Universidad de las Ciencias Informáticas.

Facultad 5.

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

Título: Editor de Estilos QSS para el Módulo Interfaz Hombre Máquina del SCADA "Guardián del Alba".

Autor:

Danay Méndez Sánchez.

Tutor:

Ing.Raudi Agdel Bacallao Sánchez.

Co-Tutora:

Ing.Yadira Ramírez Rodríguez.

Ciudad de La Habana, del 2013.

" Si lo puedo pensar e imaginar, es que lo puedo hacer."

Albert Einstein

Declaración de autoría

En el presente proyecto de tesis declaro que cada una de sus observaciones, análisis, evaluaciones, conclusiones y recomendaciones emitidas, es de absoluta responsabilidad de la autora Danay Méndez Sánchez y autorizo a la Universidad de las Ciencias Informáticas hacer uso del mismo en su beneficio.

Además, es necesario indicar que la información de otros autores empleada en el presente trabajo está debidamente especificada en fuentes de referencia y apartados bibliográficos.

Para que así conste firmo la presente a los ____ días del mes de _____ año _____.

Autor:

Danay Méndez Sánchez _____

Tutor:

Ing. Raudi Agdel Bacallao Sánchez_____

Co-Tutora:

Ing. Yadira Ramírez Rodríguez_____

Datos de contactos

Ing. Raudi Agdel Bacallao Sánchez

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

Categoría Docente: Instructor.

E-mail: rabacallao@uci.cu.

Graduado en la UCI en el año 2008, profesor con 4 años de experiencia docente en la Universidad de las Ciencias Informáticas (UCI) y en el desarrollo de software.

Ing. Yadira Ramírez Rodríguez

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

Categoría Docente: Asistente.

E-mail: yramirezr@uci.cu.

Graduada de Ingeniera en Ciencias Informáticas en el 2007. Profesora Asistente con 5 años de experiencia como docente en la facultad 5, Jefa del Colectivo de Ingeniería de Software y Asesora de Relaciones Internacionales de dicha facultad.

Agradecimientos

Dedicatoria

Resumen

El presente trabajo de diploma se centra en el desarrollo de un Editor de Estilos QSS para el Módulo Interfaz Hombre Máquina del SCADA (Sistemas de control, supervisión y adquisición de datos), del Centro de Informática Industrial (CEDIN). Este editor brindará la posibilidad de gestionar estilos en cascada que permitirán personalizar los componentes y objetos visuales del framework que fue desarrollado en el módulo de visualización del SCADA “Guardián del Alba” en correspondencia a las necesidades del usuario, permitiendo la reutilización de las interfaces de usuario en otros SCADA basados en las mismas consolas de operación creadas en el módulo.

En el documento se plasman los resultados del trabajo investigativo realizado. Se hace un análisis sobre las herramientas y tecnologías existentes seleccionando las más apropiadas y se examinan diferentes editores de estilo CSS pues de ellos se extraerán las principales funcionalidades que estarán presentes en el editor. Se muestran los resultados del diseño de la propuesta del sistema y se le aplican pruebas funcionales para comprobar el correcto funcionamiento del mismo. Finalmente se dejan algunas recomendaciones para el mejoramiento futuro de la aplicación.

Palabras clave: Editor; CSS; Framework; HMI; Interfaz; SCADA.

Índice

Índice de tablas

Índice de figuras

Introducción

La necesidad de aumentar los niveles de eficiencia, la minimización de los costos y la optimización de los procesos productivos, resulta de vital importancia para la aumentar la productividad de las industrias así como la calidad de los procesos. Actualmente, uno de los mecanismos más importantes para dar solución a estas necesidades, es la automatización de los distintos procesos que componen la cadena de producción, por tal motivo la necesidad de automatizar dichos procesos resulta prioritaria para cualquier industria que desee mantenerse a la vanguardia.

El país se encuentra enfrascado en una lucha por el aumento de la eficiencia y eficacia de las industrias con el objetivo de aumentar la productividad de las mismas y con ello disminuir las importaciones. En la Universidad de las Ciencias Informáticas (UCI) se encuentra el Centro de Informática Industrial (CEDIN) el cual se especializa en soluciones de automatización de procesos industriales, desde el año 2006 se comenzó a desarrollar un sistema de supervisión y control denominado SCADA Guardián del ALBA (GALBA) para la empresa estatal venezolana Petróleos de Venezuela SA(PDVSA) el cual se desarrolla bajo el paradigma del software libre y representa una alterativa de soberanía tecnológica para dicha empresa y Cuba.

El SCADA GALBA está compuesto por varios módulos, los más significativos por su impacto tanto en el núcleo del SCADA como con la interacción directa con el usuario final son Visualización, Adquisición, Histórico y Seguridad, todos intercomunicados a través del módulo de Comunicación. El módulo de Visualización o HMI[1] es una interfaz de usuario, que permite la comunicación entre los usuarios y el sistema. Los módulos HMI generalmente están compuestos por dos aplicaciones principales, ambiente de configuración y ambiente de ejecución. El ambiente de configuración se utiliza para configurar todos los módulos del SCADA así como las consolas de operación, haciendo uso de sinópticos gráficos, imágenes y texto para su diseño. El ambiente de ejecución es el encargado de visualizar, a través de la pantalla, los datos recolectados de los dispositivos de campo, para su supervisión y control.

El SCADA Guardián del ALBA (SCADA GALBA),se ha especializado en la rama del petróleo a lo largo del tiempo ,en la actualidad el SCADA GALBA se desea extender hacia

[1] Human machine interface(Interfaz hombre maquina)

otras ramas como son la industria eléctrica ,la hidráulica ,entre otras, las cuales poseen especificaciones de requerimientos que varían de acuerdo a su complejidad , cantidad de variables a monitorizar y gráficos representativos en pantallas , con esto cada componente de software que integra el SCADA debe adaptarse a los nuevos cambios , siendo el HMI uno de los más afectados pues, las interfaces gráficas variarían de acuerdo al cliente o empresa a la que pertenece la industria siendo este proceso bastante complejo, los componentes gráficos con los que se configuran las pantallas del HMI presentan algunas restricciones en cuanto a las propiedades que puedes ser modificadas, por el momento no existe un mecanismo que permita editar muchas de sus propiedades en tiempo de diseño por lo cual en ocasiones para lograr comportamientos personalizados, hay que recompilar las clases que representan dichos componentes gráficos, haciendo que el costo del tiempo estimado del desarrollo sea muy alto. Es posible reutilizar el HMI en casi todas sus funcionalidades, pero no hay formas de identificar visualmente.

Teniendo en cuenta lo antes expuesto se define como **problema científico** la siguiente interrogante:

¿Cómo adaptar las interfaces de usuario de un módulo HMI de manera rápida sin la necesidad de recompilar el código fuente?

Según el problema científico se plantea como **objeto de estudio**: Los editores de estilos para componentes gráficos.

Definiendo como **objetivo general** de la investigación: Desarrollar un software de aplicación que permita la gestión de hojas de estilos en cascada aplicables a los HMI.

Teniendo en cuenta la relación entre el problema, el objeto de estudio y el objetivo general de la investigación se obtiene como **campo de acción**: Herramientas de edición de hojas de estilos en cascada para módulos HMI.

Como **idea a defender** se define: Con la creación de un editor de estilos QSS [2]se hará posible la reutilización de las interfaces de usuario del módulo HMI del SCADA “Guardián del Alba” así como la modificación de los componentes gráficos en tiempo de ejecución sin la necesidad de la interacción de los desarrolladores y operadores que contará con un

[2] (Hojas de estilo en QT).

mecanismo de gestión de estilos que permitirá adaptar los componentes y objetos visuales en correspondencia a las necesidades del usuario.

Para darle cumplimiento al objetivo propuesto, aquí se proponen las siguientes **tareas:**

- Elaboración del marco teórico para sustentar teóricamente la propuesta de solución.
- Estudio de las herramientas de edición de estilos en casada en el marco nacional e internacional para demostrar la necesidad de la propuesta de solución.
- Identificación de las funcionalidades que debe cumplir el sistema para satisfacer las necesidades del cliente.
- Diseño de la propuesta de solución para guiar la implementación del sistema.
- Implementación de la propuesta de solución para darle cumplimiento al objetivo de la presente investigación.
- Realización pruebas al sistema para verificar las funcionalidades que desencadenaron la investigación.

  Esperando obtener como resultado:

  - Un editor de estilo que pueda ser utilizado para la edición de los estilos CSS en el módulo HMI del SCADA Guardián del Alba.

Para el cumplimiento de estos objetivos se utilizan varios **métodos y técnicas en la búsqueda y procesamiento de la información** como son:

A nivel teórico:

- **Método analítico-sintético:** Para el estudio de los contenidos relacionados con los editores de estilos , los sistemas de supervisión y control y sus componentes , se analizaron documentos elaborados por desarrolladores, para la extracción de los elementos relevantes.

- **Análisis histórico–lógico:** Para la comprensión de los antecedentes y las tendencias actuales referidas a la evolución en el mundo de los editores de estilo.

El presente documento está estructurado en tres capítulos a continuación se hace un resumen sobre lo que abordará cada capítulo:

- **En el capítulo 1** :Fundamentación teórica, se realiza un esbozo teórico del presente trabajo, centrado en las principales características y conceptos asociados a los editores de estilo en cascada vinculados al módulo HMI del SCADA. Además se realiza un estudio de los editores de estilos existentes a nivel internacional. Se definen las principales metodologías y herramientas usadas en la construcción de la solución.

- **En el capítulo 2:** Construcción de la solución, se propone la solución al sistema, se definen las historias de usuario, se diseña la arquitectura del sistema, así como los patrones del diseño reflejados en el sistema.

- **En capítulo 3**: Validación de la solución propuesta, se realizan pruebas a las diferentes funcionalidades del sistema, para comprobar el buen funcionamiento del mismo.

1 Fundamentación teórica

Introducción

En el presente capítulo se introducen las principales características y conceptos asociados a los editores de estilo en cascada vinculados al módulo HMI del SCADA. Para ello se abordan definiciones generales de un SCADA y su módulo. Se realiza una tabla comparativa entre diferentes editores de estilos en cascada para una mejor comprensión de las características y funcionalidades presentes en este tipo de aplicación. Además se describen las principales herramientas y metodologías que serán empleadas en la construcción de esta solución.

1.1 Definición de SCADA

Un SCADA se define como una aplicación de software para ordenadores que permite controlar y/o supervisar procesos a distancia. Proporcionando comunicación con los dispositivos de campo y controlando el proceso de forma automática desde la pantalla del operador. Además, provee a diversos usuarios, toda la información que se genera en el proceso productivo, control de calidad, supervisión, mantenimiento. (1)

1.2 Generalidades de los sistemas SCADA

Los SCADA son sistemas diseñados para ser implementados o usados en numerosas industrias, contando con módulos de software genéricos para proporcionar las capacidades requeridas por las distintas entidades. La mayoría de los SCADA que se instalan en la actualidad, se han convertido en una parte integral en la gestión de la información corporativa. Estos sistemas no son solamente herramientas operacionales, sino un recurso importante de información y esto se pone de manifiesto en los recursos que se exportan a otros sectores como la plataforma web y la tecnología móvil, permitiendo a usuarios externos ver lo que sucede en el proceso de producción controlado por el SCADA, así como obtener distintos reportes relacionados con datos importantes. Además de lo antes mencionado los actuales SCADA alcanzan un nivel aceptable de tolerancia a fallos y cuentan con ordenadores redundantes operando en paralelo con el mismo SCADA permitiendo la transferencia automática de la responsabilidad del control de cualquier ordenador que pueda tener cualquier tipo de colisión o falla por cualquier razón, a una computadora de reserva en línea permitiendo que no se detenga el servicio bajo ningún concepto.

