http://symfony.com/what-is-symfony

Eisenberg, D. (2002). *SVG Essentials.* O'Reilly Media.

Española, R. A. (2014). *Diccionario de la lengua española.*

Fabriq.js. (s.f.). Obtenido de https://www.google.com.cu/?gws_rd=cr,ssl&ei=Wv2AWJ3PGMPymAGThJPACQ#q=fabriqjs.com

Francois Zaninotto, F. P. (2016). *Symfony, la guia definitiva*.

Gómez, M. (16 de 09 de 2013). *Patrones Arquitectónicos.* Recuperado el 10 de 03 de 2017, de Ingenio DS: https://ingeniods.wordpress.com/2013/09/16/patrones-arquitectonicos/

Hernández, Leovigilda. Modelo de Implementación. [En línea] 01 de 06 de 2013. [Citado el: 15 de 06 de 2017]. http://ithleovi.blogspot.com/2013/06/unidad-5-modelo-deimplementacion-el.html.

IBM Knowledge Center. (2016). Obtenido de https://www.ibm.com/support/knowledgecenter/es/SSVRGU_8.5.3/com.ibm.designer.domino.main.doc/H_CA_USING_COMPONENT_PICKER_DD.html

Ingenieria de Software. (s.f.). Obtenido de http://ingenieriadesoftware.mex.tl/63758_AUP.html

ISO. (1998). Obtenido de Ergonomic requirements for office work with visual display terminals (VDTs)-Part 11: Guidance on usability.

Joyce Suárez Fabre. (2013). Módulo de Seguridad para SIPP v 2.0. PDF,Documento de Tesis del instituto de la Universidad de las Ciencias Informaticas, La Habana.

JQuery. (s.f.). Obtenido de http://api.jquery.com/

Korry Douglas, S. D. (2005). *PostgreSQL* (2da edicion ed.). Estados Unidos.

Leiva Pereira, Yusemí;. (2012). *Diseño e implementación del módulo de graficar del Sistema de Manejo Integral de Perforación de Pozo.* Tesis, UCI, La Habana. Recuperado el 02 de 03 de 2016

McGuigan, C. (1988). *Psicología experimental un enfoque metodológico.*

McFarland, D. S. (2014). *JavaScript & jQuery: The Missing Manual* (3ra edicion ed.). Estados Unidos: ISBN-13: 987-1-491-94707-4.

Mendoza, A. T. (2005). *El diseño gráfico en el espacio social.* Mexico.

Pfaff, G. E. (1983). *User Interface Management.* G. Enderle and D. Duce.

Rodriguez, J. (31 de juio de 2012). *Unidad 7: Profundizando en la arquitectura MVC de Symfony — index 1.00 documentation:.* Recuperado el 13 de marzo de 2017, de Unidad 7: Profundizando en la arquitectura MVC de Symfony — index 1.00 documentation:: http://juandarodriguez.es/cursosf14/unidad7.html

Rubén Alvarez, M. A. (2013). *desarrolloweb*. Obtenido de Manual de PHP: www.desarrolloweb.com

Rubén Alvarez, M. A. (s.f.). *desarrolloweb*. Obtenido de Manual de PHP: www.desarrolloweb.com

Rubenfa. (14 de 07 de 2014). *Genbeta Dev.* Recuperado el 10 de 03 de 2017, de Patrones de diseño: qué son y por qué debes usarlos: https://www.genbetadev.com/metodologias-de-programacion/patrones-de-diseno-que-son-y-por-que-debes-usarlos

Samara, T. (2015). *Los elementos del diseño.* España: Barcelona: Editorial Gustavo Gili, S.L.

Suaza, K. V. (2014). *Definición de equivalencias entre historias de usuario y especificaciones en para el desarrollo ágil de software.* PDF, Facultad de Minas – Departamento de Ciencias de la Comp, Medellín, Colombia.

Tamayo, M. F. (2006). *El mapa conceptual una herramienta para aprender y enseñar* (Vol. 5).

Tencio, Diego**.** Prezi. [En línea] 23 de 11 de 2015. [Citado el: 03 de 06 de 2016.] https://prezi.com/9r2q23n4kzpw/administracion-de-calidad/.

Ubiquitour. (17 de julio de 2014). Obtenido de ubiquitour.com

ANEXOS

Anexo 1: Diagrama de Clases Persistentes

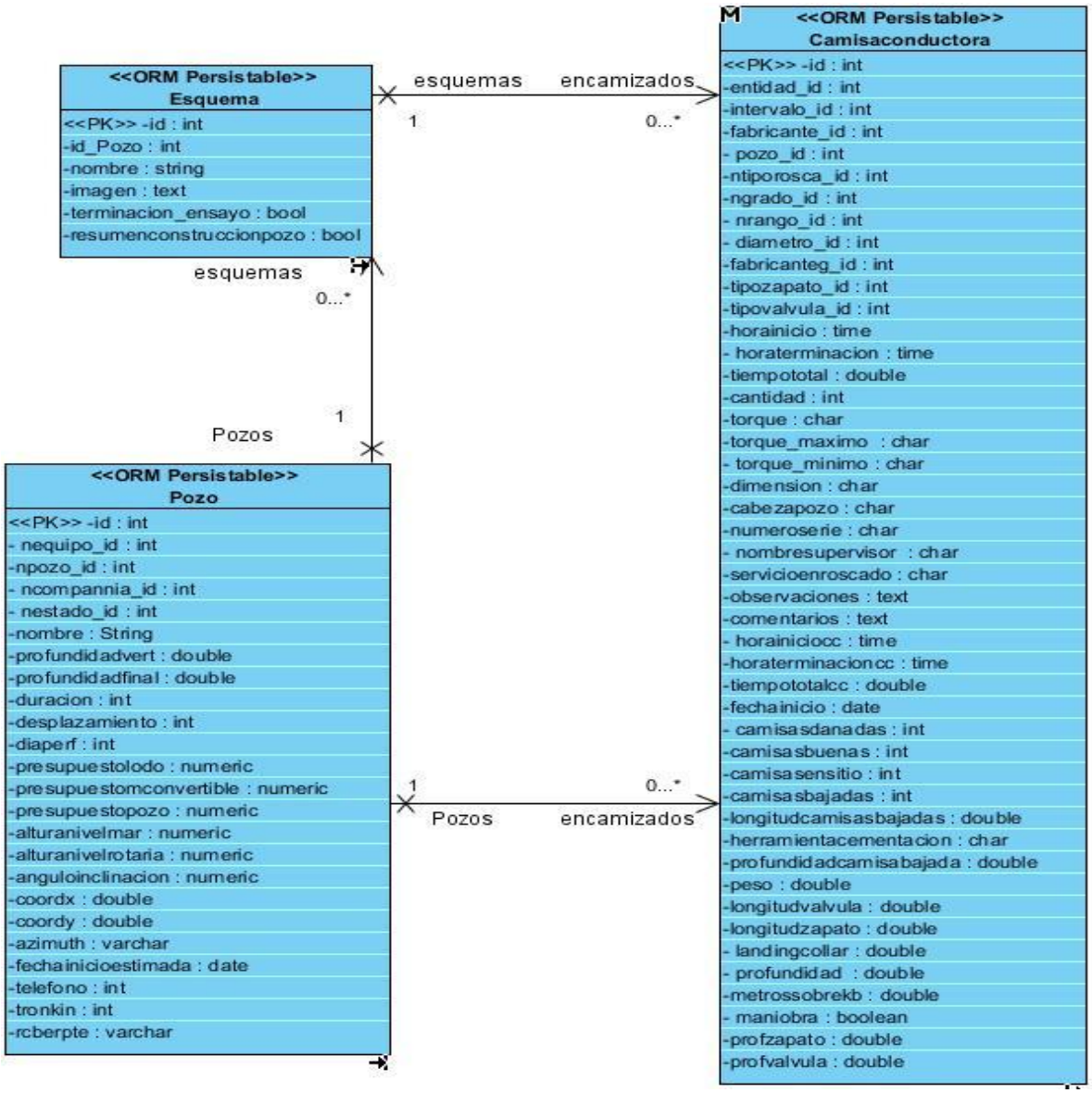


Fig.22: Diagrama de clases Persistente.