Los SCADA deben cumplir ciertos y determinados requerimientos para su fácil instalación, configuración, mantenimiento y puesta en funcionamiento. Se puede decir que deben ser sistemas de arquitectura abierta, capaces de crecer o adaptarse según las necesidades cambiantes de la empresa. También deben comunicarse con toda facilidad y de forma transparente al usuario con el equipo de planta y con el resto de la empresa (redes locales y de gestión). Además deben ser programas sencillos de instalar, sin excesivas exigencias de hardware, y fáciles de utilizar, permitiendo interfaces de usuario amigables. Es un aspecto principal en los SCADAs, el tratamiento y manejo de los datos adquiridos, como se muestra en la figura 1. (2)

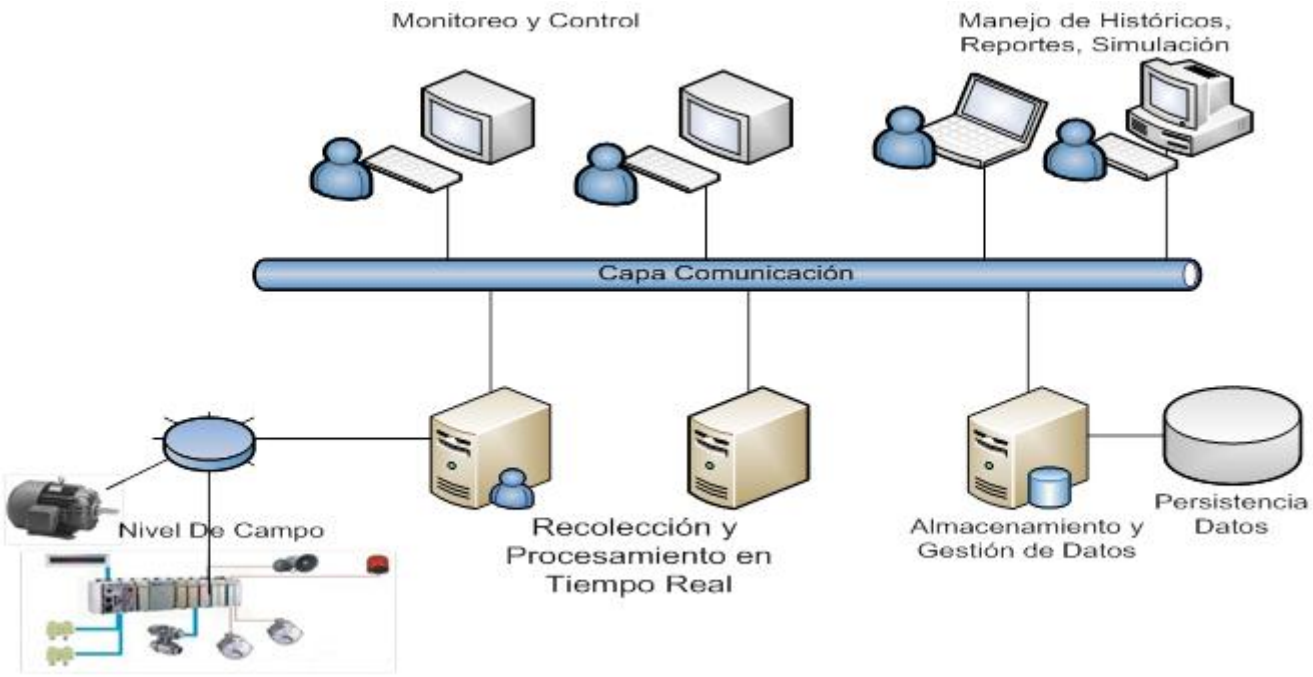

Figura 1. Ejemplo del esquema de un SCADA.

1.2.1 SCADA funcionalidades principales

Adquisición de datos: Significa que el sistema tiene la capacidad de obtener información desde el sistema de control, por ejemplo, sobre valores de temperatura o presión y a diferencia de la "supervisión" los datos son registrados o almacenados para su posterior explotación.

Supervisión: Significa poder observar o monitorear aquello que sucede en el proceso industrial, el equipo o la maquinaria, por ejemplo, conocer (generalmente de una forma gráfica) si un motor se encuentra encendido o apagado.

Control: Significa tener la posibilidad de ejecutar comandos; en otras palabras poder enviar instrucciones hacia el sistema de control.

El usuario, mediante herramientas de visualización y control, tiene acceso al procesador digital, generalmente un ordenador donde reside la aplicación de supervisión y control. La comunicación entre estos dos sistemas se suele realizar a través de redes de comunicación corporativas. El procesador digital capta el estado del sistema a través de los elementos sensores e informa al usuario a través de la herramienta HMI. Basándose en los comandos ejecutados por el usuario, el procesador digital inicia las acciones pertinentes para mantener el control del sistema, a través de los elementos actuadores. (3)

Prestaciones de un SCADA.

El paquete SCADA, comprende toda una serie de funciones y utilidades encaminadas a establecer una comunicación lo más clara posible entre los procesos y el operador, entre las prestaciones de una herramienta de este tipo destacan:

Monitorización: Representación de datos a los operadores de la planta. Se leen los datos de los autómatas (temperatura, velocidad, detectores).Pueden ser vigilados desde muchos kilómetros de distancia una máquina simple, una hidroeléctrica, un parque eólico.

Visualización de los estados de las señales del sistema (alarmas y eventos): Reconocimiento de eventos excepcionales acaecidos en la planta y su inmediata puesta en conocimiento a los operarios, para efectuar las operaciones correctas pertinentes. Además los paneles de alarmas pueden exigir alguna acción de reconocimiento por parte de los operarios, de forma que quede registrada la incidencia.

Mando: Posibilidad de que los operadores puedan cambiar consignas u otros datos claves del proceso, directamente desde el ordenador (marcha, paro, modificación de parámetros). Se escriben datos sobre elementos de control.

Seguridad de los datos: Restringiendo zonas de programa comprometidas a usuarios no autorizados, registrando todos los accesos y acciones llevadas a cabo por cualquier operador.

Programación numérica: Permite realizar cálculos aritméticos de elevada resolución sobre la CPU de ordenador.

1.3 Principales módulos del SCADA GALBA.

Los módulos o bloques software que permiten las actividades de adquisición, supervisión y control son los siguientes:

- **Base de Datos Históricos**

Implementa los mecanismos para almacenar la información recibida desde los dispositivos de campo, así como las alarmas y eventos generados. La información almacenada es utilizada por las diferentes aplicaciones del sistema.

- **Comunicaciones**

Constituye la capa de software encargada de la comunicación entre los distintos procesos distribuidos, de mediano y alto nivel.

- **Seguridad**

Brinda las interfaces necesarias para que los usuarios se autentiquen en el sistema, y accedan únicamente a los recursos que tengan asignados según su rol. Incluye herramientas para la protección ante ataques, fallos eléctricos, problemas de red, entre otros.

- **Reportes**

Brinda al usuario, la información referente al comportamiento de las variables a través de reportes especializados, los cuales se diseñan, teniendo en cuenta la necesidad de los usuarios, gracias a un editor que posee el mismo.

- **Ambiente de Configuración**

Permite configurar varios procesos o parte de ellos, aquí se definen y gestionan las variables, la configuración de los manejadores, los comandos, las alarmas y variadas opciones adicionales.

- **Visualización(HMI)**

Permite representar gráficamente, de forma segura los procesos operacionales con los que los usuarios interactúan. A través del mismo se pueden visualizar condiciones operacionales, valores de variables, visualizar alarmas, navegar entre despliegues, enviar comandos, entre otros.

- **Núcleo de procesamiento de datos (Recolector)**

El núcleo de procesamiento de datos es el encargado del tratamiento y análisis en tiempo real de la información adquirida desde el campo.

- **Manejadores**

El módulo de manejadores de dispositivos asegura la comunicación del SCADA con los dispositivos de campo. Esta comunicación se realiza a través de una interfaz genérica única para todos los protocolos y bibliotecas dinámicas (Manejadores o drivers) que implementan esta interfaz en cada caso específico.

- **Configuración**

Este módulo se encarga de almacenar, persistir y suministrar la información base para el funcionamiento de los demás módulos del SCADA, (BDTR [3], BDH[4], HMI entre otros), cada módulo del sistema posee un agente (biblioteca), que permite establecer las comunicaciones con el servidor de configuración, a través de la capa de conectividad, permitiendo crear, eliminar y modificar los recursos configurables del SCADA. (5)

1.4 El módulo Interfaz Hombre-Máquina

Esta interfaz, muestra los datos de uno o varios procesos a un operador (humano) y además permite al operario, ejercer el control sobre los mismos. Los HMI surgen esencialmente de la necesidad de estandarizar la manera de monitorear y de controlar múltiples sistemas remotos, PLC (Controlador lógico programable en inglés Programmable Logic Controller) y otros mecanismos de control. Aunque un PLC realiza automáticamente un control pre-programado sobre un proceso, normalmente se distribuyen a lo largo de toda la planta, haciendo difícil recoger los datos de manera manual, los sistemas SCADA lo hacen de manera automática. Históricamente los PLC no tienen un modo estándar de presentar la información al operador. La obtención de los datos por el sistema SCADA parte desde el PLC o desde otros controladores y se realiza por medio de algún tipo de red, a continuación esta información es combinada y formateada. Un HMI puede tener también vínculos con una base de datos para proporcionar las tendencias, los datos de diagnóstico y manejo de la información así como un cronograma de procedimientos de mantenimiento, información logística, esquemas detallados para un sensor o máquina en particular, incluso sistemas expertos con guía de resolución de

[3] Módulo de base de datos en tiempo real

[4] Base de dato Histórico

problemas. Desde cerca de 1998, virtualmente todos los productores principales de PLC ofrecen integración con sistemas HMI/SCADA, muchos de ellos usan protocolos de comunicaciones abiertos y no propietarios. Numerosos paquetes de HMI/SCADA de terceros ofrecen compatibilidad incorporada con la mayoría de PLC, incluyendo la entrada al mercado de ingenieros mecánicos, eléctricos y técnicos para configurar estas interfaces por sí mismos, sin la necesidad de un programa hecho a medida escrito por un desarrollador de software. (4)

1.4.1 Aplicaciones que componen un HMI

- **Ambiente de configuración:**

Permite configurar varios procesos o parte de ellos, posibilitando al usuario definir el entorno de trabajo de su aplicación, según la disposición de pantallas requeridas y los niveles de acceso para los distintos usuarios. Dentro del módulo de configuración el usuario define las pantallas gráficas o de texto que va a utilizar, se seleccionan los drivers de comunicación que permitirán el enlace con los elementos de campo y la conexión o no en red de estos últimos. Estas configuraciones son las que más tarde utilizará el visualizador para poner en ejecución la representación gráfica de la planta, fábrica o lugar donde ha sido instalado.

- **Ambiente de ejecución:**

Proporciona al operador las funciones de control y supervisión de la planta. El proceso a supervisar se representa mediante sinópticos gráficos que cambian dinámicamente a diferentes formas y colores, según los valores leídos en la planta o en respuesta a las acciones del operador.

1.4.2 Funcionalidades principales

- **Visualización de despliegue**: Es el encargado de mostrar de forma gráfica los procesos que se siguen en una planta. Esto se realiza a través de objetos gráficos que suelen dividirse en dos grupos, simples y complejos, los primeros se caracterizan por ser objetos estáticos, como circunferencias, líneas, imágenes, y los complejos se caracterizan por variar su estado en el transcurso del tiempo según los datos leídos en la planta, entre los principales se encuentran: las gráficas de tendencia, los relojes y las reglas.

- **Visualización de alarmas**: Es una interfaz gráfica que concentra las condiciones de procesos críticas, medias y bajas presentes en el sistema, cuyo objetivo primordial es

guiar al operador a la detección del origen de la falla y supervisar la ejecución de las medidas de corrección automatizada o manual.

- **Visualización de puntos**: Es la interfaz gráfica encargada de visualizar los detalles de un punto, tales como: nombre, calidad, valor, fecha, entre otros valores.
- **Visualización de eventos**: Muestra las acciones que se han realizado en el sistema.

1.5 Hojas de estilos en cascada

Las Hojas de Estilo en Cascada (CSS por sus siglas en inglés) son utilizadas para dar formato a los diferentes componentes gráficos que contienen las páginas web. Pueden ser utilizados para definir los estilos de texto, tamaños de tablas y otros aspectos, características que anteriormente solo podían ser definidas en lenguaje HTML.

Las hojas de estilo ayuda a los desarrolladores a crear un aspecto uniforme a través de varias páginas en un mismo sitio web, en lugar de definir un estilo para cada tabla y cada caja de texto dentro de una página web el desarrollador puede definir un único estilo el cual será aplicado a una familia de componentes sin la necesidad de realizar el diseño por cada uno de los componentes existentes en una página o en un sitio, esto se logra creando una hoja de estilo CSS la cual podrá ser utilizada por más de una página web que haga referencia a un archivo CSS, facilitando el cambio de estilo entre varias páginas a la vez, por ejemplo un desarrollador puede desear aumentar el tamaño de texto predeterminado de 10 puntos a 12 puntos a través de cincuenta páginas en un sitio, cambiando simplemente el tamaño de texto en el CSS que es referenciado por esas páginas.

Esta tecnología es ideal para la creación de estilos de texto, es también útil para dar formato a otros aspectos del diseño de páginas web, Por ejemplo, se puede utilizar para definir el relleno los campos en una tablas de celdas, el estilo, el grosor y color del borde de una tabla, el relleno alrededor de las imágenes u otros objetos. Ofrece a los desarrolladores Web un control más exacto sobre cómo las páginas Web se visualizarán. Esta es la razón por la cual la mayoría de las páginas web actuales incorporan hojas de estilo en cascada. (6)

1.5.1 Características de las hojas de estilo

El modo de funcionamiento de las CSS consiste en definir, mediante una sintaxis especial, la forma de presentación que se le aplicará a:

- Un sitio web, de modo que se puede definir la forma de todo el sitio de una sola vez.

- Un documento HTML[5] o página, se puede definir la forma, en un pequeño trozo de código en la cabecera, a toda la página.
- Una porción del documento, aplicando estilos visibles en un trozo de la página.
- Una etiqueta en concreto, llegando incluso a poder definir varios estilos diferentes para una sola etiqueta. Esto es muy importante ya que ofrece potencia en la programación. Se puede definir, por ejemplo, varios tipos de párrafos: en rojo, en azul, con márgenes, sin ellos.

1.5.2 Ventajas de usar CSS

Ventajas

Algunas ventajas de utilizar CSS son:

- Control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo.
- Separación del contenido de la presentación, lo que facilita al creador, diseñador, usuario o dispositivo electrónico que muestre la página, la modificación de la visualización del documento sin alterar el contenido del mismo, sólo modificando algunos parámetros del CSS.
- Optimización del ancho de banda de la conexión, pues pueden definirse los mismos estilos para muchos elementos con un sólo selector; o porque un mismo archivo CSS puede servir para una multitud de documentos.
- Mejora en la accesibilidad del documento, pues con el uso del CSS se evitan antiguas prácticas necesarias para el control del diseño (como las tablas), y que iban en perjuicio de ciertos usos de los documentos, por parte de navegadores orientados a personas con algunas limitaciones sensoriales. (11)

1.6 Generalidades sobre los editores de estilo en cascada.

Un editor de estilo es una herramienta que permite crear y editar estilos de cualquier complejidad, grandes o pequeñas, tanto de forma totalmente manual como siguiendo un proceso automático guiado por el propio programa. El mismo incluye ciertas funciones que hacen el trabajo más fácil: código de color para los comandos de sintaxis, autocompletado de

[5] HyperText Markup Language(Lenguaje de Etiquetas de Hipertexto)

tags, utilidad de búsqueda y reemplazo, previsualización de los cambios que se realicen en el código, explorador de ficheros integrado y mucho más. Disponen de una interfaz personalizable que se puede adaptar a los gustos y necesidades del usuario.

1.6.1 Principales Herramientas.

CSS Tab Designer

Con CSS Tab Designer se pueden diseñar fácilmente listas y menús interactivos basados en hojas de estilo, sin la necesidad de conocimientos en programación.

El programa cuenta con una interfaz de uso muy sencilla, en la que se pueden crear tantos elementos como opciones se requieran que contenga el menú a configurar, asignarles nombre, enlace y luego elegir el esquema de diseño con el que se desea aparezcan en la web. Se podrá ver tanto el código fuente como el aspecto gráfico del menú (en la interfaz del programa y en el navegador web), una vez conseguido el efecto deseado solo debe copiarse el código CSS que genera el CSS Tab Designer. En la figura 2 se muestra la interfaz de esta herramienta. (7)

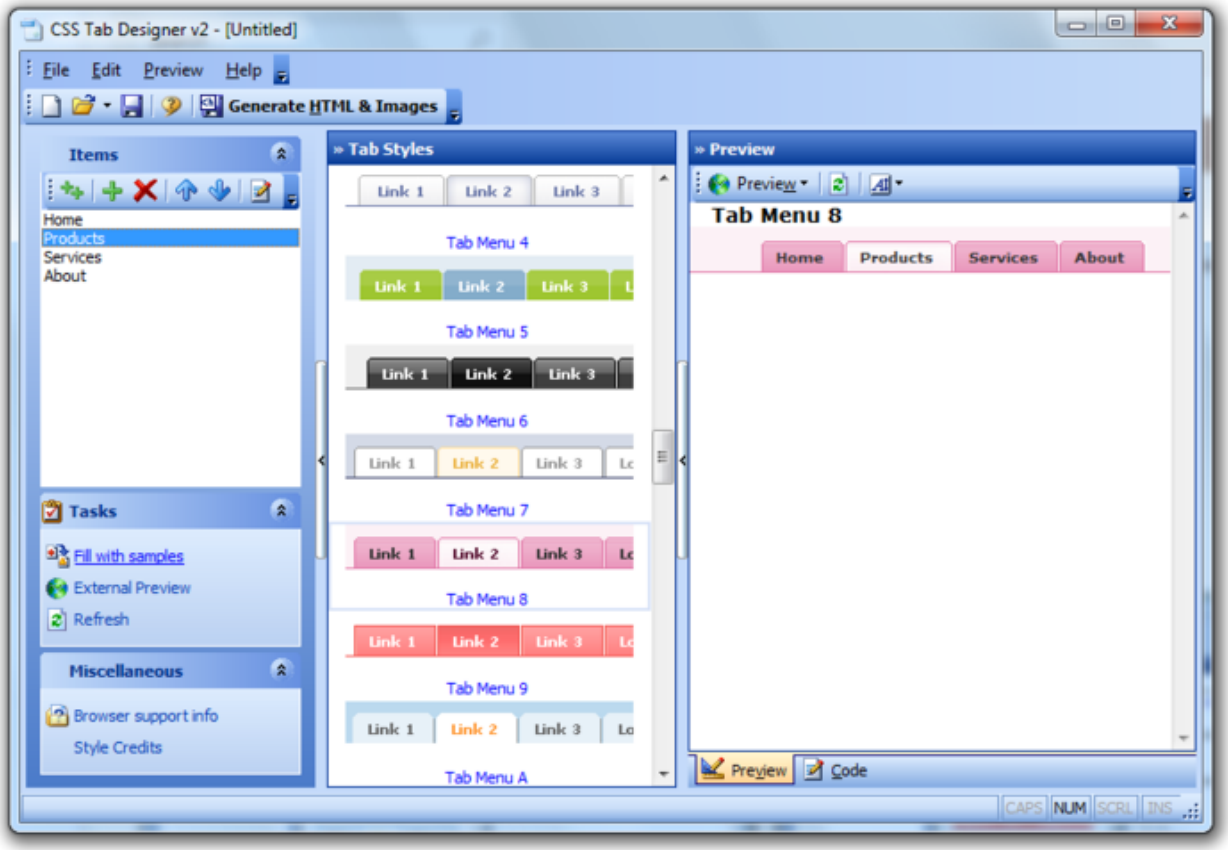

Figura 2.Ejemplo de la herramienta CSS Tab Designer.

EclipseStyle

EclipseStyle permite trabajar con hojas de estilo CSS de manera muy cómoda y efectiva, sin la necesidad de utilizar la engorrosa edición de código fuente en algún editor de texto.

Es también una buena forma de empezar a familiarizarse con este concepto, dado que incorpora una introducción al uso de CSS, así como un completo tutorial del programa.

EclipseStyle cuenta con una interfaz dividida en pestañas, con una ventana de pre visualización para mostrar el resultado gráfico y otra ventana donde aparece el código fuente que EclipseStyle se encarga de generar automáticamente. En la figura 3 se muestra la interfaz de esta herramienta. (8)

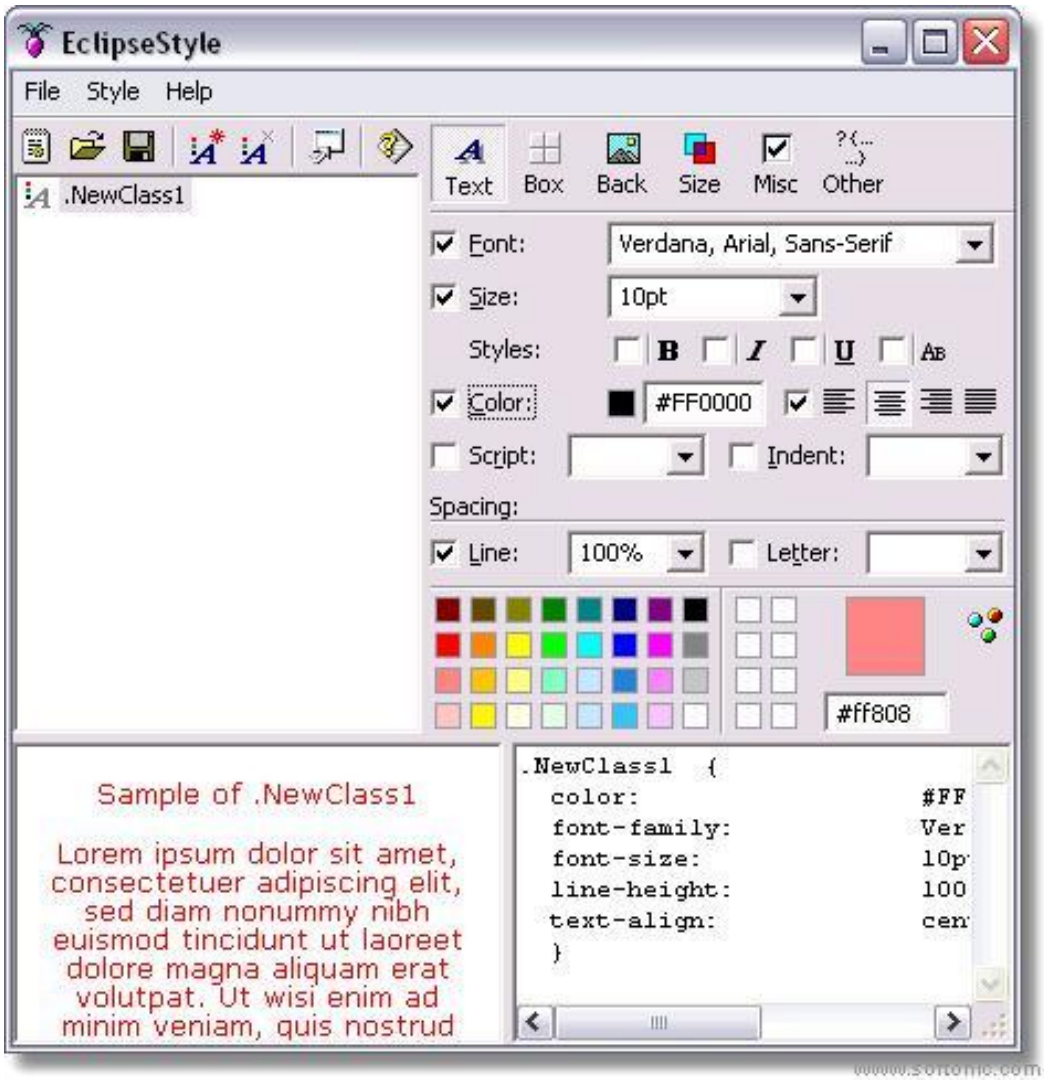

Figura 3 Ejemplo de la herramienta EclipseStyle.

Rapid CSS

Rapid CSS Editor permite crear y editar fácilmente hojas de estilo de cualquier complejidad, grandes o pequeñas, tanto de forma totalmente manual como siguiendo un proceso automático guiado por el programa.

El editor incluye ciertas funciones que hacen el trabajo más fácil: código de color para los comandos de sintaxis, autocompletado de tags, utilidad de búsqueda y reemplazo, previsualización de los cambios realizados en el código, explorador de ficheros integrado y mucho más.

Dispone de una interfaz personalizable que se puede adaptar a las necesidades y gustos del cliente. En la figura 4 observa la interfaz de esta herramienta. (9)

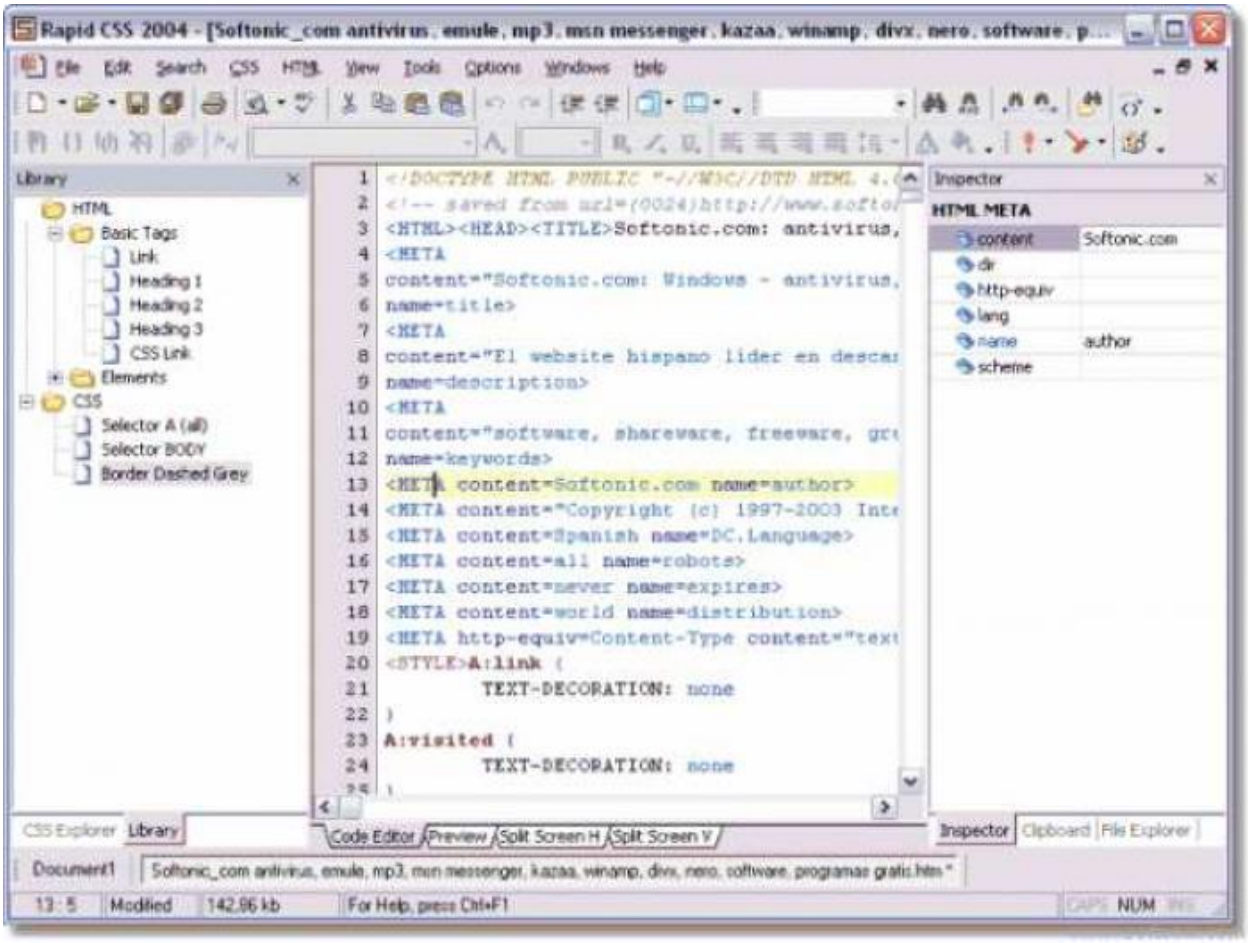

Figura 4.Ejemplo de la herramienta Rapid CSS.

QT Designer

Qt Designer es una herramienta de QT para diseñar y construir interfaces gráficas de usuario (GUI por sus siglas ingles) utilizando los componentes gráficos que brinda dicho framework.

Se pueden componer y personalizar los widgets y cuadros de diálogos de una manera "Lo que ves es lo que tienes"(WYSIWYG-What You See Is What You Get) además de poder probarlos y visualizarlos utilizando varios estilos y resoluciones.

Los widgets y formularios son creados utilizando la herramienta Qt-Designer la cual se integra con el código programado, utilizando el mecanismo de signals y slots el cual permite asignarle comportamientos a los elementos gráficos. Todas las propiedades modificadas en QT Designer pueden ser modificadas dinámicamente en el código. Además características como promoción de widgets y la utilización de plugins personalizados permiten utilizar los componentes creados por terceros con el QT Designer. En la figura 5 se observa la interfaz de esta herramienta. (10)

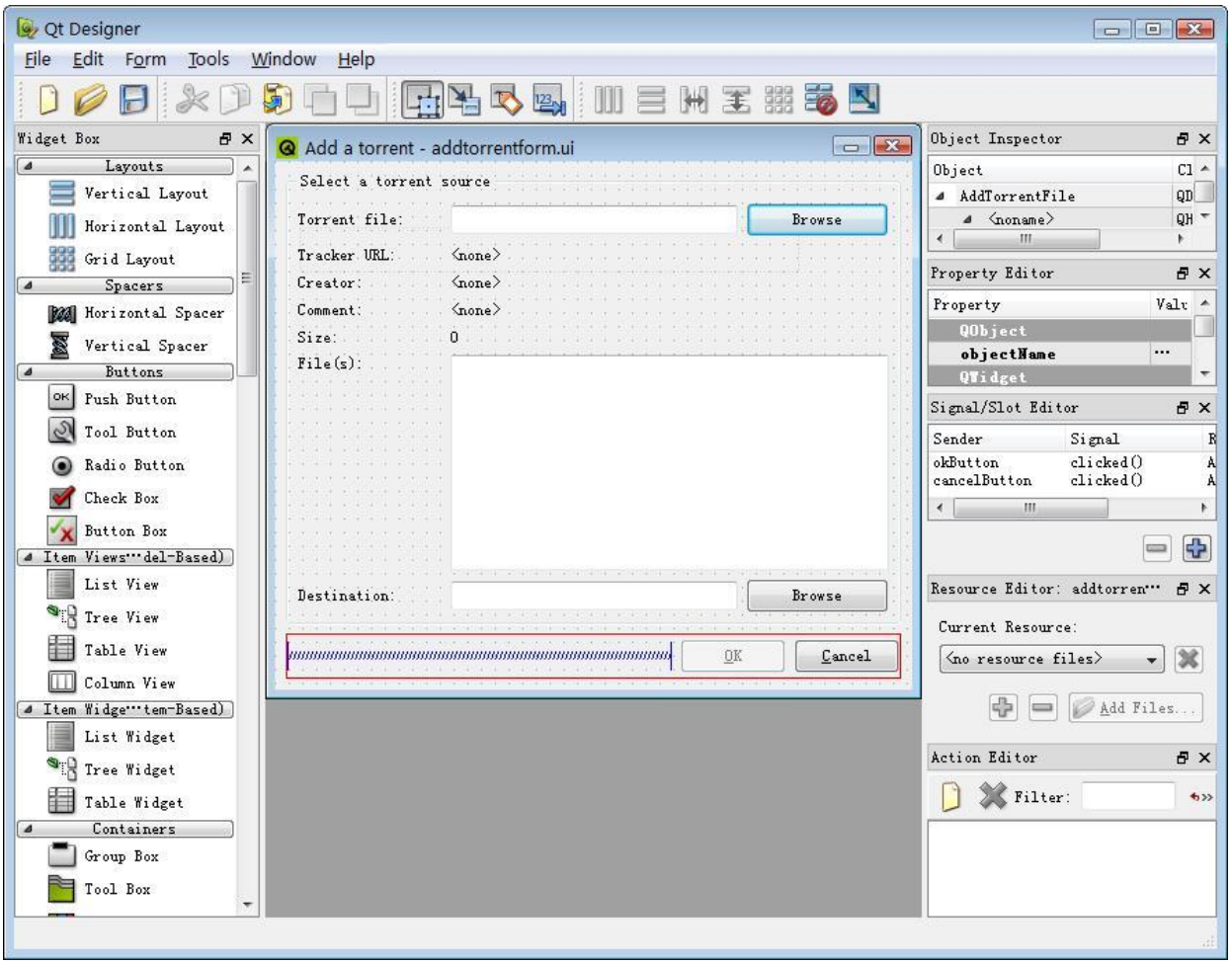

Figura 5.Ejemplo de la herramienta QT Designer.

Para la selección de la herramienta más semejante al sistema que se desea desarrollar y determinar características y funcionalidades que presentará se determinó realizar una tabla

comparativa de las diferentes herramientas existentes teniendo en cuenta los siguientes criterios:

1. Licencia.
2. Compatibilidad con QSS.
3. Sistemas Operativos soportados (S.O.S).
4. Resaltado de sintaxis (R.S).
5. Previsualización.

Herramientas	**Licencia**	**Compatibilidad con QSS.**	**S.O.S**	**R.S**	**Previsualización**
TopStyle Lite	Propietario	No	Windows 7,Windows 95,Windows 98,Windows 98 SE, Windows ME, Windows 2000,Windows NT.	No	Si
CSS Tab Designer	Freeware	No	Windows 7,Windows 98 Windows 98 SE, Windows ME, Windows 2000,Windows XP, Windows Vista, Windows 8	No	No
Rapid CSS	Propietario	No	Windows 7,Windows 2000 Windows XP, Windows 2003	Si	Si
Qt Designer	GPL v3 LGPL v2	Si	Windows XP , Windows 7, OS X, GNU/Linux,	Si	Si

			FreeBSD		

Tabla 1. Comparación de editores de estilo.

Después de haber analizado la tabla anterior se arriba a la siguiente conclusión:

Las herramientas estudiadas en el presente epígrafe brindan una idea de las funcionalidades principales con las que debe contar una herramienta de edición de estilos, en este caso la herramienta que más se asemeja a la solución deseada es el Qt Designer por su compatibilidad con el QSS. Se decidió crear una aplicación la cual pueda ser integrada en un futuro en el HMI-GALBA con el objetivo de tener una aplicación más robusta la cual puede ser adaptada a las necesidades siempre crecientes de los clientes.

1.6.2 Integración del framework QT con las Hojas de Estilo en Cascada.

Las Hojas de Estilo en Qt son un mecanismo poderoso que permite personalizar la apariencia de los widgets. Los conceptos, la terminología y la sintaxis de las hojas de estilo de Qt están fuertemente inspirado en las Hojas de Estilo en Cascada de HTML pero adaptado al mundo de los widgets. (10)

1.7 Metodologías de Desarrollo de Software.

En el proceso de construcción de un software las metodologías de desarrollo de software constituyen un pilar fundamental evitando que este vaya rumbo al fracaso. Se definen como un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software. Una metodología puede seguir uno o varios modelos de ciclo de vida, es decir, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo. La metodología indica cómo hay que obtener los distintos productos parciales y finales.

Se han desarrollado en los últimos años dos corrientes en lo referente a los procesos de desarrollo, los llamados métodos pesados y los métodos ligeros. La diferencia fundamental entre ambos consiste en que mientras los métodos pesados intentan conseguir su objetivo común por medio de orden y documentación, los métodos ligeros (también llamados métodos ágiles) tratan de mejorar la calidad del software por medio de la comunicación directa e inmediata entre las personas que intervienen en el proceso. (12)

1.7.1 Metodologías en la actualidad

El Proceso Unificado de Rational (RUP)

Como sus siglas en inglés lo indican es el Proceso Unificado de Rational, de las metodologías existentes, una de las más generales y completas. Su surgimiento se remonta a aproximadamente treinta años atrás, lo cual da una medida de lo bien estructurada y refinada que está. Fue creada por James Jacobson y se puede decir que unifica los mejores elementos de las metodologías anteriores. Además está preparado para desarrollar grandes y complejos proyectos. Está enfocado al uso del paradigma orientado a objetos y utiliza a UML como su lenguaje de representación visual. Dentro de sus principales características están, que es guiado por casos de usos, que es iterativo e incremental y que centra en la arquitectura el esqueleto del producto. También es válido destacar que se divide en cuatro fases importantes (Inicio, Elaboración, Construcción y Transición).Constituye una metodología tradicional. (13)

Programación Extrema (XP)

Esta metodología llamada Programación Extrema, o Extreme Programming, es otra de las existentes en la actualidad. Mientras que RUP intenta reducir la complejidad del software por medio de estructura y la preparación de las tareas pendientes en función de los objetivos de la fase y actividad actual, XP, como toda metodología ágil, lo intenta por medio de un trabajo orientado directamente al objetivo, basado en las relaciones interpersonales y la velocidad de reacción. (18)

XP propone la descripción de las funcionalidades del sistema a partir de la utilización de las historias de usuarios (HU), que son escritas por el cliente y describen los diferentes escenarios de funcionamiento del software en construcción, es por ello que se hace necesaria la presencia del cliente en el equipo de trabajo. Las iteraciones en XP generalmente son cortas buscando obtener cuanto antes un software funcional, siempre trabajando sólo en las funcionalidades requeridas para la entrega más cercana es una metodología ligera. (19)

1.8 Selección de la Metodología

Para la selección de la metodología de desarrollo de software (MDS) se estableció una matriz de decisión, en la cual se tomaron como parámetros a medir:

1. Tiempo estimado del proyecto (T.E.P).

2. Tamaño del Equipo de desarrollo (T.E.D).

3. Cantidad de artefactos generados (C.A.G).

4. Nivel de uso en la actualidad (N.U.A).

Para cada uno de estos parámetros se estableció una puntuación mínima y máxima de 1 y 5 respectivamente.

M.D.S	T.E.P	T.E.D	C.A.G	N.U.A	Total
RUP	3	3	2	4	12
XP	*5*	4	5	4	18

Tabla 2.Matriz de decisión de Metodologías de Desarrollo.

A partir de lo antes expuesto se escoge XP como la metodología de desarrollo de software que va a regir el desarrollo del Editor de estilos QSS del módulo HMI del SCADA Guardián del ALBA ya que es una metodología ligera hecha especialmente para equipos de desarrollo pequeños buscando una mayor eficiencia, un menor tiempo de desarrollo y una buena calidad del producto final.

1.9 Lenguajes utilizados:

UML 2.0

El Lenguaje Unificado de Modelado (UML) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema de software orientado a objetos (OO). Ofrece un estándar para describir un plano del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Debido a las características del problema a resolver y el componente a desarrollar, se ha decidido utilizar este lenguaje ya que es el más recomendable para el trabajo con lenguajes orientados a objetos. (14)

C++

C++ es un lenguaje de programación, orientado a objetos derivado del C, en realidad un súper conjunto de C, que nació para añadirle cualidades y características de las que carecía, diseñado a mediados de los años 1980, por Bjarne Stroustrup. Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de

compiladores más modernos. Existen también algunos intérpretes como ROOT. Las principales características del C++ son el soporte para programación orientada a objetos, el soporte de plantillas o programación genérica (templates), la portabilidad, brevedad, programación modular, compatibilidad con C y velocidad. Se puede decir que C++ es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos. Además, se trata de un lenguaje de programación estandarizado (ISO/IEC 14882:1998), ampliamente difundido, y con una biblioteca estándar C++, que lo ha convertido en un lenguaje universal, de propósito general, y ampliamente utilizado tanto en el ámbito profesional como en el educativo. Además, posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel entre las cuales se destacan la posibilidad de redefinir los operadores (sobrecarga de operadores) y la identificación de tipos en tiempo de ejecución (RTTI). C++ está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel, sin embargo, es a su vez uno de los que trae menos automatismos (obliga a hacerlo casi todo manualmente al igual que C) lo que dificulta mucho su aprendizaje. (15)

Algunas de sus características distintivas clave incluyen:

- Muy claro, sintaxis legible.
- Fuertes capacidades de introspección.
- La orientación a objetos intuitiva.
- Expresión natural del código de procedimiento.
- La modularidad completa, compatible con paquetes jerárquicos.
- Excepción basada en el manejo de errores.
- Muy altas a nivel de los tipos de datos dinámicos.
- Extensas librerías estándar y módulos de terceros para prácticamente todas las tareas.
- Extensiones y módulos fácilmente escritos en C o C++ Integrable dentro de las aplicaciones como una interfaz de scripting.
- Poderoso y rápido.
- Multiplataforma (Linux / Unix, Windows, OS / 2, Mac, Amiga, entre otros).
- Fácil de aprendizaje.

- Libre uso y distribuibles, incluso para uso comercial.

1.10 Ambiente de Desarrollo Integrado (IDE)

Un entorno de desarrollo integrado, llamado también IDE[6], es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. El lenguaje Visual Basic, por ejemplo, puede ser usado dentro de las aplicaciones de Microsoft Office, lo que hace posible escribir sentencias Visual Basic en forma de macros para Word. Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, PHP, Python, Java, C#, Delphi, Visual Basic. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto, como es el caso de Smalltalk u Objective-C. (16)

1.10.1 Qt Creator versión 2.5

Qt Creator ha sido desarrollado para ser un entorno de desarrollo integrado (IDE) multiplataforma adaptado a las necesidades de los desarrolladores de Qt. Qt Creator se ejecuta en los sistemas operativos de escritorio Windows, Linux/X11 y Mac OS X y permite a los desarrolladores crear aplicaciones para múltiples escritorios y plataformas de dispositivos móviles.

Estas son algunas de las características clave de "Qt Creator":

- **Editor de código sofisticado:** El editor de código avanzado de Qt Creator ofrece compatibilidad con la edición del lenguaje C++ y QML (JavaScript), ayuda sensible al contexto, finalización de código y mucho más.
- **Diseñadores integrados de interfaz de usuario:** Qt Creator ofrece dos editores visuales integrados, Qt Designer para la creación de interfaces de usuario de widgets Qt y Qt Quick Designer para el desarrollo de interfaces de usuario animadas con el lenguaje QML.

[6] Integrated Development Environment(entorno de desarrollo integrado)

- **Administración de proyectos y versiones:** Independientemente de si importas un proyecto existente o creas uno desde cero, Qt Creator genera todos los archivos necesarios. Es compatible con cross-qmake y CMake.
- **Objetivos de ordenadores de escritorio y portátiles:** Qt Creator ofrece compatibilidad con la creación y ejecución de aplicaciones Qt para ordenadores de escritorio y dispositivos móviles. La configuración generada te permite cambiar rápidamente entre los destinos de generación.

Dada las características descritas acerca del entorno de desarrollo Qt Creator se elige este, para la implementación del Editor de estilos QSS, además de las ventajas que ofrece, que el equipo de desarrollo del módulo HMI del SCADA –GALBA esté familiarizado con esta herramienta.

1.11 Framework gráfico de desarrollo Qt versión 4.8

Qt es una biblioteca multiplataforma para el desarrollo de interfaces gráficas, producida por la división de software Qt de Nokia, que entró en vigor después de la adquisición por parte de Nokia de la empresa noruega Trolltech, el productor original de Qt, el 17 de junio de 2008. Es utilizado en aplicaciones como KDE, Google Earth, Skype, Adobe PhotoshopAlbum, VirtualBox y Opie. Qt está desarrollado de forma nativa en C++, pero ha sido portado para ser utilizado desde varios lenguajes de programación como Java, Python, C#, entre otros y está disponible para varios sistemas operativos entre ellos: Windows, Linux y Mac OSx, teniendo un amplio respaldo en el sector empresarial y en las comunidades de desarrollo. Incluye un amplio conjunto de widgets que proporcionan las funcionalidades estándar de interfaz gráfica de usuario, introduce una innovadora alternativa para la comunicación entre objetos, llamados "signals y slots". Puede soportar toda la funcionalidad de interfaz de usuario que requieren las aplicaciones modernas, tales como menús, menús contextuales, arrastrar y soltar, y barras de herramientas acoplables. También propone el patrón de diseño model/view (modelo/vista) para la representación gráficas de los datos con la separación de las funcionalidades introducidas por esta arquitectura, ofrece a los desarrolladores una mayor flexibilidad para personalizar la presentación de elementos y proporciona una interfaz de modelo estándar que permitir una amplia gama de fuentes de datos. Hay que destacar que comenzó como un framework para el desarrollo de interfaces gráficas, pero ya el API de la biblioteca cuenta con tecnologías que facilitan el trabajo a los desarrolladores como son:

- Sistema de pintado.

- Modelos dinámicos de objetos.
- Trabajo con extensiones.
- Contenedores genéricos.
- Estilos.
- Eventos y filtrado de eventos.

Además permite crear aplicaciones de bases de datos independientes de la plataforma que se utiliza. Incluye drivers nativos para Oracle, Microsoft SQL Server, SybaseAdaptive Server, IBM DB2, PostgreSQL TM, MySQL, BorlandInterbase, SQLite, y bases de datos compatibles con ODBC. Incluye widget personalizado para la representación de los datos de las bases de datos. Otras características son: la disponibilidad de código fuente, la excelente documentación organizada en un asistente (QtAssitant), y un editor para el diseño de formularios visualmente (QtDesigner). Qt se distribuye bajo un sistema de licenciamiento dual, que incluye una licencia comercial y una de código abierto bajo los términos de GNU Lesser General PublicLicense (LGPL).

1.12 Herramientas de modelado

Visual Paradigm versión 8.0 para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Los sistemas de modelado UML ayudan a una más rápida construcción de aplicaciones de calidad y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, ingeniería inversa, generar código a partir de diagramas, generación de objetos a partir de bases de datos, generación de bases de datos a partir de diagramas de entidad relación y generar documentación, posee licencia gratuita y comercial Permite interoperabilidad con modelos UML2 (metamodelos UML 2.x para plataforma Eclipse). Se integra con Visio. Posee una plataforma, llamada SDE, capaz de integrarse con Eclipse, NetBeans, Oracle JDeveloper, JBuilder, IntelliJ IDEA, WebLogicWorkshop, Microsoft Visual Studio, entre otras. (1)

Consideraciones Parciales

En este capítulo se hizo un estudio de las principales características de los SCADAs así como lo de sus principales módulos donde se encuentra HMI, el mismo cuenta con dos ambientes el de ejecución y el de configuración. Desde este ambiente de configuración se gestionan toda la

información referente a los sinópticos de las consolas de operación del SCADA que está desarrollado en el framework grafico de QT, además la utilización de hojas de estilos css permiten enriquecer las interfaces de usuario de la aplicación sin la necesidad de recompilar el código haciendo uso de un editor de estilo QSS basado en las características de los estilos en cascada para QT que tienen por nombre QCSS. Esta herramienta se implementará usando el lenguaje de programación C++, usando el IDE QT Creator por las potencialidades de integración con el framework de desarrollo. Y todo esto basado en la metodología de desarrollo ágil XP utilizando la herramienta Visual Paradigm.

2 Construcción de la solución.

Introducción

En el siguiente capítulo se hace una descripción general de la propuesta del sistema, Se definen historias de usuarios que regirán el desarrollo de la solución propuesta y se muestran tablas y diagramas que describen el funcionamiento del sistema, se diseña una arquitectura y se hace una descripción del diseño.

2.1 Solución Propuesta

El desarrollo una herramienta de edición de estilos en cascada basado en los estándares del framework gráfico soportado por el módulo HMI del SCADA-GALBA. Este software permite mostrarle al usuario una paleta de componentes visuales para su edición en un área de trabajo donde se seleccionara cada componente y se le definen los estilos de acuerdo al gusto y necesidades del usuario en el área de edición de estilo, igualmente cuenta con un editor de propiedades el cual permite la modificación de las mismas y un área de edición de los estilos, donde se le aplican estilos a los componentes visuales. Será capaz de brindar la posibilidad de previsualizar los estilos aplicados, posibilitando su exportación a un formato .qss que cuenta con los estilos aplicados para su utilización en el HMI del SCADA-GALBA. Finalmente si se desea realizar algún cambio en los estilos salvados, se carga este fichero y se realizan los cambios correspondientes.

2.2 Exploración

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la entrega del producto. Mientras que los desarrolladores se familiarizan con las herramientas a utilizar durante el desarrollo del sistema. Exploran activamente las posibilidades de la arquitectura del sistema y observan los límites de las tecnologías a utilizar.

2.2.1 Actores del sistema

Actores	Descripción
Usuario	Es el encargado de ejecutar las funcionalidades del Editor de estilos QSS.

Tabla 3. Actores del sistema.

2.2.2 Historias de usuario

En la metodología XP las historias de usuario son utilizadas para especificar los requisitos del software. Se trata de tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. Las HU consisten en descripciones cortas y escritas sin terminología técnica que el cliente hace acerca del software, cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas con ellas el cliente describe y prioriza sus necesidades.

Los programadores estiman el esfuerzo asociado y las dependencias entre ellas y planifican el trabajo desde el punto de vista técnico, las HU son divididas en tareas para las cuales también se realiza una estimación. Teniendo en cuenta el esfuerzo asociado a las HU y las prioridades del cliente se define una versión que sea de valor y que tenga una duración de unos pocos meses.

Los contenidos de las fichas de las HU quedan estructurados de la siguiente forma:

Número: A cada HU se le asigna un número para facilitar su identificación por parte del equipo de desarrollo.

Nombre: Nombre descriptivo de la HU.

Prioridad en Negocio: Grado de prioridad que le asigna el cliente a la HU en dependencia de sus necesidades. Los valores que puede tomar son (Alta, Media o Baja).

Riesgo en Desarrollo: Grado de complejidad que le asigna el equipo de desarrollo a la HU luego de analizarla. (Alto, Medio o Bajo).

Puntos Estimados: Unidades de tiempo estimadas por el equipo de desarrollo para darle cumplimiento a la HU. Una unidad de tiempo equivale a una semana de trabajo de 40 horas.

Puntos Reales: Unidades de tiempo reales que el equipo de desarrollo necesitó para darle cumplimiento a la HU. Una unidad de tiempo equivale a una semana de trabajo de 40 horas.

Iteración Asignada: Número de la iteración en la cual será implementada la HU.

Descripción: Descripción simple sobre lo que debe hacer la funcionalidad a la que se hace referencia.

Historia de Usuario	
Número: 1	**Nombre Historia de Usuario:** Adicionar componente visual
Actor: Usuario	**Iteración Asignada:** 1
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.8
Nivel de Complejidad: Alta	**Puntos Reales:** 0.8
Descripción: El usuario selecciona un componente de la paleta de componentes como un push button, line edit entre otros, posicionando el mismo en el área de trabajo.	
Observaciones:	

Tabla 4. Seleccionar componente visual

Historia de Usuario	
Número: 2	**Nombre Historia de Usuario:** Eliminar componente visual
Actor: Usuario	**Iteración Asignada:** 1
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.4
Nivel de Complejidad: Media	**Puntos Reales:** 0.4
Descripción: El usuario selecciona un componente dentro del área de trabajo y luego presiona la tecla ¨Shift+Del¨y el componente es suprimido.	
Observaciones:	

Tabla 5.Eliminar componente visual.

Historia de Usuario	
Número: 3	**Nombre Historia de Usuario:** Mostrar propiedades de un componente visual.

Actor: Usuario	**Iteración Asignada:** 1
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.9
Nivel de Complejidad: Alta	**Puntos Reales:** 0.9
Descripción: El usuario selecciona un componente dentro del área de trabajo y se muestran las propiedades específicas de este componente en el editor de propiedades.	
Observaciones:	

Tabla 6.Mostrar propiedades de un componente visual.

Historia de Usuario	
Número: 4	**Nombre Historia de Usuario:** Modificar propiedad de un componente visual
Actor: Usuario	**Iteración Asignada:** 2
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.1
Nivel de Complejidad: Media	**Puntos Reales:** 0.1
Descripción: El usuario selecciona el componente, mostrándose las propiedades del mismo, selecciona una y modifica su valor. El componente en el área de trabajo se ve afectado por el cambio realizado.	
Observaciones: El componente no sufre cambios al ser modificada la propiedad de visibilidad.	

Tabla 7. Modificar propiedad de un componente visual.

Historia de Usuario	
Número: 5	**Nombre Historia de Usuario:** Aplicar estilo CSS al componente visual.
Actor: Usuario	**Iteración Asignada:** 2
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.3
Nivel de Complejidad: Media	**Puntos Reales:** 0.3
Descripción: El usuario selecciona el botón nombrado “Aplicar” después de haber editado el estilo en el área de edición de estilos, mostrándose en el área de trabajo los cambios efectuados en los estilos de los componentes.	
Observaciones:	

Tabla 8.Aplicar código CSS

Historia de Usuario	
Número: 6	**Nombre Historia de Usuario:** Borrar contenidos del editor.
Actor: Usuario	**Iteración Asignada:** 2
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.1
Nivel de Complejidad: Media	**Puntos Reales:** 0.1
Descripción: El usuario selecciona el botón nombrado “Limpiar”, eliminando así el contenido del editor de estilo.	
Observaciones:	

Tabla 9. Borrar contenidos del editor.

Historia de Usuario	
Número: 7	**Nombre Historia de Usuario:** Salvar fichero
Actor: Usuario	**Iteración Asignada:** 2
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.2
Nivel de Complejidad: Media	**Puntos Reales:** 0.2
Descripción: El usuario selecciona en el menú la opción “Salvar”, la herramienta debe ser capaz de salvar un fichero .qss con las reglas CSS generadas.	
Observaciones:	

Tabla 10.Salvar fichero.

Historia de Usuario	
Número: 8	**Nombre Historia de Usuario:** Cargar fichero
Actor: Usuario	**Iteración Asignada:** 2
Prioridad de Negocio: Alta	**Puntos Estimados:** 0.2
Nivel de Complejidad: Media	**Puntos Reales:** 0.2

Descripción: El usuario selecciona el botón llamado “Cargar”, la herramienta debe permitir cargar ficheros con configuraciones anteriores para modificar los estilos visuales de los componentes en el área de trabajo.
Observaciones:

Tabla 11. Cargar fichero.

2.3 Planificación de la Entrega.

El propósito de la fase de planificación es establecer un acuerdo entre los clientes y desarrolladores sobre el menor tiempo en que la mayor cantidad de historias de usuario puedan ser realizadas. Para ello se realizó el juego de planeación con el objetivo de maximizar el valor del software producido a partir de la puesta en producción de las características más importantes lo antes posible, en el mismo participan en conjunto desarrolladores y clientes con el fin de determinar rápidamente el alcance de la versión a construir; esto se logra gracias a tener al cliente siempre disponible, lo cual facilita la interacción continua entre los involucrados. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración. (19)

En esta fase el cliente establece la prioridad de cada HU y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debe obtenerse en no más de tres meses.

2.3.1 Estimación de esfuerzo por Historias de usuarios

Teniendo en cuenta la prioridad que tiene una determinada HU en el desarrollo del componente se decide en que iteración será implementada. Las HU que cuentan con mayor importancia por ser funcionalidades indispensables para la aplicación deben ser implementadas en las primeras iteraciones del ciclo de desarrollo.

A continuación se muestra mediante una tabla la planificación de las diferentes HU para cada iteración teniendo en cuenta su prioridad.

No	HU	Prioridad	Esfuerzo Estimado

1	Adicionar componente visual.	Alta	0.8
2	Eliminar componente visual.	Alta	0.4
3	Mostrar propiedades de un componente visual.	Alta	0.9
4	Modificar propiedades de un componente visual.	Alta	0.1
5	Aplicar estilo CSS al componente visual.	Alta	0.3
6	Borrar contenidos del editor.	Alta	0.1
7	Salvar Fichero	Alta	0.2
8	Cargar Fichero	Alta	0.2

Tabla 12. Estimación de esfuerzo por Historias de usuarios.

El tiempo total estimado para el desarrollo de los escenarios en general es de 3 puntos de estimación, que equivalen aproximadamente a 120 horas de trabajo equivalente a 3 semanas.

La metodología XP no requiere la descripción del sistema por medio de diagramas de clase utilizando notación UML, sino que se guía por técnicas como las tarjetas CRC (Contenido, Responsabilidad y Colaboración). Esto no implica que no se utilicen los diagramas para obtener una mejor visión y comunicación entre el equipo de trabajo, siempre y cuando su complejidad no sea alta y defina información importante.

2.3.2 Plan de Duración de las Iteraciones

En el ciclo de vida de un proyecto regido por la Metodología XP se crea el plan de duración de cada una de las iteraciones, en este caso se hace para el único equipo de desarrollo con el cual se cuenta. Este plan tiene como finalidad mostrar el tiempo de duración de cada iteración, así como el orden en que serán implementadas las HU en cada una de las mismas.

Iteración	Orden de las HU a implementar	Duración total de la iteración
Iteración 1	Adicionar componente visual. Eliminar componente visual. Mostrar propiedades de un componente visual.	2.1 semanas 84 horas
Iteración 2	Modificar propiedades de un componente visual. Aplicar estilo CSS al componente visual. Borrar contenidos del editor. Salvar Fichero. Cargar Fichero.	0.9 semanas 36 horas
Total de semanas		3 semanas

Tabla 13.Plan de duración de las iteraciones.

2.3.3 Plan de entrega

En el plan de entregas se establecen los plazos de entregas de las liberaciones del producto, el cual contribuye a la organización del trabajo.

Artefacto	Iteración	Entrega
Versión 1.0	**Final de la primera iteración.**	**Del 4-7 de febrero.**
Versión 1.1	**Final de la segunda iteración.**	**Del 1-5 de abril.**

Tabla 14.Plan de entregas.

2.3.4 Implementación

Una vez identificadas las HU y estimado el esfuerzo propuesto para el desarrollo del sistema de cada una de estas historias, se procede a la planificación de la etapa de implementación del sistema. Teniendo en cuenta la prioridad que tiene una determinada HU en el desarrollo del componente se decide en que iteración será implementada.

Iteración 1

La primera iteración tendrá como objetivo darle cumplimiento a las HU 1, 2 y 3 que representan un mayor valor para el cliente, pues con las mismas se conformará la base de la estructura del negocio. Estas recogen funcionalidades de gran importancia para el proyecto, pues a través de ellas se definen aspectos que serán utilizados luego por las demás funcionalidades.

Tareas de Programación para la Iteración 1:

Adicionar componente visual.	
Número de tarea: 1	**Número de HU: 1**
Nombre de la tarea: Implementación de los componentes, paleta de componentes y conectarlos con el área de trabajo.	
Tipo de tarea: Desarrollo.	**Puntos estimados: 0.8**

Descripción: El desarrollador implementa las clases y métodos necesarios para la representación de los componentes a visualizar en el área de trabajo.

Tabla 15 . Tarea de Programación #1 Iteración 1

Eliminar componente visual.	
Número de tarea: **2**	**Número de HU: 2**
Nombre de la tarea: Implementación de la funcionalidad eliminar un componente del área de trabajo.	
Tipo de tarea: Desarrollo.	**Puntos estimados: 0.4**
Descripción: El desarrollador implementa la funcionalidad eliminar componente del área de trabajo mediante las teclas Shift+Del.	

Tabla 16. Tarea de Programación #2 Iteración 1

Mostrar propiedades de un componente visual.	
Número de tarea: **3**	**Número de HU: 3**
Nombre de la tarea: Implementación de las clases y métodos para la conexión con el componente editor de propiedades.	
Tipo de tarea: Desarrollo.	**Puntos estimados: 0.9**
Descripción: El desarrollador implementa la clase ObjectController permitiendo representar las propiedades de los componentes seleccionados en el área de trabajo.	

Tabla 17.Tarea de Programación #3 Iteración 1

Iteración 2

La segunda iteración está centrada en desarrollar la HU 4, 5, 6, 7, 8 y 9 que son de gran importancia también para el cliente, pues tributan al completo funcionamiento del sistema en desarrollo.

Tareas de Programación para la Iteración 2:

Modificar propiedades de un componente visual.	
Número de tarea: **4**	**Número de HU: 4**
Nombre de la tarea: Implementación de las funcionalidades de modificación de las propiedades de los componentes visuales.	
Tipo de tarea: Desarrollo.	**Puntos estimados: 0.1**
Descripción: El desarrollador debe implementar las funcionalidades necesarias para que al modificar las propiedades de un componente visual los cambios se reflejen en el componente seleccionado en el área de trabajo.	

Tabla 18.Tarea de Programación #4 Iteración 2

Aplicar estilo CSS al componente visual.	
Número de tarea: **5**	**Número de HU: 5**
Nombre de la tarea: Implementar las funcionalidades para aplicar el estilo contenido en el editor CSS a los componentes en el área de trabajo.	
Tipo de tarea: Desarrollo.	**Puntos estimados: 0.3**
Descripción: El desarrollador implementa la funcionalidad de aplicar el estilo QSS representado en el editor de estilo cambiando así el estilo visual de los componentes en el área de trabajo.	

Tabla 19.Tarea de Programación #5 Iteración 2

Borrar contenido del editor.	
Número de tarea: **6**	**Número de HU: 6**
Nombre de la tarea: Implementación de la funcionalidad borrar contenido del editor QSS.	
Tipo de tarea: Desarrollo.	**Puntos estimados: 0.1**
Descripción: El desarrollador implementa el método onClearClicked el cual elimina todo el contenido del editor QSS.	

Tabla 20.Tarea de Programación #6 Iteración 2

Salvar Fichero.	
Número de tarea: **7**	**Número de HU: 7**
Nombre de la tarea: Implementar las funcionalidades para salvar un fichero .qss con el contenido del estilo.	
Tipo de tarea: Desarrollo.	**Puntos estimados: 0.2**
Descripción: El desarrollador implementa la funcionalidad onSaveClicked el cual guarda en un archivo con extensión .qss el contenido del componente editor de estilos QSS.	

Tabla 21.Tarea de Programación #4 Iteración 2

Cargar Fichero.	
Número de tarea: **8**	**Número de HU: 8**
Nombre de la tarea: Implementar las funcionalidades para cargar un fichero .qss.	
Tipo de tarea: Desarrollo.	**Puntos estimados: 0.2**

Descripción: El desarrollador implementa la funcionalidad onLoadClicked el cual carga un fichero .qss con el contenido del estilo y lo representa en el editor de estilos QSS.	

Tabla 22.Tarea de Programación #4 Iteración 2

2.4 Arquitectura del sistema

El diseño de la arquitectura de un sistema, es el proceso por el cual se define una solución para los requisitos técnicos y operacionales del mismo. Este proceso define qué componentes conforman el sistema, cómo se relacionan entre ellos, y cómo mediante su interacción llevan a cabo la funcionalidad especificada.

Para el diseño de la herramienta se seleccionó una arquitectura en 3-capas, la capa de presentación, la de lógica del negocio y la de accesos a datos. Las arquitecturas en capas son técnicas que se utilizan para dividir los sistemas. Esta se seleccionó porque permite tener acopladas las clases de la aplicación y conocer el flujo de datos generado por las mismas.

La arquitectura por capas es una de las técnicas más comunes que los arquitectos de software utilizan para dividir sistemas de software complicados. En un sistema, en términos de capas, cada capa descansa sobre la inferior. En este esquema la capa más alta utiliza varios servicios definidos por la inferior, pero la última es inconsciente de la superior. Además, normalmente cada capa oculta las capas inferiores de las siguientes superiores a esta.

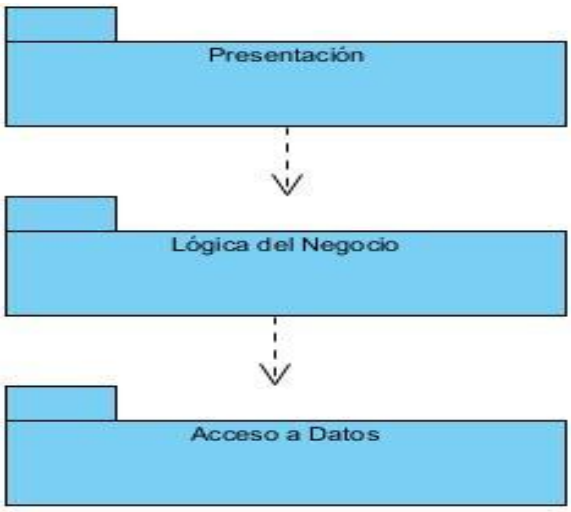

Figura 6.Arquitectura del Sistema.

El editor se encuentra dividido en tres capas como son la de presentación, lógica del negocio y acceso a datos. En la capa presentación están presente las clases: MainWindow, QssEditor, QSsEditorWidget, SampleWidget, WidgetList, CheckBox, ColumnView, ComboBox, DateEdit, DateTimeEdit, DialogButtonBox, DockWidget, DoubleSpinBox, Frame, GroupBox, Label, LineEdit, ListView, ProgressBar, PushButton, RadioButton, ScrollArea, ScrollBar, Slider, SpinBox, TabWidget, TableWidget, TextEdit, TimeEdit, ToolBox, ToolButton, Widget. En la capa de lógica del negocio: Highlighter, ObjectController y en la capa de acceso a datos la clase FileManager.

Los beneficios de trabajar un sistema en capas son:

- Se puede entender una capa como un todo, sin considerar las otras.
- Las capas se pueden sustituir con implementaciones alternativas de los mismos servicios básicos.
- Se minimizan dependencias entre capas.
- Las capas posibilitan la estandarización de servicios.
- Luego de tener una capa construida, puede ser utilizada por muchos servicios de mayor nivel.

2.4.1 Patrones de diseño

Un patrón es una descripción de un problema y la solución, a la que se da un nombre, y que se puede aplicar a nuevos contextos; idealmente, proporciona consejos sobre el modo de aplicarlo en varias circunstancias, y considera los puntos fuertes y compromisos. Muchos patrones proporcionan guías sobre el modo en el que deberían asignarse las responsabilidades a los objetos, dada una categoría específica del problema.

Los patrones de diseño ayuda a como se debe estructurar las clases y objetos para guiar todo el procesos de creación de software. Componen soluciones concretas a problemas que se

presentan durante el diseño de una aplicación. Entre los patrones de diseño más utilizados se encuentran los patrones GRASP y los patrones GoF.

En la construcción del diagrama de clases para la solución se emplearon los siguientes patrones GRASP:

- **Experto:** Asigna una responsabilidad a la clase que cuenta con la información necesaria para cumplirla. Un ejemplo de la aplicación de este patrón en la implementación es la clase ObjectController.
- **Creador:** Guía la asignación de responsabilidades relacionadas con la creación de objetos. Se aplica en todos los casos donde una clase tiene la responsabilidad de crear una nueva instancia de la otra. Un ejemplo donde se utiliza este patrón es en el método createWidget de la clase SampleWidget.
- **Bajo Acoplamiento**: Asigna una responsabilidad para mantener bajo acoplamiento. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Un ejemplo de esto se ve en las clases QSsEditorWidget y QssEditor.
- **Alta Cohesión**: Asigna una responsabilidad de modo que la cohesión siga siendo alta. La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase.

Los patrones GoF utilizados fueron:

- **Observador:** Define una dependencia de uno a muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos. El uso de este patrón se ve reflejado en las clases donde se utilizan señales y slot, por ejemplo en las clases QSsEditorWidget, SampleWidget.

2.5 Tarjetas CRC (Cargo o clase, Responsabilidad y Colaboración)

Las características más sobresalientes de las tarjetas CRC (Clase, Responsabilidad y Colaboración) son su simpleza y ductilidad. Una tarjeta CRC no es más que una ficha de papel o cartón que representa a una entidad del sistema.

Estas tarjetas se utilizan para estructurar las clases y a su vez definir las responsabilidades sobre las mismas, así como la simulación de escenarios en el sistema y tienen la siguiente estructura. (20)

El sistema a desarrollar dispone de diversos objetos, por lo que se definen las siguientes clases:

MainWindow, QssEditor, SampleWidget, WidgetList, Label, Highlighter, Object Controller.

Tarjetas CRC del Sistema

Para el desarrollo del sistema se hizo necesaria la creación de clases para la representación de las entidades del sistema, las cuales se describen en las siguientes tarjetas CRC. Existen otras clases presentes en el sistema pero por poseer una responsabilidad menor se muestran en las tarjetas CRC del Anexo 4.

Una tarjeta CRC representa un objeto. El nombre de la clase se coloca a modo de título en la tarjeta, las responsabilidades se colocan a la izquierda, y las clases que se implican en cada responsabilidad a la derecha, en la misma línea que su requerimiento correspondiente, tal y como muestra la tabla 16.

Clase	
Responsabilidades	Colaboradores

Tabla 23.Targeta CRC

Clase: es cualquier persona, cosa, evento, concepto, pantalla o reporte.

Responsabilidades: las responsabilidades de una clase son las cosas que conoce y las que realizan, sus atributos y métodos.

Colaboradores: los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades.

MainWindow

Contiene la paleta de componentes, el área de trabajo, el editor de propiedades y el editor de qss.	ObjectController,WidgetList, QSsEditorWidget, SampleWidget.

Tabla 24.MainWindow

QssEditor	
Editor de QSS (Qt Style Sheets) con resaltado de sintaxis y autocompletamiento.	Highlighter

Tabla 25. QssEditor.

WidgetList	
Muestra una lista de componentes de prueba.	CheckBox,ColumnView, ComboBox,DateEdit, DateTimeEdit,DialogButtonBox, DockWidget,DoubleSpinBox, Frame,GroupBox,Label,LineEdit, ListView,ProgressBar,PushButton ,RadioButton,ScrollArea,ScrollBar ,Slider,SpinBox,TabWidget,Table Widget,TextEdit,TimeEdit, ToolBox, ToolButton, Widget.

Tabla 26. WidgetList.

Label

Componente que hereda de QLabel que contiene funcionalidades para ser arrastrados.	

Tabla 27. Label.

Highlighter	
Contiene las reglas de resaltado de sintaxis.	

Tabla 28. Highlighter.

QSsEditorWidget	
Contiene editor de qss y las funcionalidades de guardas, cargar y aplicar estilos.	QssEditor

Tabla 29. QSsEditorWidget.

SampleWidget	
Área de trabajo donde se visualizan los componentes de muestra.	CheckBox,ColumnView, ComboBox,DateEdit, DateTimeEdit,DialogButtonBox, DockWidget,DoubleSpinBox, Frame,GroupBox,Label,LineEdit, ListView,ProgressBar,PushButton

	,RadioButton,ScrollArea,ScrollBar ,Slider,SpinBox,TabWidget,Table Widget,TextEdit,TimeEdit, ToolBox, ToolButton, Widget.

Tabla 30. SampleWidget.

2.5.1 Estándares de codificación

Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física, que permite que todos los participantes del proyecto lo puedan entender en menos tiempo y que el código en consecuencia sea mantenible, para facilitar la lectura, comprensión y mantenimiento del código.

La metodología XP se centra en la comunicación entre los miembros del equipo de desarrollo y en especial trata de lograr una mayor comunicación entre los programadores a través del código. Por lo antes mencionado se hace indispensable que se sigan un grupo de estándares de programación, manteniendo el código legible para los miembros del equipo. Para la implementación del sistema desarrollado se siguieron normas y estándares por los que se rige el proyecto de desarrollo y se exponen en el documento titulado *Estándares de codificación para C++* perteneciente al proyecto SCADA Guardián del ALBA del autor Ariel Chávez Lorenzo.

Consideraciones Parciales

Después de conocer el flujo de trabajo del sistema en desarrollo, se diseñó una propuesta de la arquitectura, la cual se implementó en el sistema, especificando las diferentes funcionalidades con que debe contar el software, así como la relación entre los diferentes objetos del sistema. Se desarrollaron las diferentes HU, las mismas se describieron y se llevaron a cabo teniendo en cuenta el orden de prioridad asignado por el cliente para que su posterior implementación siguiera este orden. En su mayoría las HU que suman un total de 8 se dividen en dos grandes grupos, los cuales representan las iteraciones por las que paso el proyecto en su fase de implementación. A partir de cada HU se obtuvieron las diferentes tareas de diseño o

implementación que se realizaron para que cada HU estuviera completa. En el presente capítulo se conformó el estilo de código por el cual se rige el sistema.

3 Validación de la solución propuesta.

Introducción

En el presente capítulo quedan especificados los resultados de la ejecución de las pruebas para probar las funcionalidades descritas en el sistema.

3.1 Pruebas

El desarrollo de software implica la realización de una serie de actividades entre las que se encuentran las pruebas, siendo esta una actividad que garantiza la calidad del producto desarrollado.

La metodología XP anima a probar constantemente tanto como sea posible, siendo las pruebas uno de sus pilares fundamentales. Esto permite aumentar la calidad de los sistemas, reduciendo el número de errores no detectados. También permite aumentar la seguridad de evitar efectos colaterales no deseados al realizar modificaciones y refactorizaciones.

3.1.1 Pruebas de Aceptación

Las pruebas de aceptación tienen como fin validar que el sistema cumple los requisitos básicos de funcionamiento esperado antes de desplegarlas en un entorno real y permitir que el usuario determine la aceptación del sistema.

Caso de prueba de aceptación	
Número: 1	**Historia de Usuario: 1**
Nombre: Adicionar componente visual.	
Descripción: El caso de prueba permite al usuario desde la paleta de componentes, la posibilidad de seleccionar un componente visual.	
Condiciones de ejecución: Debe existir una paleta de componentes, que muestre los componentes visuales a seleccionar.	
Descripción: El usuario selecciona de la paleta un componente como un push button, line edit entre otros, posicionando el mismo en el área de trabajo.	

Entradas/Pasos de ejecución: El usuario oprime el botón primario del mouse sobre el componente visual que desee seleccionar y arrastra el mismo hacia área de trabajo.
Resultado esperado: El sistema adiciona correctamente el componente visual hacia el área de trabajo.
Evaluación de la prueba:

Tabla 31. Caso de prueba Seleccionar componente visual.

Caso de prueba de aceptación	
Número: 2	**Historia de Usuario: 2**
Nombre: Eliminar componente visual.	
Descripción: El caso de prueba permite al usuario desde el área de trabajo, la posibilidad de seleccionar un componente visual para ser eliminado.	
Condiciones de ejecución: Debe existir un área de trabajo que muestre los componentes visuales a seleccionar.	
Descripción: El usuario selecciona un componente dentro del área de trabajo y luego presiona la tecla que alude al eliminar en el teclado y el componente es suprimido.	
Entradas/Pasos de ejecución: El usuario oprime el botón primario del mouse sobre el componente visual que desee eliminar.	
Resultado esperado: El sistema elimina correctamente el componente visual del área de trabajo.	
Evaluación de la prueba:	

Tabla 32. Caso de prueba Eliminar componente visual.

Caso de prueba de aceptación	
Número: 3	**Historia de Usuario: 3**
Nombre: Mostrar propiedades de un componente visual.	
Descripción: El caso de prueba permite al usuario desde el editor de propiedades, la posibilidad de visualizar las propiedades del componente visual.	

Condiciones de ejecución: Debe ser seleccionado algún componente visual del área de trabajo.
Descripción: El usuario selecciona un componente dentro del área de trabajo y se muestran las propiedades de este componente en el editor de propiedades.
Entradas/Pasos de ejecución: El usuario oprime el botón primario del mouse sobre el componente visual del cual desee visualizar las propiedades.
Resultado esperado: El sistema muestra correctamente las propiedades del componente visual mediante el editor de propiedades.
Evaluación de la prueba:

Tabla 33.Caso de prueba Mostrar propiedades de un componente visual.

Caso de prueba de aceptación	
Número: 4	**Historia de Usuario: 4**
Nombre: Modificar propiedad de un componente visual.	
Descripción: El caso de prueba permite al usuario modificar desde editor de propiedades, cualquier propiedad que posee el componente visual.	
Condiciones de ejecución: Debe mostrarse las propiedades de un componente visual.	
Descripción: El usuario selecciona el componente, mostrándose las propiedades del mismo, selecciona una y modifica su valor. El componente en el área de trabajo se ve afectado por el cambio realizado.	

Entradas/Pasos de ejecución: El usuario oprime el botón primario del mouse sobre el componente visual del cual desee visualizar las propiedades, una vez mostrada las mismas pueden ser modificadas.
Resultado esperado: El sistema muestra correctamente las propiedades del componente visual mediante el editor de propiedades permitiendo modificar las mismas de forma adecuada.
Evaluación de la prueba:

Tabla 34. Caso de prueba Modificar propiedad de un componente visual.

Caso de prueba de aceptación	
Número: 5	**Historia de Usuario: 5**
Nombre: Aplicar estilo CSS al componente visual.	
Descripción: El caso de prueba permite al usuario modificar desde el editor de estilos, cualquier propiedad CSS de los componentes visuales.	
Condiciones de ejecución: Debe haber introducido algún código CSS en el editor.	
Descripción: El usuario selecciona el botón nombrado “Aplicar” después de haber editado el estilo en el área de edición de estilos, mostrándose en el área de trabajo los cambios efectuados en los estilos de los componentes.	
Entradas/Pasos de ejecución: El usuario oprime el botón primario del mouse sobre el botón nombrado “Aplicar” y se le aplicara el estilo los componentes en el área de trabajo.	
Resultado esperado: El sistema aplica correctamente el estilo los componentes visuales.	
Evaluación de la prueba:	

Tabla 35.Caso de prueba Aplicar código CSS al componente visual.

Caso de prueba de aceptación	
Número: 6	**Historia de Usuario: 6**
Nombre: Borrar contenidos del editor.	

Descripción: El caso de prueba permite al usuario poder borrar todo el código CSS del edito de estilo.
Condiciones de ejecución: Debe haber contenido en el editor de estilo.
Descripción: El usuario selecciona el botón nombrado “Limpiar”, eliminando así el contenido del editor de estilo.
Entradas/Pasos de ejecución: El usuario oprime el botón primario del mouse sobre el botón denominado “Limpiar” y se borra todo el contenido del editor de estilo.
Resultado esperado: El sistema aplica correctamente los cambios aplicados en el componente visual.
Evaluación de la prueba:

Tabla 36.Caso de prueba Limpiar cambios efectuados.

Caso de prueba de aceptación	
Número: 7	**Historia de Usuario: 7**
Nombre: Salvar fichero.	
Descripción: El caso de prueba permite al usuario poder salvar en un fichero el código CSS generado.	
Condiciones de ejecución: Deben existir reglas CSS en el editor de estilos.	
Descripción: El usuario selecciona en el menú la opción “Salvar”, la herramienta debe ser capaz de salvar un fichero .qss con las reglas CSS generadas.	
Entradas/Pasos de ejecución: El usuario oprime el botón primario del mouse sobre el botón denominado “Salvar” y se guarda en un fichero las reglas CSS del editor de estilos.	
Resultado esperado: El sistema guarda las reglas CSS generadas de manera correcta.	
Evaluación de la prueba:	

Tabla 37. Caso de prueba Salvar fichero.

Caso de prueba de aceptación

Número: 8	**Historia de Usuario: 8**
Nombre: Cargar fichero.	
Descripción: El caso de prueba permite al usuario poder cargar desde un fichero .qss.	
Condiciones de ejecución: Debe haberse seleccionado un fichero .qss anteriormente.	
Descripción: El usuario selecciona el botón llamado “Cargar”, la herramienta debe permitir cargar ficheros con configuraciones anteriores para modificar características o propiedades en los estilos visuales de acuerdo a las necesidades del usuario.	
Entradas/Pasos de ejecución: El usuario oprime el botón primario del mouse sobre el botón denominado “Cargar” y el sistema carga el fichero seleccionado generado .qss.	
Resultado esperado: El sistema carga el fichero seleccionado de manera correcta.	
Evaluación de la prueba:	

Tabla 38. Caso de prueba Cargar fichero.

3.1.2 Resultados arrojados por las pruebas de aceptación.

Caso de prueba de aceptación	Fase 1	Fase2
1	Satisfactorio	Satisfactorio
2	Insatisfactorio	Satisfactorio
3	Satisfactorio	Satisfactorio
4	Satisfactorio	Satisfactorio
5	Satisfactorio	Satisfactorio
6	Insatisfactorio	Satisfactorio
7	Insatisfactorio	Satisfactorio
8	Insatisfactorio	Satisfactorio

Tabla 39.Resultados de los casos de prueba de aceptación.

Consideraciones Parciales

En el presente capítulo se desarrollaron las pruebas de aceptación, cada una de estas fue elaborada a partir de las diferentes historias de usuarios, teniendo en cuanta el resultado de las mismas se concluye que la herramienta se encuentra lista para su puesta en funcionamiento. Se realizaron 9 pruebas de aceptación, todas con resultados satisfactorios del sistema, confirmando esto la integridad y el buen funcionamiento del editor de estilo desarrollado.

Conclusiones

Una vez finalizada la investigación se puede concluir que:

- Se realizó un estudio y análisis de las herramientas y tecnologías utilizadas para la edición de hojas de estilo lo que permitió un desarrollo más rápido y eficaz.
- Se logró la generación de los artefactos propuestos por la metodología seleccionada para el desarrollo de la solución, con los cuales quedaron definidos los requerimientos exigidos por el cliente.
- Se diseñó e implementó un editor de estilos QSS para editar los estilos de los componentes visuales del módulo HMI del SCADA-GALBA.
- Se diseñó un sistema de pruebas para validar las funcionalidades que fueron implementadas, demostraron que los indicadores de calidad cumplieron satisfactoriamente con los requisitos, garantizando su correcto funcionamiento.

Recomendaciones

- Integrar mediante el mecanismo de extensiones el editor QSS al editor de HMI.
- Brindar la capacidad de configuración de los componentes del HMI.
- Añadir una funcionalidad al editor de propiedades que permita crear selectores de clase directamente en el editor de estilos.

Referencias Bibliográficas

1. **Ecured.** ***Ecured.*** **[En línea] Cuba. [Citado el: 12 de octubre de 2013.] http://www.ecured.cu/index.php/Sistema_SCADA.**

2. Sánchez, Raudi Agdel Bacallao. ***Implementación de un módulo de generación de reportes para sistemas de supervisión, control y adquisición de datos.*** **La Habana : s.n., 2008.**

3. ***Sistema SCADA.*** **Rodríguez. España : Marcombo, 2007, Vol. II.**

4. iaci.unq.edu.ar. ***iaci.unq.edu.ar.*** **[En línea] [Citado el: 8 de 1 de 2013.] http://iaci.unq.edu.ar/materias/laboratorio2/HMI%5CIntroduccion%20HMI.pdf.**

5. PDVSA. ***Introducción a la arquitectura del guardián del alba y descripción de módulos.*** **Venezuela : s.n., 2009.**

6. techterms. ***techterms.*** **[En línea] [Citado el: 5 de 1 de 2013.] http://www.techterms.com/definition/css.**

7. css-tab-designer.softonic.com. ***css-tab-designer.softonic.com.*** **[En línea] [Citado el: 8 de 1 de 2013.] http://css-tab-designer.softonic.com/.**

8. eclipsestyle.softonic.com. ***eclipsestyle.softonic.com.*** **[En línea] [Citado el: 8 de 1 de 2013.] http://eclipsestyle.softonic.com/.**

9. rapid-css-editor.softonic.com. ***rapid-css-editor.softonic.com.*** **[En línea] [Citado el: 8 de 1 de 2013.] http://rapid-css-editor.softonic.com/.**

10. Qt Docs home. ***Qt style sheets.*** **[En línea] [Citado el: 8 de 1 de 2013.] http://qt-project.org/doc/qt-4.8/stylesheet.html.**

11. Consorcio world wide web. ***Guias breves de hojas de estilo.*** **[En línea] [Citado el: 6 de 1 de 2013.] http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo.**

12. alarcos. [En línea] [Citado el: 12 de 1 de 2013.] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema04.pdf..

13. MolpeceresTouris, Alberto. ***Procesos de desarrollo: RUP, XP y FDD.*** **2002.**

14. Guillermo Javier Lafuente. UML Unified Modeling Languaje. [En línea] [Citado el: 15 de 1 de 2013.] http://gidis.ing.unlpam.edu.ar/personas/glafuente/uml/uml.html...

15. organizator. ***El lenguaje C++.*** **[En línea] http://www.zator.com/Cpp/E1_2.htm.**

16. Qt creator IDE. [En línea] http://qt.digia.com/product/developer-tools/.

17. Paradigma visual para UML. ***Free Download Manager.*** **[En línea] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28LE%29_14725_p/.**

18. Beck, Kent. *Extreme Programming Explained.* 1999.

19. Joskowicz, Jose. *Reglas y Practicas en eXtreme Programming.* 2008.

20. Y. M REYES. Espacio de comunicación e intercambio para la comunidad técnica cubana de PostgreSQL. [En línea] 2011. http://rcci.uci.cu.

GLOSARIO DE TÉRMINOS

Software: Programas, procedimientos y reglas para la ejecución de tareas específicas en un sistema de cómputo.

Generador de reportes: Herramienta que permite generar reportes a partir de datos primarios.

Diseño del editor: Apariencia final con la cual será generada el editor.

HTML (Hyper Text Markup Language): Lenguaje con el que se escriben las páginas web. Permite escribir texto de forma estructurada, y que está compuesto por etiquetas, que marcan el inicio y el fin de cada elemento del documento.

SCADA (Supervisory Control And Data Acquisition): Control Supervisor y Adquisición de Datos.

Framework: En desarrollo de software, es una estructura en la cual pueden ser desarrollados distintos tipos de proyectos de software. Normalmente incluye programas de ayuda, bibliotecas de código y lenguajes de programación.

Caso de Prueba: Conjunto de condiciones o variables bajo las cuáles el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio.