J_ROHI SISTEMA INTEGRADO

Presentado por:

EDISSON JAVIER OQUENDO MARIN

SOFTWARE J_ROHI SISTEMA INTEGRADO

FUNDACIÓN UNIVERSITARIA SAN MARTIN SABANETA

FACULTAD DE INGENIERÍA DE SISTEMAS

COLOMBIA

2015

NOTA DE ACEPTACIÓN:

NOTA DE ACEPTACIÓN:

_____

_____

_____

_____

Firma de presidente del Jurado

_____

Firma del Jurado 1

_____

Firma del Jurado 2

_____

Sabaneta, mayo de 2015

1. RESUMEN

Lo fundamental para un proyecto de desarrollo de software, es cada uno de los requerimientos que soporta su estructura y definen su dinámica, para garantizar que el producto final logre satisfacer las necesidades y expectativas de los usuarios, para esto se debe construir con elementos tomados del ambiente real del negocio, los cuales sean consistentes y actuales. Es por esta razón que se toma como referente el objeto social de la fundación, el cual, facilita la recolección de información necesaria para resaltar de manera iterativa los diferentes puntos que fueron tomados como base para la elaboración del software. Toda la información recopilada se toma como base para la construcción de un sistema que sea no solo útil sino que sus bases sean sólidas, por tal motivo se hace indispensable la correcta recolección de información para llegar al análisis exacto de lo que busca la corporación y poder suplir a cabalidad sus necesidades.

En este documento se presenta la obtención de los requerimientos como solución a un sistema de información que le permita en gran parte la reducción del reproceso y costos dentro de la operación de la Fundación, permitiendo la valorización de tiempo y esfuerzos acordes a la realidad, y facilitando la realización de un producto con calidad orientado a un sector de la sociedad; el cual no recibe un apoyo directo del estado.

2. ABSTRACT

The basis of a software development project what each contribute requirements supporting structure and its dynamics defined to ensure that the final product meets the needs and expectations of users, should be built with requirements taken from actual environment business, which are consistent and current. It is for this reason that modeling is taken as the objects of the foundation, which facilitates the collection of information needed to develop iteratively different points that were taken as a basis for the development of software. All information collected is taken as the basis for building a system that is not only useful but its foundations are solid, for that reason the correct collection of information is essential to arrive at accurate analysis of what the corporation and seeks to fully meet their needs.

This information forms the basic foundation of requirements elicitation. While obtaining requirements is the phase of which depends largely on the degree of success of the project, therefore it is necessary to apply a flexible, iterative and user-oriented method, which makes use of a whole and techniques tools that facilitate the collection, analysis, specification and validation of product requirements. This document provides a business modeling and requirements as obtaining an information system solution that allows you to largely reduce rework and costs within the operation of the foundation is presented, allowing the estimation of time and effort commensurate with reality, and facilitating the development of a quality product aimed at a segment of society which receives no direct state support.

3. TÍTULO DEL PROYECTO

"J_ROHI"

SISTEMA INTEGRADO FUNDACIÓN ELENA Y JUAN

Significado: El nombre del proyecto, se toma como inspiración en el significado y nombres que le han dado a DIOS en el pasar de los tiempos, dicho nombre es tomado de las referencias bíblicas.

El proyecto se llama en forma abreviada J_ROHI, sin embargo su nombre completo traduce "JEHOVÁ ROHI" que como lo expreso anteriormente y tomando el objeto social de la corporación Elena y Juan, aplica cada una de las cualidades descritas en su significado. *El Señor es mi Pastor*, teniendo como aplicación: *El señor protege, provee, dirige guía y cuida de su pueblo. Dios nos cuida tiernamente como un pastor poderoso y paciente*.

Cualidades que se hacen evidentes en la misión y visión de la corporación ELENA Y JUAN. (Ver Ilustración 1)

Ilustración 1: Titulo del Proyecto

CONTENIDO

LISTA DE TABLAS E ILUSTRACIONES

INTRODUCCIÓN

A continuación en el desarrollo de este proyecto se describe detalladamente el proceso que se realizó en la Corporación Elena y Juan, lográndose la participación de la Directora General Dra. María Cristina Calle Calle, donde se realizaron variadas entrevistas para detallar las necesidades en cuanto a infraestructura y sistemas de información fuera posible que ayudasen a la optimización de los procesos en la organización y especificar sus debidos requerimientos.

Luego de analizados todos los requerimientos de la corporación, se procedió al análisis de la información y estructuración de las posibles soluciones al problema presentado. Se aplicó la Ingeniería de software para articular la funcionalidad de un sistema de información, donde se logró llegar a un producto final, cubriendo las necesidades planteadas dentro de las cuales, fueron la estandarización de los procesos, controles administrativos de la organización, infraestructura de red, y contar con un sistema de información que permita el control de manera eficaz de la información, que será requerida de forma oportuna para cada uno de los niños, niñas y adolescentes que protege de manera integral la fundación.

1. PROBLEMA

1.1 Planteamiento del Problema

La Corporación Elena y Juan, desde que inició su labor ha direccionado todo su esfuerzo en ayudar la infancia desprotegida. Este trabajo comenzó desde el año 1992, brindando apoyo a los niños del área Metropolitana y del Valle de Aburra, que son víctimas de la violencia sociopolítica, abandono, desnutrición, maltrato, abuso sexual y laboral. Su Fundadora y directora la Señora María Cristina Calle gracias a su entrega desinteresada, al amor y liderazgo; al igual que otras personas como el padre Álvaro Murillo, fundador y director de "Amigos de la Infancia" y muchas otras obras, se ha venido fortaleciendo para seguir adelante con una nueva Obra, "Hogar Elena y Juan". En donde a la fecha promueve el desarrollo de sus capacidades individuales mediante una asistencia educativa y pedagógica estructurada en valores, y contribuir a la prevención y promoción de la salud. (Elena Y Juan Corporación, s.f.)

En la actualidad dicha fundación cuenta con alrededor de 200 niños, niñas y jóvenes, que necesitan un hogar; este hogar funciona 24 horas de lunes a viernes, y los fines de semana se atienden los servicios básicos de algunos niños. Todos reciben los servicios de alojamiento, alimentación, educación, salud y recreación.

1.2. Formulación del problema

Es necesario resolver mediante un sistema de información el manejo documental, el asiento de donaciones y actividades correspondientes a la asistencia educativa, salud y pedagógica de la Fundación Elena y Juan. Este sistema debe permitir obtener información oportuna, eficaz y en

tiempo real; convirtiéndose una ayuda fundamental para la toma de decisiones en el mejoramiento continuo del portafolio de servicios.

1.3. Sistematización del Problema

La fundación es una entidad que depende del apoyo de instituciones, o de grupos profesionales, o particulares que se vinculan a los programas. Vemos la necesidad de crear un sistema de información que le permita estar a la vanguardia tecnológica actual en donde le ofrece de primera mano un valor agregado a la fundación y especialmente al área administrativa en sus actividades operativas, tal como se encuentra en el organigrama de la institución. Este sistema tendrá como alcance los siguientes módulos:

a) Control Documental de cada Infante que se encuentra en la Institución

b) Registro en el área de salud.

c) Registro de Información de Nutrición

d) Registro pedagógico

e) Registro de los ingresos del Plan Padrino.

f) Registro y Control de los tipos de donaciones que se reciben en la corporación

g) Extracción de información de manera oportuna y veraz cuando se requiera, de aquellos movimientos que se hagan, los servicios que se prestan, como también de los diferentes tipos de ingresos que se reciben por el concepto del plan padrino, alimentación, entre otros.

2. JUSTIFICACIÓN

El presente informe tiene como principio la necesidad de explorar un vacío en dos aspectos de la realidad contemporánea dentro de los proyectos de software: el primero, es el hecho de que la fundación requiere de ventajas competitivas para llevar a cabo sus procesos de análisis y control de cada una de la áreas que atiende en beneficio de los menores de edad; el segundo, es que este proceso debe ser mejorado con herramientas específicamente enfocadas al aspecto de herramientas disponibles en el medio como es el internet, sistemas operativos, instancias de base de datos en la web bajo el enfoque de Open Source o también llamado Software Libre. (Perens, 1998)

Teniendo en cuenta lo anterior este proyecto es creado debido a la necesidad de tener un estudio desarrollado para tener un análisis claro de los requerimientos del cliente para lograr de esta manera un correcto desarrollo del software dentro de los proyectos de la corporación Elena y Juan; esto nos va a permitir crear nuevos elementos que sirvan de ayuda para afrontar las dificultades que se presenten y así mismo lograr oportunidades de mejora y contar con las tendencias tecnológicas que nos brinda el medio, contribuyendo con un valor agregado al beneficio de la fundación y de todos los que hacen parte de ella.

El proyecto está relacionado con los siguientes aspectos:

- Es una Idea Innovadora y su finalidad es para entidades sin ánimo de lucro.
- Es un proyecto Real
- Es aplicable de Inmediato
- Es un Software Multiempresa
- Utiliza toda la tecnología de sistemas de información existentes en el mercado (Redes de

computadoras, Internet, Sistemas de Información, Herramientas de Diseño, entre otros).

- Tiene un fuerte matriz hacia la sociedad
- La aplicación se crea una sola vez y su uso es orientado para cualquier ciudad, estado o país. Que su fin sea su misma actividad económica de la Corporación Elena y Juan.

3. OBJETIVOS

3.1. Objetivo General

Diseñar e implementar un sistema de información, que considere las necesidades actuales de la Fundación y los recursos económicos con los que cuenta, que son mínimos, en cuanto a la inversión tecnológica de información e infraestructura.

3.2. Objetivos Específicos

- Modelar y elaborar un prototipo con el alcance de mejorar los procesos del área administrativa de la Fundación.
- Construir un módulo que permita el reemplazo del proceso manual de archivo por un modelo virtual de carpetas, que minimicen las actividades operativas y físicas haciendo más efectiva la búsqueda de la información puntual.
- Optimizar el proceso de donaciones recibidas por la Fundación para que el sistema controle los mecanismos, medios o tipos de donaciones que se reciben y así determinar, en tiempo real, estadísticas de resumen útiles para la toma de decisiones gerenciales.
- Concurrir con un registro o bitácora en los diferentes programas que realiza la Fundación.
- Implementar una agenda que permita la creación, supervisión y control de los eventos que coordina la entidad. Para tal efecto el sistema contará con una región en donde se programarán las actividades de la Fundación diariamente.
    - Poner en funcionamiento una opción en el sistema donde se le dé acceso a cualquier usuario para que descargue y acceda a los módulos, así como, a las diversas herramientas que tendrá el sistema, siendo así información de primera mano.

4. MARCO TEÓRICO

4.1. Introducción a la Ingeniería de Software

Los elementos de un sistema informático se clasifican de forma tradicional en hardware y software. Los primeros, componentes o equipos físicos, generalmente son de relativa clasificación sencilla y fácilmente identificables. El segundo tipo, el referente software, tiene la peculiaridad de ser difícilmente clasificado y caracterizado, puesto que en los contextos actuales una clasificación de software simplemente como "todo aquello que no es hardware" no tiene cabida dentro de ámbito científico. En efecto, el software puede incluir programas que se encargan de controlar el funcionamiento de un ordenador aunque puede incluir otros elementos como documentos, bases de datos o algunos procedimientos de operación o mantenimiento de relativa frecuencia programada (Cerrada, 1997).

El software ha entrado en una especie de crisis porque no ha dado a una evolución estable sino que ha estado en permanente cambio, provocando una reducción en el costo de los equipos informáticos a la vez que se incrementa su capacidad y ha forzado a un incremento de la complejidad de las aplicaciones de software para distintas áreas. Éste proceso seguramente continuará indefinidamente, generando cada vez más revoluciones y avances que es necesario identificar e interpretar para poder estar a la vanguardia.

Por ello, los avances continuos en la llamada *Ingeniería de Software* resultan poco a poco insuficientes para abarcar todos los elementos que surgen casi que diariamente. El término aquí introducido a veces se discute puesto que conforma una disciplina que al enfocarse en la ingeniería debe haber alcanzado alguna madurez en las técnicas a aplicar, utilizando un fundamento científico válido y no una tendencia casi artesanal (Cerrada, 1997).

El software constituye un producto de consumo utilitario o masivo, e interviene de forma casi imprescindible en la gestión de procesos productivos o en la moderación de muchos procedimientos de tipo industrial. Dos de los grandes problemas a los que se enfrenta el software son los relacionados con calidad y productividad, que se han evidenciado y desarrollado especialmente en los últimos treinta años. La alta demanda de software que ha habido obliga a mejorar la predictibilidad y productividad de los sistemas de software (Campderrich Falgueras, 2003).

Respecto a la calidad, el principal problema es que muchas veces el software utilizado es de alta complejidad, en comparación con otros tipos de productos, provocando que muchas veces no sea posible utilizar y comprobar el software en todas las configuraciones posibles, y es algo relevante puesto que en la actualidad se asume que los sistemas de calidad ya deben ser imprescindibles y lo suficientemente sensible como para convertirse en factor de competitividad en los mercados cada vez más saturados y exigentes. Por ello, la ingeniería de software no puede escapar a estas condiciones.

Frente a la productividad, cualquier tipo de fabricación en serie requiere que más que elevar la producción pueda controlarse de forma remota o directa, muchas veces en tiempo real. Existen productos muy complejos donde el grado de productividad requerido a alcanzar es complicado, y el software debe ser lo suficientemente sensible y capacitado como para medir dicha productividad y proponer sistemas correctivos y aplicaciones diversas a todas las situaciones que pudieran presentarse. En este aspecto, la ingeniería de software debe mejorar sus métodos de solución de problemas y generar respuestas de software que cumplan a cabalidad con todas las condiciones requeridas (Campderrich Falgueras, 2003)

4.2. Ciclo de Vida en Cascada de un *Software*

Más que programación, un software requiere pasar por unas etapas determinadas donde el conjunto de ellas constituyen el ciclo de vida del mismo. Sin embargo, hay distintos modelos de vida de software, lo que hace que ninguno sea el ideal o que siempre existirán limitaciones. Es indispensable que en todo proyecto siempre se desarrolle, si compete, un ciclo de vida definido para poder cumplir con los plazos exigidos y respetarlos límites de recursos asignados. Del mismo modo, el desarrollo de un software dentro de un ciclo de vida permite seguir con los lineamientos de las garantías de calidad y certificaciones afines. La siguiente gráfica muestra las diversas etapas del ciclo de vida (Campderrich Falgueras, 2003):

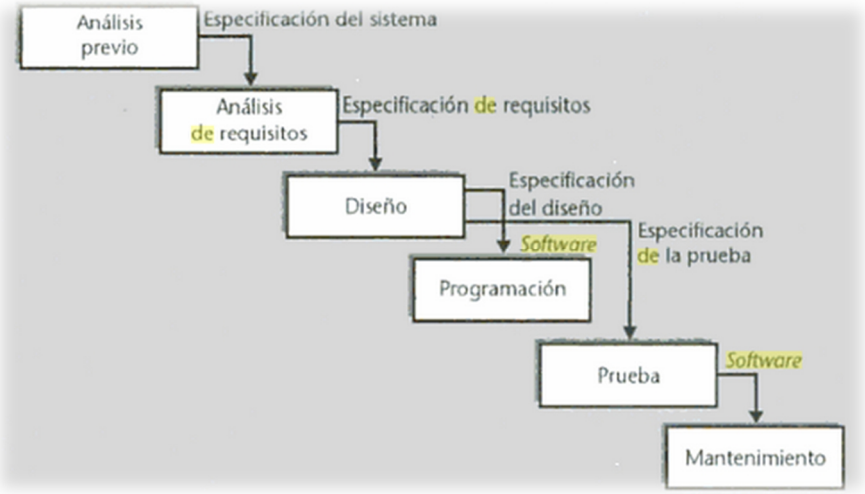

Ilustración 2. Ciclo de vida en cascada de un producto.

Fuente: Campderrich Falgueras (2003)

El gráfico anterior es un poco estático en cuanto al ciclo de vida. Las diversas etapas se explican a continuación (Campderrich Falgueras, 2003):

4.2.1. Análisis previo

También llamada Análisis de Sistemas o Ingeniería de Sistemas, se refiere a la definición de los grandes rasgos (los más generales) del sistema de software que se va a emplear para dar soporte informático a las actividades definidas de los usuarios dentro del marco empresarial u organizacional. Es importante ésta etapa porque ayuda a realizar bosquejos del plan de trabajo que, en conjunto con la etapa siguiente, permiten obtener un resultado más preciso de los requerimientos y la satisfacción de los mismos por parte del software a desarrollar.

Del mismo modo, en este sistema deberá funcionar cierto entorno de hardware y red, realizar intercambios de información con otros software o bases de datos que previamente existían y consignar toda la información con el objetivo de dejar constancia del proceso seguido. Los recursos necesarios para el desarrollo de software deben ser considerados y los condicionamientos temporales (como previamente se había indicado), restricciones eventuales y condiciones adicionales que sea necesario respetar y las viabilidades (técnica, económicas, científicas y legales) del proyecto de desarrollo de software.

4.2.2. Análisis de requisito

Llamada a veces simplemente “Análisis”, busca definir detalladamente las necesidades de información que tendrá que resolver el software sin tener en cuenta, al menos momentáneamente, los medios técnicos para el desarrollo del mismo. Así como es posible reutilizar el lenguaje de programación, también puede hacerse con los gestores de bases de datos, componentes, etc. La figura que es responsable del análisis (el analista de software) debe tener o adquirir los conocimientos generales de dominio de la aplicación y obtener información de los usuarios y otras fuentes que le permitan hacerse una idea precisa de las funciones y requisitos

generales del software. La información obtenida es la materia prima de elaboración de un documento que se denomina “Especificación de Requisitos” (Documento de Requisitos de Software o *Software Requirements Document – SRD)* que debe cumplir con dos funciones: especificar qué se debe hacer con el software (con precisión adecuada como para desarrollarlo) y servir de base para que funcione como una forma de contrato entre el equipo de desarrollo del software y los posibles usuarios del mismos.

4.3. Ciclos de Vida Iterativos e Incrementales

El ciclo de vida no siempre es en cascada, este tipo ha sido muy criticados, por lo que se han propuesto modelos alternativos. El ciclo de vida en cascada no es muy realista porque considera que cada etapa comienza una vez ha finalizado la anterior, y actualmente se sabe que eso no es así: muchas etapas suceden simultáneamente o en un tiempo de desarrollo de una etapa en particular. Asimismo, se considera que una vez ha terminado una etapa ya queda un porcentaje cuantitativo de lo que queda por hacer de proyecto, y a veces el cumplimiento de una etapa origina nuevos problemas y retos que no implican un avance real en el desarrollo del proyecto

Muchos proyectos son muy flexibles en cuanto a sus funciones porque no siempre deben ser muy explícitas en sus definiciones, pero son proyectos que carecen de base sólida para calcular costes y tiempos y tienen amplios márgenes de error. En estos proyectos los análisis de requisitos suelen ser incompletos o cambian antes de que se haya comenzado a construir el software. Una especificación de requisitos insegura e incompleta (a veces hasta poco real) generará diseños y programaciones problemáticos que provocan retrasos y aumentos considerables de coste para el trabajo que se debe hacer y el trabajo imprevisto que debe

realizarse para solventar éstos errores. Elaborar requisitos estables y completos desde el primer momento es imposible porque los usuarios a menudo no conocen el entorno completo de desarrollo del software o no saben muy bien qué quieren que el software les ayude a completar; y el trabajo de consolidación de las peticiones de los usuarios nunca será perfecto (Campderrich Falgueras, 2003).

Como solución a los ciclos de vida en cascada, se han propuesto ciclos de vida con prototipos de software, es decir, construcciones con herramientas y técnicas que priorizan la rapidez y facilidad de modificación antes que una correcta eficiencia de funcionamiento, y que pretenden que los usuarios puedan ver cómo es el contenido o apariencia de los resultados arrojados por algunas de las funciones del futuro software. Los prototipos ayudan a que los usuarios confirmen que lo que se les muestra es lo que necesitan y puedan tener un pedido de software por comparación, para lo que se preparará una nueva versión del prototipo. Una vez se confirman las características y requisitos del nuevo prototipo, se puede comenzar un desarrollo tipo cascada pero que parte de una base mucho más sólida (Campderrich Falgueras, 2003). Para reducir el costo de desarrollo de un prototipo respecto al sistema final, es posible realizar las siguientes acciones (Cerrada, 1997):

- Limitar las funciones y desarrollar sólo unas pocas.
- Limitar la capacidad del sistema, de modo que procese sólo algunos datos.
- Limitar la eficiencia, permitiendo su operación de forma lenta.
- Evitar limitaciones de diseño utilizando un soporte de software más potente.
- Reducir la parte a desarrollar utilizando un apoyo de software más potente.

Los prototipos pueden ser de dos tipos (Cerrada, 1997):

4.3.1. Prototipos rápidos

Su única finalidad es la adquisión de experiencia sin pretender ser aprovechados como producto. Se denominan prototipos *throw-away* (usar y tirar) porque su funcionalidad o capacidad es bastante limitada. Generalmente, éstos prototipos son aprovechados en las fases de análisis y/o diseño de un sistema para experimentar con algunas alternativas y poder garantizar que las desiciones de diseño de software futuras que se tomen son las correctas. Cuando se completan éstas fases, el sistema se codifica de nuevo desde cero, es decir, no hay un aprovechamiento del código del prototipo. La siguiente figura muestra el uso de un prototipo de éste tipo dentro de un ciclo de vida en cascada, en la fase de análisis:

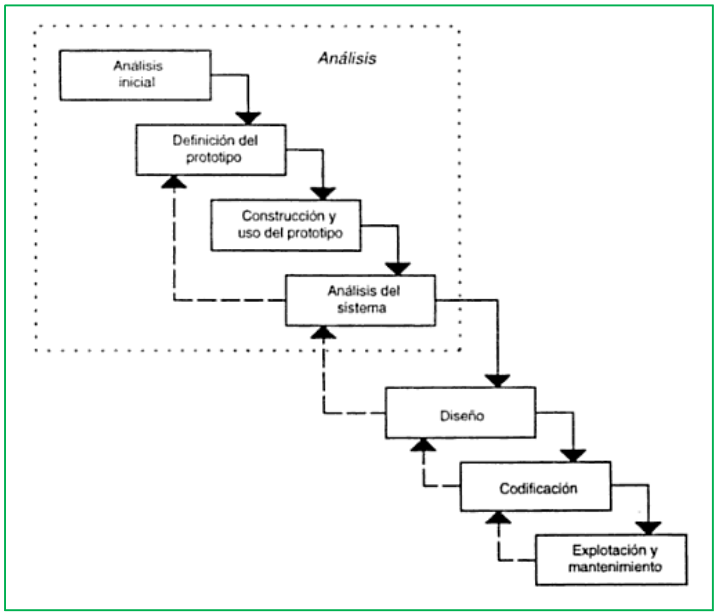

Ilustración 3. Utilización de prototipo rápido dentro de un ciclo de vida en cascada.

4.3.2. Prototipos evolutivos

Son prototipos que buscan aprovechar al máximo su código por lo que utilizan el mismo soporte de hardware y software del sistema final, pero sólo con una representación parcial de las fases de análisis y diseño. La experimentación con estos prototipos ayuda a avanzar en fases

parciales y su continuación ayuda a que se convierta en el sistema final tras la adición de elementos sucesivos que van surgiendo, corrigiéndose y validándose. En estos prototipos se construyen varias versiones de los mismos hasta obtener el deseado y los documentos relacionads (espeficación, diseño, etc.) también se van completando sucesivamente.

Por colocar un ejemplo de propuestas de solución, Cerrada (1997) propone un gráfico de ciclo de vida en cascada donde se indica una solución a uno de los aspectos más criticados del ciclo de vida en cascada y es la presunción de que cada etapa se realiza de forma independiente a las demás, y que la finalización de una es requerida para iniciar la siguiente:

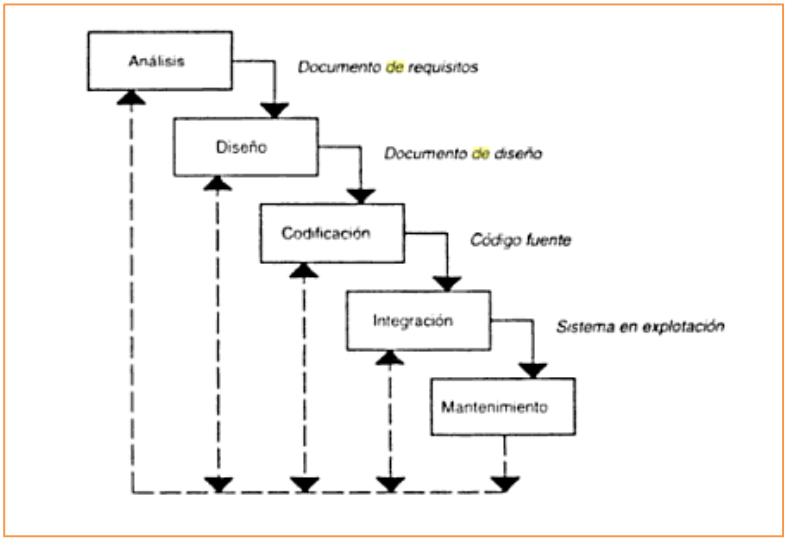

Ilustración 4. Propuesta de ciclo de vida en cascada de un producto

4.4. Modelo en Espiral del Ciclo de Vida:

Es un modelo que contempla un ciclo repetitivo donde cuatro etapas, como se muestran a continuación, se desarrollan progresivamente, cada vez con más y mejor información de la misma:

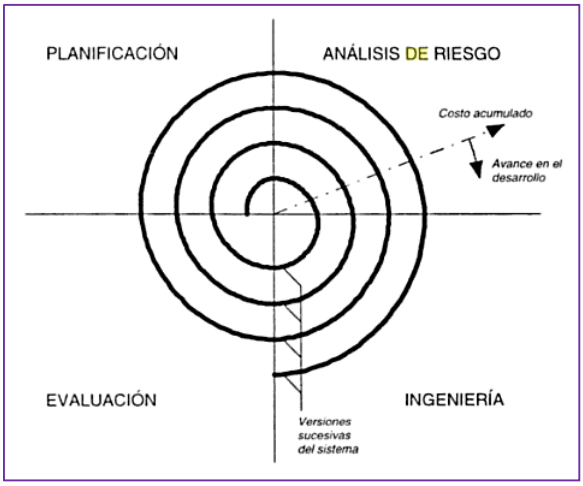

Ilustración 5. Modelo en Espiral del Ciclo de Vida. Cerrada (1997)

Como se observa, se desarrollan cuatro etapas en particular (Cerrada, 1997):

- *Planificación:* Establecen el contexto del desarrollo y deciden en qué parte del mismo se abordará determinado ciclo de la espiral.

- *Análisis de riesgo:* Evalúan las diferentes alternativas para la realización de la parte de desarrollo elegida, donde se selecciona la más ventajosa y se toman precauciones para evitar los inconvenientes previstos en el proyecto.

- *Actividades de ingeniería:* Son todas aquellas de los modelos clásicos de ciclo de vida de desarrollo (análisis, diseño, codificación, etc.) donde en cada versión del ciclo se obtiene un resultado mucho más complejo que ha considerado todos los posibles problemas, cambios de especificaciones y otros elementos que se han ido estableciendo en momentos previos del ciclo.

- *Actividades de evaluación:* Se analizan los resultados de la etapa anterior, generalmente en colaboración con el cliente para el que se realiza el desarrollo de software. Estos resultados permiten utilizar la información como entrada para planificar el ciclo siguiente.

4.5. Metodología RUP

El RUP, abreviatura de *Rational Unified Process* o Proceso Unificado Racional es un proceso propietario de la ingeniería de software creado por la *Rational Software Corporation*, adquirida por IBM, ganando un nuevo nombre: IRUP, que ahora es una abreviatura de *IBM Rational Unified Process.* Proporciona técnicas que deben seguir los miembros del equipo de desarrollo de software con el fin de aumentar su productividad en el proceso de desarrollo (Krutchen, 2004).

RUP utiliza el enfoque orientado a objetos en su diseño y está diseñado y documentado mediante la notación UML –*Unified Modeling Language*– para ilustrar los procesos de trabajo. Utiliza técnicas y prácticas probadas comercialmente.

4.5.1. Directrices

La metodología RUP define las siguientes líneas maestras y plantillas para los miembros del personal de producción, y una evaluación de los avances del proyecto por su gestión. También ayuda a los programadores a mantenerse enfocados en el proyecto (Bergström & Råberg, 2004).

- *Requisitos de gestión*. La documentación apropiada es esencial para cualquier proyecto de gran envergadura; tenga en cuenta que RUP describe cómo documentar la funcionalidad, las limitaciones del sistema, restricciones de diseño y requerimientos del negocio. Los casos de uso y los escenarios son ejemplos de artefactos que dependen del proceso, se consideran muy eficaces en la captura de requisitos funcionales.

- *Uso de arquitectura basada en componentes*. La arquitectura basada en componentes crea un sistema que puede ser fácilmente extensible, promoviendo la reutilización de software y una comprensión intuitiva. Un componente por lo general se refiere a un objeto en la programación

orientada a objetos. El RUP proporciona una manera sistemática para construir este tipo de sistema, centrándose en la producción de una arquitectura ejecutable desde el principio en el proyecto, es decir, antes de comprometer recursos a gran escala.

- *Uso de software de modelos visuales.* Al abstraer la programación de su código y representarla mediante bloques gráficos de construcción, RUP provee una manera efectiva de obtener una visión general de la solución. El uso de modelos visuales también permite a individuos con menos perfil técnico (*i.e.* clientes) tener una mejor comprensión de un problema dado, y así participar más activamente en el proyecto como un todo.

- *Verificación de calidad del software.* No garantizar la calidad del *software* es el fallo más común en todos los proyectos de sistemas informáticos. Por lo general se piensa en la calidad del software después de la finalización de los proyectos, o la calidad es responsabilidad de un equipo de desarrollo de equipo diferente. La metodología RUP tiene como objetivo ayudar en el control de la planificación de la calidad: verificándola en la construcción de todo el proceso e involucrando a todos los miembros del equipo de desarrollo.

- *Gestión y control de cambios.* En todos los proyectos de software la existencia del cambio es inevitable, RUP define métodos para controlarlos y monitorearlos. Puesto que un pequeño cambio puede afectar a las aplicaciones de formas totalmente impredecibles, su control es crítico para el éxito del proyecto; RUP también permite definir *áreas de trabajo seguras*, garantizando que modificaciones realizadas en otro sistema no afectarán el propio.

4.5.2. Fases

Hasta el momento, las anteriores son los lineamientos generales a seguir durante el ciclo de vida de un *software*. Pero para capturar la dimensión temporal de un proyecto, RUP lo divide en cuatro etapas diferentes (Krutchen, 2004).

Iniciación. La fase de diseño o de iniciación contiene los flujos de trabajo requeridos para el acuerdo de las partes interesadas con los objetivos, la arquitectura y la planificación del proyecto. Si estos actores tienen un buen conocimiento, se requiere poco análisis. De lo contrario, se requerirá una prueba más elaborada. En esta etapa, los requisitos esenciales del sistema se transforman en *casos de uso*. Esta fase es generalmente corta y sirve para definir si es factible continuar con el proyecto y definir los riesgos y el coste del mismo. Se puede hacer un prototipo para que el cliente lo pre-apruebe.

Elaboración. La fase de elaboración será sólo para el diseño del sistema, buscando complementar la documentación de los casos de uso, apuntando a la arquitectura del sistema y la revisión del modelo de negocios. Conjuntamente, se inicia la versión del manual del usuario

Construcción. Durante la fase de construcción, el desarrollo físico de software comienza: códigos de producción y pruebas alfa; las pruebas beta se realizarán a principios de la fase de transición.

Transición. En esta etapa se produce la entrega del software, se lleva a cabo el plan de despliegue y entrega, y la verificación de calidad del software. Esta etapa también se lleva a cabo la formación de los usuarios.

En la siguiente figura, se indica el énfasis que se le da en el proyecto a cada una de las fases en un instante dado:

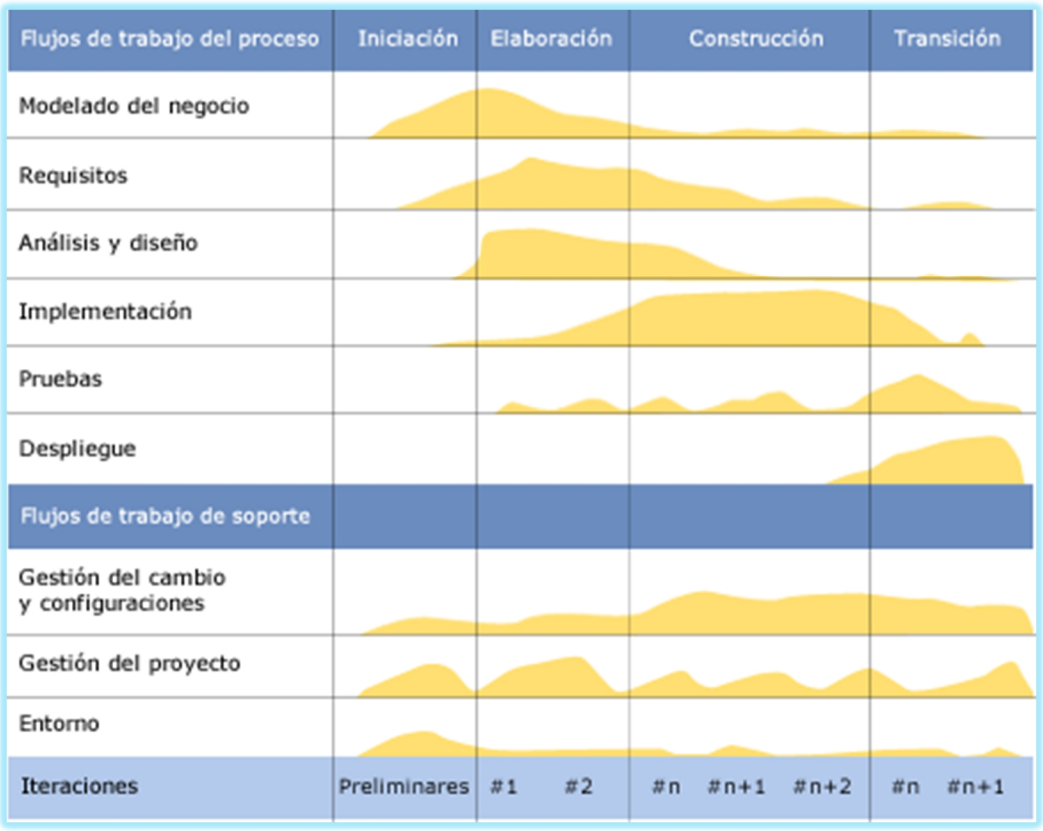

Ilustración 6. Esfuerzo en actividades según fase del proyecto
Fuente: IBM, http://www-01.ibm.com/software/br/rational/

5. MARCO CONCEPTUAL

Este capítulo da a conocer (tomando como referencia el siguiente mapa conceptual) los conceptos necesarios para la solución de problemas a través de fases de programación. Se establece la secuencia de órdenes sistemáticamente, por ejemplo no puede pintarse una pared antes de levantarla; no puede después de terminar una casa que fue planeada para hacer de un piso, convertirla en un edificio, el proceso de crear un programa no es la excepción.

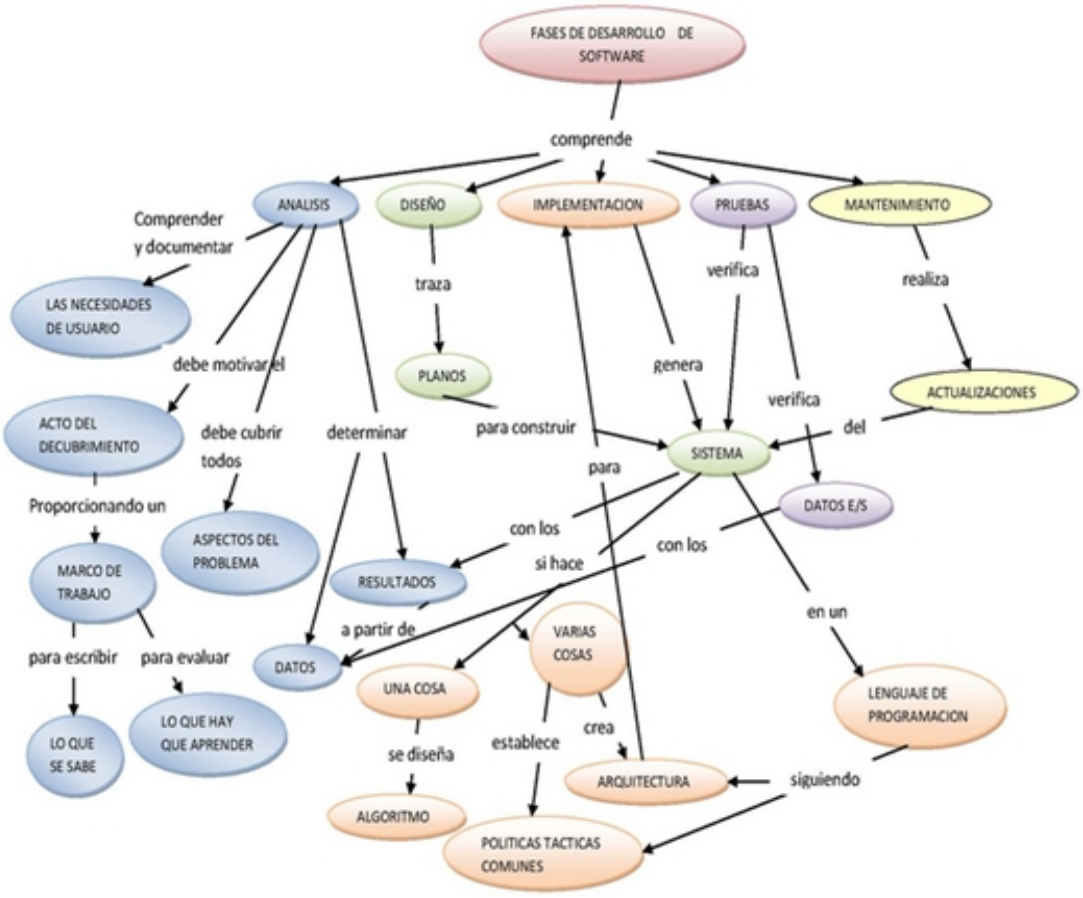

Ilustración 7. Mapa conceptual desarrollo de software
Fuente: Rawson-Chubut

En múltiple textos se han expuesto las etapas de desarrollo de software, siguiendo como referencia a Grady Booch, desarrollador del UML, es posible estructurar la solución de problemas mediante *software* como divisible en las siguientes fases. (Booch, 1996)

5.1. Análisis

Lo primero que hay que hacer para estructurar un sistema informático es definir, con precisión, lo que debe que hacer tal sistema. Esta fase de análisis se refiere al proceso por el cual se descubre qué es lo se necesita exactamente y se llega a comprender los requerimientos que el sistema debe tener. La fase de análisis es esencial, sencillamente porque si no se sabe con puntualidad qué es lo que se necesita, ninguna secuencia de desarrollo permitirá obtenerlo. De ahí, la dificultad es que probablemente ni el mismo cliente sepa lo que requiere. Por tanto, debe ayudársele a encontrarlo.

De hecho, el costo de estructurar de forma correcta un sistema informático, desde el comienzo, es mucho menor que el costo de estructurar otro sistema que habrá que transformar más adelante. Cuanto antes se detecte un error, mejor. Distintos estudios han demostrado que eliminar un error en las fases iniciales de un proyecto (en la etapa de análisis) resulta de 10 a 100 veces más económico que subsanarlo al final del proyecto. Conforme avanza el proyecto, el software se va describiendo con un mayor nivel de detalle, se concreta cada vez más y se convierte en algo cada vez más rígido.

Lamentablemente, no es posible determinar de antemano todos los requerimientos de un sistema de información (Berszal, 2014, pág. 10). En efecto, la primera causa de fallo en el desarrollo es la inestabilidad de los requerimientos del sistema –la segunda es una incorrecta proyección del esfuerzo requerido–. Para la segunda, el problema puede solucionarse fijando

objetivos más realistas; no obstante, para la primera causa no se dispone, al inicio, de la información suficiente para determinar con precisión el problema a resolver. Por más tiempo que se invierta en la fase análisis –fenómeno conocido como la parálisis del análisis– la inestabilidad de los requerimientos es inevitable: Se estima que un 25% de los requerimientos iniciales de un sistema cambian antes de que el sistema comience a usarse.

Para dar solución a estos conflictos, existen técnica de gestión de requerimientos, útiles también para dirigir su evolución: elicitación de requerimientos, modelado de sistemas y la metodologías de análisis (Berszal, 2014, pág. 11).

5.2. Diseño

En tanto, que las estrategias empleadas en la anterior fase de análisis representan los requisitos del usuario –el qué–, los modelos de la fase de diseño contienen las propiedades del sistema que permitirán ejecutarlo efectivamente –el cómo–.

Un *software* correctamente diseñado exhibe propiedades particulares: su diseño es modular en vez de monolítico; sus módulos son cohesivos –se encargan de una *task* concreta y sólo de una– y están ligeramente ensamblados entre sí para simplificar su mantenimiento–. Además, en un *software* con diseño eficiente, cada módulo ofrece a los restantes unas interfaces bien definidas (Meyer, 2009, págs. 48-49), y ocultar sus detalles de implementación –conforme el principio de ocultación de Parnas–.

En la etapa de diseño se contrastan las probables alternativas de implementación para el sistema a crear y se decide su diseño arquitectónico. El diseño de un sistema es complejo y el proceso de diseño ha de realizarse de forma iterativa.

5.3. Implementación

A partir del análisis y diseño de la solución, en la fase de implementación se desarrollar las bases de datos e interfaces Web correspondientes, que solucionen el problema analizado mediante el uso de compiladores específicos.

5.4. Pruebas

Los errores lógicos potenciales, en el ambiente de la programación, son muchos y aumentan exponencialmente con la complejidad del problema. Cuando se termina de implementar un algoritmo, es obligado realizar las debidas pruebas que garanticen el buen funcionamiento del *software* bajo el mayor número de situaciones posibles a las que se pueda enfrentar (Villegas & Eduado, 2007, pág. 9).

5.5. Mantenimiento

Los programas informáticos del mundo real necesitan evolucionar a través del tiempo con para no tornarse obsoletos e ineficientes. Por ello es necesario actualizarlos cada cierto tiempo. Esta es la última etapa de vida del software donde se explota el software como tal, ya que siempre que se utilice habrá que mantenerlo, es decir, realizar cambios (grandes o pequeños) para corregir errores, mejorar las funciones del programa o su eficiencia o adaptarlo a un nuevo hardware o realizar cambios en las necesidades de información. Se considera que un software está en explotación aproximadamente 10 años o más, por lo que se considera que el costo de mantenimiento durante la vida del software puede ser de dos a cinco veces mayor que el desarrollo.

Algunos mantenimiento de software pueden llevarse a cabo con las herramientas CASE (*Computer Aided Software Engineering*), que básicamente son otros software de apoyo al desarrollo, mantenimiento y documentación informatizada de software. Estas herramientas, por lo tanto, más que generar tratamientos de texto, hojas de cálculo, planificaciones o dibujos, se utilizan principalmente para codificar el software (compiladores, entornos de 4ta generación, editores de programas). Estas herramientas pueden ser (Campderrich Falgueras, 2003):

- Herramientas diagramáticas que, a diferencia de las de dibujo, reconocen cada símbolo como una clase. Estas herramientas aceptan documentación textual de algunos elementos que utilizan. Estas herramientas reciben el nombre de *UpperCase.*
- Herramientas de gestión de la prueba y gestión de calidad en general.
- Herramientas de gestión de cambios. Estas herramientas, junto a las anteriores, reciben el nombre de *LowerCase.*

Las modificaciones realizadas en esta etapa ayudan a generar el Documento de Cambios, que recopilan información de cada problema detectado, la solución adoptada y las modificaciones realizadas al sistema para aplicar la solución (Campderrich Falgueras, 2003).

6. MARCO HISTÓRICO

Desde hace ya varios años, el autor de este proyecto, se dedica a la informática: bases de datos, desarrollo de *software* y aplicaciones en línea. Es una ciencia innovadora, apasionante e inquietante, pero a su vez es difícil. Sin duda, es una de las ciencias dónde más intrusismo existe, debido a lo accesible que es para cualquier individuo, y ello ejerce presión sobre los profesionales. De ahí, que en la informática nunca se interrumpe el aprendizaje nunca se sabe lo suficiente. Continuamente, el autor ha trabajado para convertir lo que comenzó como su hobby para convertirlo en el trabajo de su vida.

Han sido múltiples los lenguajes y bases de datos usados para el desarrollo de aplicaciones: PHP y MySQL, Java-Mysql, Java-Oracle, Visual Basic y SQL Server, C#, .NET-SQL. Pero ninguna de estas tecnologías parecía le definitiva, hasta hace poco, se empleó Oracle Application Express. La versión actual de *Oracle Application Express* se ha mejorado mucho y es un producto maduro, se nos proporciona un Entorno de desarrollo para diseñar y desarrollar aplicaciones web para la Base de Datos Oracle.

En efecto, si se tiene en cuenta que las aplicaciones Web se están dominando el mercado, gracias a la difusión de los dispositivos móviles y que la Base de Datos ORACLE es considerada la mejor del momento, puede decirse que Oracle Application Express (Apex a partir de este momento) es una tecnología interesante y con grandes argumentos para ser usada. Incluso ORACLE está desarrollando sus aplicaciones en Apex, de hecho el servicio Cloud de ORACLE está desarrollado con Apex. Como todo en la vida, Apex tiene cosas buenas y cosas malas, a día de hoy veo muchas más cosas buenas que malas. Las cosas buenas serán vistas a lo largo de este proyecto, en cuento a las cosas malas, quizás la más significativa es que Apex es una tecnología

propietaria de Oracle, a diferencia de otros lenguajes como PHP que son Open Source, pero Apex al día de hoy es gratuito y Oracle proporciona una versión Express de su Base de Datos, por lo que no tendremos que pagar una licencia para desarrollar aplicaciones mediante Ápex.

6.1. Breve Historia

En 2003, se publica una herramienta denominada HTML DB 1.5, este producto era gratuito y funcionaba sobre Oracle 10 R1. A partir de ese momento, han ido saliendo diferentes versiones que han ido introduciendo mejoras y funcionalidades. Como se muestra a continuación una lista con la evolución de Apex en los últimos años.

- ✓ HTML DB 1.6 (2004). Introdujo temas, formularios maestro-detalles, grupos de páginas, bloque de páginas y algunas capacidades multilingües.
- ✓ HTML DB DB 2.0 (2005). Introdujo SQL Workshop, el generador gráfico de consultas, objetos basados en el navegador para la base de datos y la protección por estado de sesión.
- ✓ APEX 2.2 (2006). Introdujo las aplicaciones empaquetadas, las vistas de diccionario de APEX, y los controles de asistentes.
- ✓ APEX 3.0 (2007). Introdujo la impresión PDF con la herramienta BI Publisher, la migración de Microsoft Access, y la página de almacenamiento en caché y el concepto de región.
- ✓ APEX 3.1 (2008). Introdujo los informes iterativos, mejoras en tiempo de ejecución y de seguridad.
- ✓ APEX 3.2 (2009). Esta versión presentó un asistente para la migración de sistemas basados en Oracle Forms y varias mejoras en seguridad.

- ✓ APEX 4.0 (2010). Introdujo grandes novedades, como la implantación de acciones dinámicas y plugins para la lógica de desarrollo y ampliaciones en el entorno de Apex. También se introdujo el módulo de Desarrollo de Equipos.
- ✓ APEX 4.1 (2011). Incluye nuevas funciones para la carga de datos, mayor capacidad de control de errores y mejor soporte en los formularios tabulares.

6.2. Presente y futuro de APEX

La versión actual de Apex en el periodo de desarrollo de este trabajo es la 4.2, para ser más exacto la 4.2.2.00.11. En esta versión de Apex se han introducido grandes novedades y características, que convierten a Apex en una gran plataforma para el desarrollo de aplicaciones basadas en Oracle. Recientemente y a fecha 16 de septiembre de 2013, Oracle ha lanzado una nueva versión de Apex, la 4.2.3. Con la versión de Apex usada se pueden crear acciones dinámicas que definen comportamientos en el lado del cliente. Con conocimientos sobre javaScript y Ajax se pueden crear acciones dinámicas que realizan cálculos y acciones complejas.

Esta versión incorpora un motor de gráficos mejorado basado en la última versión del AnyChart que no sólo proporciona declarativa basada en gráficos Flash, indicadores, mapas y diagramas de Gantt, sino que también se permite crear gráficos basados en HTML5 que funcionan en cualquier plataforma y sin necesidad de añadir ningún plugins al navegador. Otra característica interesante es la arquitectura de plugins, proporcionan extensiones que permiten a los miembros de la comunidad crear y compartir sus propios plugins.

Incluye también la función de desarrollo de equipos, que facilita la gestión del proceso de desarrollo. Incluyendo funciones de seguimiento, listas de tareas pendientes, control de errores y las etapas de desarrollo.

Otra característica incluida es la posibilidad de crear hojas de cálculo web, que proporcionan a los usuarios finales una manera rápida y directa para reunir y compartir información. Probablemente la característica más importante que incluye Apex 4.2 es la capacidad de crear Aplicaciones destinadas a dispositivo móviles. APEX incorpora jQuery Mobile para representar el contenido en la gran mayoría de los dispositivos móviles. Con esta versión de Apex podemos además crear tanto aplicaciones en versiones de escritorio como interfaces para dispositivo móviles. El tema incluido para aplicaciones móviles permite que estás se ajusten a las dimensiones de la pantalla. Con la misma interfaz se puede trabajar desde un PC, tablet y dispositivo móvil.

Como conclusión se puede decir que la funcionalidad de Apex sigue creciendo y aportando novedades en cada nueva versión, que estará para lanzamiento la versión 5.0 y está prevista ente finales del mes de septiembre y comienzos de octubre del 2014 por lo que se puede afirmar, aún más, que Apex es una apuesta con futuro.

7. MARCO LEGAL

Toda organización social posee un andamiaje jurídico que regula los derechos y deberes, en las relaciones entre sus diferentes miembros. Este contexto jurídico e institucional parte desde la Constitución, la ley, los decretos, las ordenanzas y los acuerdos, hasta los reglamentos y las resoluciones, y se expresa en forma prohibitiva o permisiva. Teniendo como prelación lo descrito anteriormente y siendo consecuentes con lo descrito en este proyecto, esta sección tiene la relevancia de dejar claridad con el uso de los sistemas de información que serán instalado en la fundación, y que partiendo del objeto social que ella trabaja.

Entonces es importante declarar los siguientes conceptos como es el tema de Software Libre y los limitantes que se tendrán para el proyecto y que específicamente se orienta para los sistemas operativos y a la Base de datos que para nuestro caso es ORACLE.11g correrá el software que se implemente que para nuestro caso tendrá como denominación J_ROHI.

Para empezar a hablar sobre el software libre es necesario comenzar a mirar el régimen jurídico que rodea la protección de los programas de computador, y ello necesariamente nos transporta a las discusiones que en los años 80 buscaron determinar la forma legal más adecuada para la protección del soporte lógico. Fue durante esta década que el concierto global se reunió a debatir las alternativas de protección. Se planteó entonces la posibilidad de establecer un régimen sui generis o especial, o de pronto buscar su protección a través de la propiedad industrial vía el régimen de patentes o el de los modelos de utilidad y se pensó también, en el secreto empresarial. Sin embargo, no solo las orientaciones dadas por la Organización Mundial de la Propiedad Intelectual, sino todas las leyes nacionales establecieron como una alternativa efectiva de protección las normas del derecho de autor.

De esta manera, y entendiendo el programa de ordenador como aquella secuencia ordenada de instrucciones destinadas a ser asimiladas por un computador, a fin de lograr un resultado específico, la mayoría de legislaciones lo han equiparado a una obra literaria dando protección al proceso de orden intelectual que precede la elaboración de un código fuente. Efectivamente, la creación de un programa de ordenador implica en primera instancia la generación de un algoritmo el cual deberá transformarse en un conjunto de instrucciones en lenguaje de programación, hechas y entendibles por el ser humano, el cual denominamos código fuente. Sin embargo como este lenguaje no puede ser ejecutado por un computador, es necesario que un programa compilador lo traduzca en un lenguaje entendible para la maquina (un código binario), cuyo producto conocemos como el código objeto o código ejecutable.

7.1. Alcance de Protección

La protección se concede de manera integral a ambas formas de expresión del programa de computador, otorgándosele a su autor (al programador) las prerrogativas morales y patrimoniales que el derecho de autor concede desde el momento mismo en que se crea la obra, para el caso el software. Bajo este concepto, se procede a desarrollar el tema del software libre, es por eso que la aplicación J_ROHI, se implementa para uso exclusivo de la Fundación ELENA Y JUAN, y su filiares que tengan la misma actividad social.

7.2. Especificaciones Técnicas de Hardware y Software

A continuación los requerimientos mínimos para correcto funcionamiento de la aplicación.

Equipo de Computo	*Características*	*Tipo de Licencia*
▪ *Servidor Hp, Dell,* ▪ *Clon*	*Intel Dual Core, o Amd, Sempro, 2 RAM, DD 320 Gigas*	*Windows 7 Home, Profesional, 2008, 2008 Estándar Server o Superior*
▪ *Base de Datos OracleXE 11g.*	*Base de datos*	*Versión de base de datos de libre distribución, el cual puede crecer hasta 11 gigas,* ***Sin*** *compra de servicios o contrato de sostenimiento*
▪ *Herramienta de desarrollo Oracle Application Express*	*APEX*	*Versión libre, Distribución de ORACLE.*
▪ *Servidor de Aplicaciones GLASSFISH*	*Servidor de Aplicaciones*	*Es un servidor de aplicaciones de software libre desarrollado por Sun Microsystems, compañía adquirida por Oracle Corporation, que implementa las tecnologías definidas en la plataforma Java EE y permite ejecutar aplicaciones que siguen esta especificación. Es gratuito, de código libre y se distribuye bajo un licenciamiento dual a través de la licencia CDDL y la GNU GPL.*

Tabla 1. Especificaciones de *hardware* y *software*

8. DISEÑO METODOLÓGICO

8.1. Fase I – Iniciación

Durante esta fase, se define toda la logística en general del proyecto, se establece el alcance detallado, los riesgos de proyecto, las políticas, etc. Se formulará un cronograma de proyecto preliminar. El proyecto se inicia oficialmente con la reunión de arranque (Reunión de Kick-Off).

A esta reunión asistirá todo el equipo del proyecto El objetivo es realizar labores de *TEAM BUILDING*, divulgación de políticas que aplicarán, resaltar la importancia del proyecto para las metas globales de la nueva aplicación o software a crear y la necesidad de comprometer al director de la corporación en la importancia del proyecto y los beneficios que traerá a ELENA Y JUAN. (Ver Ilustración 15).

Nombre	***Rol***
María Cristina Calle Calle	*Directora Corporación Elena y Juan*
Edisson Javier Oquendo M	*Ingeniero de Sistemas FUSM*
Ana Cristina Gómez Duque	*Ejecutiva de Proyectos Fundación*
Luz Marina Valencia Hoyos	*Asistente de Dirección*

Tabla 2. Equipo de trabajo del proyecto

Ilustración 8. Fase I

a) Establecimiento del Alcance del Sistema
b) Estudio de la situación Actual
c) Definición de requisitos del sistema.
d) Estudio de alternativas de solución.
e) Valoración de alternativas
f) Selección de la solución.

8.2.Fase II – Elaboración

El propósito de la fase *Business Blueprint* es entender los objetivos de negocio de la fundación y determinar los procesos requeridos para apoyar tales objetivos. Se realizaron reuniones para discutir los objetivos, la estructura organizacional y los procesos de negocio de alto nivel. Para discutir cada proceso, se realizaron sesiones de trabajo específicas con el objeto

de obtener requerimientos más detallados y definir todo el plan de levantamiento de datos maestros. (Ver Ilustración 16)

Ilustración 9. Fase II

Actividades que se realizaron en esta fase del proyecto:

a) Definición del sistema.
b) Establecimiento de Requisitos
c) Identificación de subsistemas de análisis
d) Análisis de los casos de uso
e) Análisis de Clases.
f) Elaboración de un Modelo de Datos.
g) Elaboración del Modelo de procesos.
h) Definición de las interfaces de usuario a crear
i) Especificación de plan de Pruebas
j) Aprobación del análisis del sistema de información.

8.3. Fase III – Construcción

Esta fase corresponde a la parametrización total del sistema, de acuerdo al alcance detallado del *Business Blueprint* en esta sección el Ingeniero de la FUSM define los ciclos de parametrización y desarrollo por escenarios y a medida que van parametrizando comparten con el usuario líder el resultado, de tal forma que se vayan realizando las pruebas individuales. Con base en esas pruebas se elaborarán los manuales de usuario final, además paralelamente, los usuarios líderes desarrollan el plan de levantamiento de datos maestros.

Se diseñó entre todo el equipo el plan de pruebas integrales y se llevará a cabo con la participación activa de toda la organización en el proceso de validar y asegurarse de que se están cumpliendo apropiadamente los requerimientos de la organización a través de todos los procesos. (Ver Ilustración 17)

Ilustración 10. Fase III

A continuación se detallan las actividades efectuadas:

a) Definición de la arquitectura del sistema
b) Diseño de la arquitectura de soporte
c) Diseño de los casos de uso reales.
d) Diseño de clases
e) Diseño de la arquitectura de módulos del sistema
f) Diseño físico de los datos en ORACLE
g) Verificación y aceptación de la arquitectura del sistema
h) Generación de especificaciones de construcción.

8.3.1. Preparación para producción

El propósito principal de esta etapa completar las pruebas finales del sistema, entrenar a los usuarios finales y preparar el sistema y los datos para el ambiente productivo. Las pruebas finales del sistema consisten en pruebas de volumen y de carga, se cargan los datos maestros y se espera aprobación de la dirección para el arranque. (Ver Ilustración 18).

Ilustración 11. Fase III Preparación Para Producción

Para esta fase del proyecto se definen las siguientes actividades:

a) Preparación del entorno de generación y construcción
b) Generación de código fuente PL/SQL de los componentes y procedimientos
c) Ejecución de Pruebas Unitarias
d) Ejecución de Pruebas de Integración
e) Ejecución de Pruebas del Sistema
f) Elaboración de Manuales de Usuario
g) Definición de la Formación a los usuarios finales.
h) Construcción de procedimientos de migración y carga inicial de datos.

8.4. Fase IV – Transición

En esta fase las actividades que se ejecutan son:

- Se realiza el acompañamiento por parte del Ingeniero de la FUSM en las primeras labores y en el primer registro en el sistema, realizando afinamientos que fueran necesarios.
- Elaboración de Actas de Cierre como entregables de la misma. (Ver Ilustración 19).
- Realizar pruebas en el servidor de producción para verificar el funcionamiento adecuado de la aplicación J-ROHI, base de datos Oracle, y servidor de aplicación Glassfish.
- Realizar la salida a producción un día de bajo flujo de documentación, de tal manera que el proceso de adaptación afecte lo menos posible al usuario final.
- Divulgar el nuevo proceso a todos los actores del sistema, y garantizar que tengan claridad sobre el papel que cumplen y el beneficio que este cambio reporta a la organización.
- Definir un plan de contingencia en caso que el sistema no esté disponible.

Durante la estructuración, el equipo de trabajo describe las siguientes actividades:

- **Planificación del proyecto:** Plan de actividades a seguir utilizando un software de seguimientos de proyectos.
- **Plan de control de calidad**: Representa las acciones a realizar para que el producto final resultante cubra y garantice las expectativas de diseño, desarrollo, seguridad, eficiencia, entre otras.
- **Organización del equipo de trabajo**: Se especificaron las funciones y responsabilidades de cada uno de los integrantes del equipo experto.
- **Análisis del sistema**: Se analizó la funcionalidad del producto final siguiendo un formato estándar de control para la determinación de los requerimientos del producto.
- **Plan de pruebas de aceptación:** Permite generar un producto de software libre de errores y condiciones no cubiertas en el diseño del producto final.
- **Diseño del producto:** Especifica a detalle las condiciones de funcionalidad del producto final, incluyendo el diseño de las interfaces, programación y funcionalidad del producto, basándonos de nuevo en un formato estándar para la determinación del diseño del producto.
- **Pruebas efectuadas:** Indica las pruebas de eficiencia y funcionalidad del sistema.
- **Implantación del sistema:** Indica el proceso a seguir para la implantación del sistema final.
- **Garantía y mantenimiento:** Indica las condiciones de operación y funcionamiento del producto final, en las que se detallan las situaciones bajo las cuales el producto cubre de garantías.
- **Conclusiones:** Indican los resultados finales obtenidos en el proceso de desarrollo de la aplicación J_ROHI

Ilustración 12 . Fase IV

9. ALCANCE

Como se describe en el cronograma de actividades el sistema integrado J_ROHI contara con las siguientes actividades y opciones que se le presentaran al usuario Final:

9.1.Creación de Ambientes de Trabajo

Se crearan tres ambientes de trabajo los cuales tendrán como objetivo un seguimiento de las configuraciones, desarrollos que se le efectúen al sistema y así contar con el compromiso por parte del grupo de trabajo de los entregables que se harán durante todo el proyecto. Estos ambientes se catalogaron de la siguiente manera.

a) DEV = Desarrollo.

b) QAS = Ambiente de Pruebas.

c) PROD = Ambiente productivo.

9.2.Actividades de Instalación de Productos

Instalación de dase de datos ORACLE® y APEX® en WINDOWS 2008®.

- Configuración de la red.
- Configuración Gateway PL/SQL.
- *Workspace*, espacio de trabajo de APEX.
- Configuración de Roles de usuario APEX.
- Instalación de Glassfish: servidor de aplicaciones.
- Configuración de Listener de Oracle Application Express.
-

9.3.Creación del Módulo de Parámetros

- Creación de tabla de usuarios
- Creación de Vista de Usuarios
- Elaboración de Pantalla de Acceso al sistema.
- Elaboración de control de acceso y esquema de roles y permisos.

Ilustración 13. Interfaz de acceso al Sistema

9.4.Módulo de Agenda: Elena y Juan

Su alcance es que la Fundación sistematice y programe sus actividades.

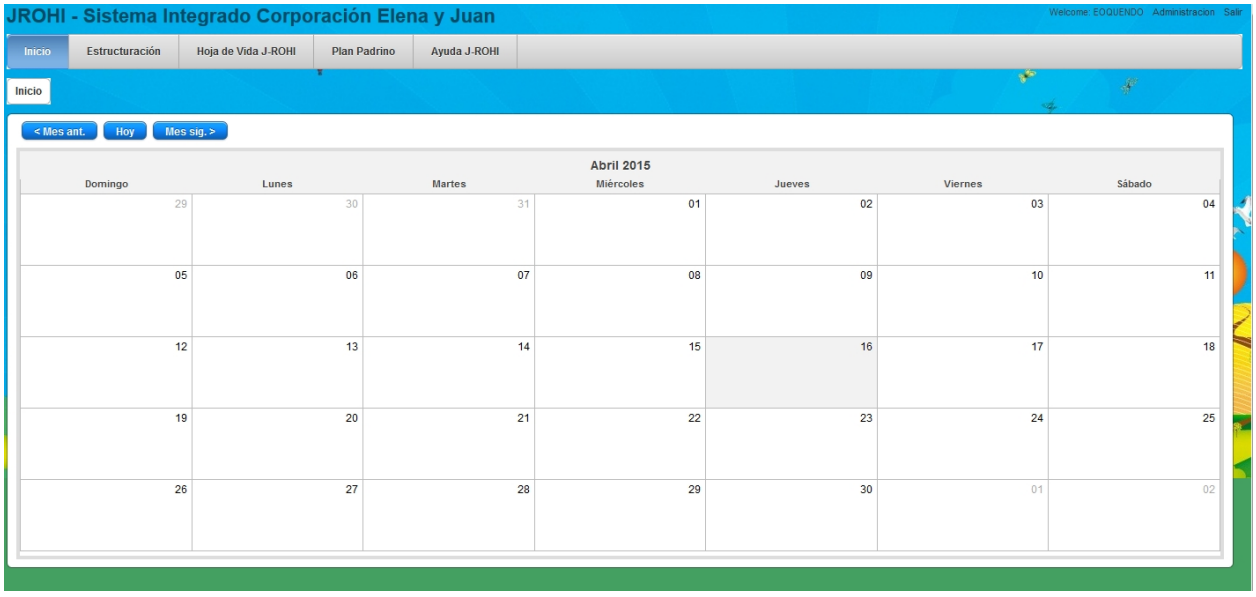

Ilustración 14. Diseño de pantalla de Agenda y Juan

9.5.Creación de Modulo de Estructuración:

El módulo de Estructuración contará con siguientes opciones de configuración:

E1. Catálogo de Empresas

E2. Centros Operativos

E3. Países/Departamentos/Ciudades

E4. Tipo de Documento de Identidad

E5. Tipos de Documentos Transacciones.

E6. Tipos de Terceros.

E7. Maestro de Donaciones.

E8. Clase Documental.

E9. Agenda Elena y Juan.

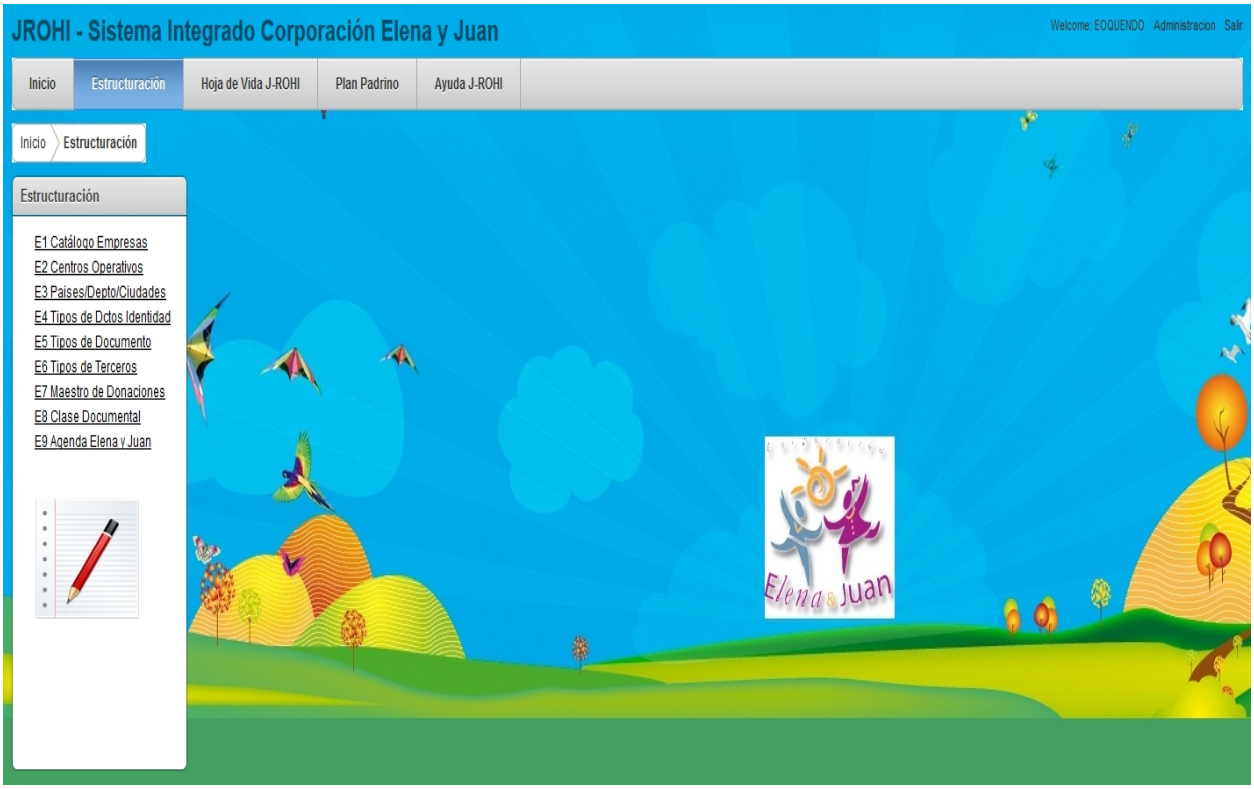

Ilustración 15. Diseño de pantalla del módulo de Estructuración

9.6.Creación de Modulo de Hojas de Vida:

Para este módulo se considera las siguientes opciones del sistema:

a) Registro de hoja de vida infante.

b) Bitácora de Registro de Movimientos de Salud, Nutrición, y Eventos pedagógicos básicos que son prestados a los infantes de la Fundación .

c) Gestión Documental: el alcance de esta opción es almacenar en la base de dato, todos los documentos que hacen referencia al infante, por ejemplo, registro civil, afiliación a salud, visitas médicas, seguimientos pedagógicos, entre otros. Permitiendo el control de la información que hoy se almacena en carpetas.

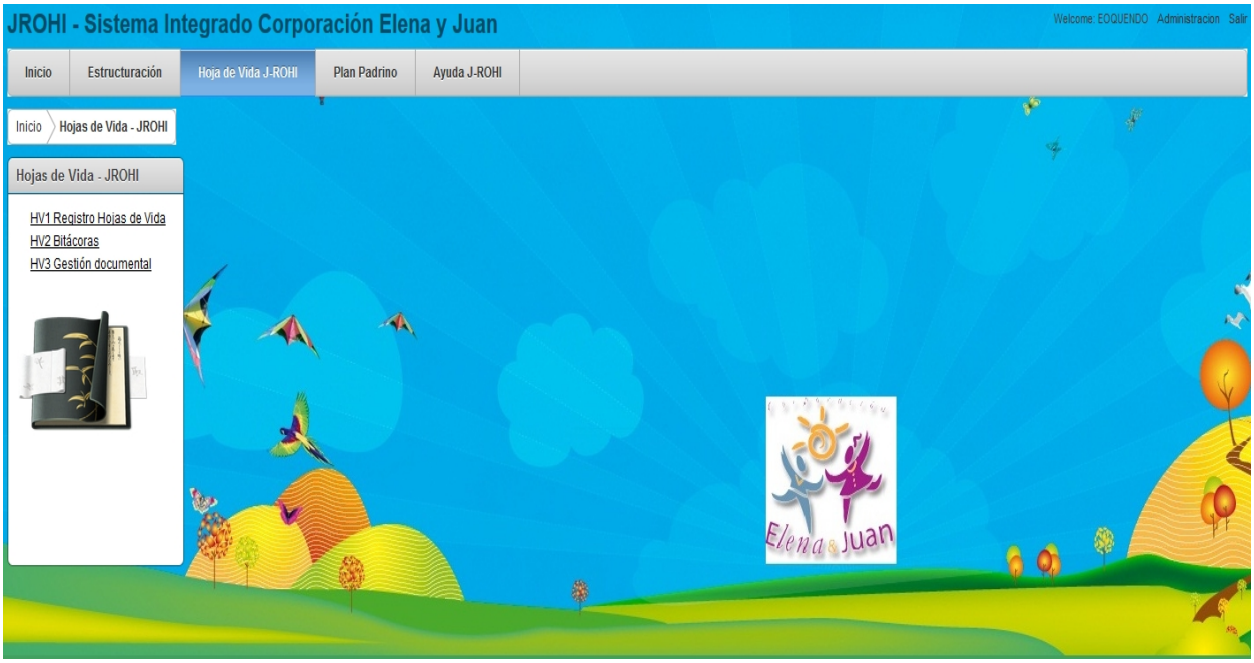

Ilustración 16. Diseño de pantalla de módulo de hojas de vida.

9.7. Módulo de Plan Padrino

El alcance de esta opción es registrar todos los movimientos correspondientes al plan padrino con el que cuenta la fundación, en sus diferentes formas de registrar sus ayudas que recibe de instituciones privadas, individuos en forma voluntaria, entre otras, formas.

Ilustración 17. Diseño de pantalla del Plan Padrino.

9.8.Modulo Ayuda J-ROHI

En esta opción se tiene considerada una opción de consulta para el usuario final en donde podrá realizar las consultas necesarias, en cuanto al manejo del sistema, la presentación actual del sistema cuenta con las siguientes opciones:

- **H1**. *Proyecto de Grado*: en este documento se consignan las especificaciones de cómo está realizado el trabajo, orientado a un proceso metodológico usado en la implementación y desarrollo de todo el proyecto J-ROHI.

- **H2**. *Manual Técnico*: en este documento se consignan las especificaciones técnicas del sistema y su construcción.
- **H3**. *Manual de Usuario*: como su nombre lo indica es el manual de usuario, correspondiente al funcionamiento del sistema, el cual este tipo de publicaciones brinda las instrucciones necesarias para que un usuario final pueda utilizarlo.

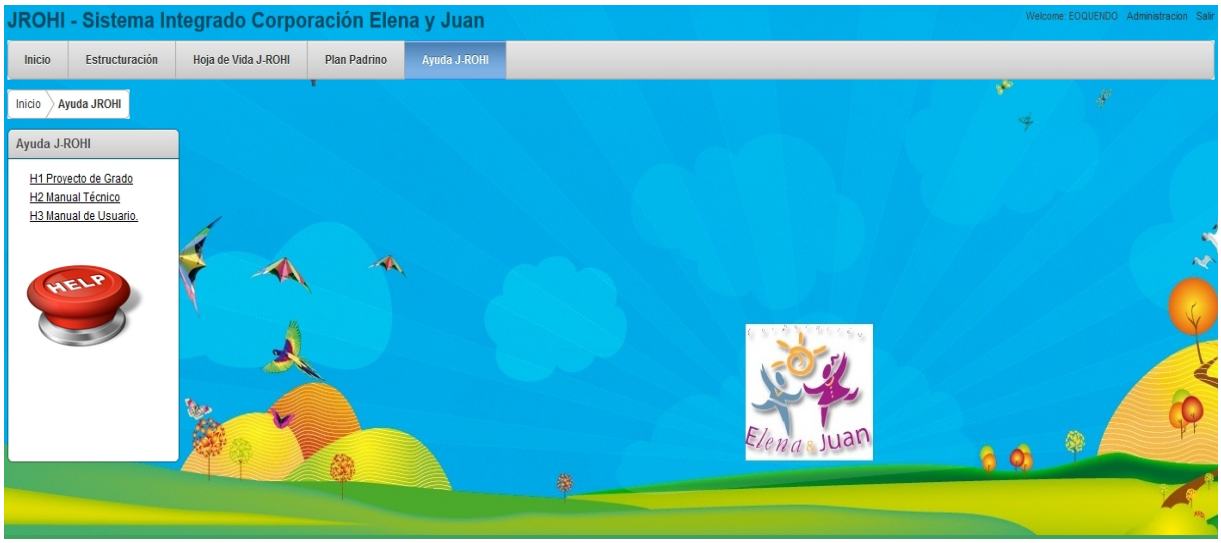

Ilustración 18. Diseño de pantalla de ayuda JROHI

9.9.Resumen del Alcance del Sistema

Ilustración 19. Resumen del Alcance del Sistema.

10. PRESUPUESTO

Tal como se ha descrito, el presupuesto se orienta con una propuesta inicial conociendo, de primera mano, que la Fundacion no tiene un rublo en sus estados financieros que le permitan destinar dineros en proyectos en tecnología y equipos de computo. Sabiendo que sus ingresos mensuales depende del apoyo de entidades privadas, profesionales e incluso voluntarios. Por tal razón se procede a realizar la solución de un sistema de información con herramientas que en el mercado son consideradas *Open Source,* y entre ellas están las herramientas que nos brinda ORACLE, con su base de datos Express, la cual tiene en su capacidad hasta 11 gigas de almacenamiento de datos y de aplicaciones. Además de su entorno de desarrollo que APEX en su versión 4.2.

Esta información se considera una ventaja para empresas donde sus recursos informáticos son escasos y teniendo como premisa que la versión actual de *Oracle Application Express* APEX se ha mejorado mucho y es un producto maduro, ya que nos proporciona un entorno de desarrollo para diseñar y desarrollar aplicaciones web para la Base de Datos Oracle. Si se tiene en cuenta que las aplicaciones web se están cada vez más implantando, gracias entre otras cosas a la difusión de los dispositivos móviles y que la Base de Datos Oracle es considerada la mejor del mercado, podemos decir que *Oracle Application Express* (Apex a partir de este momento) es una tecnología interesante y con grandes argumentos para ser usada.

10.1. Arquitectura de APEX

Apex utiliza un explorador web que se comunica con la Base de Datos Oracle mediante un Listener web. No se necesita ningún software cliente distinto del *browser* para el desarrollo,

despliegue o tiempo de ejecución de una aplicación. Las páginas de APEX se representan en el explorador utilizando HTML. Las solicitudes y ejecuciones de páginas se envían al motor de Apex en la Base de Datos Oracle (ver Ilustración 27).

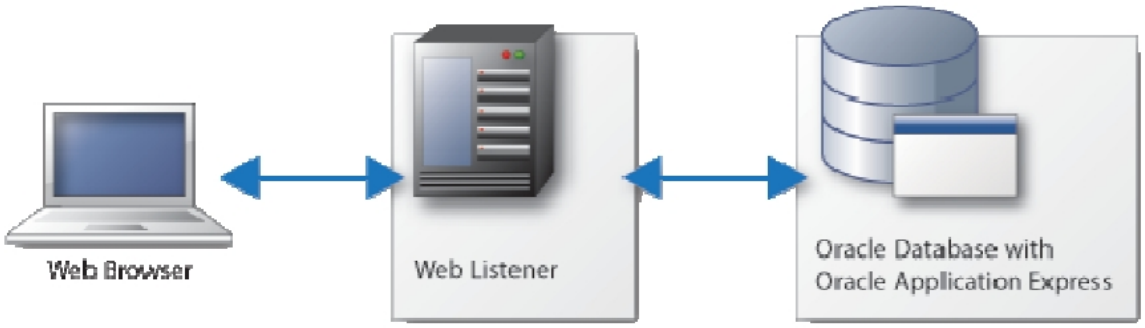

Ilustración 20. Arquitectura Básica de Ápex.
Fuente: Universidad de Almería

APEX es una función integrada de la ORACLE, consiste en un conjunto de más de 300 tablas y 200 objetos PL/SQL con más de 300.000 líneas de código. El motor de APEX ejecuto las páginas de forma dinámica en tiempo real a partir de los datos contenidos en el repositorio de metadatos (Naranjo, 2001, pág. 19).

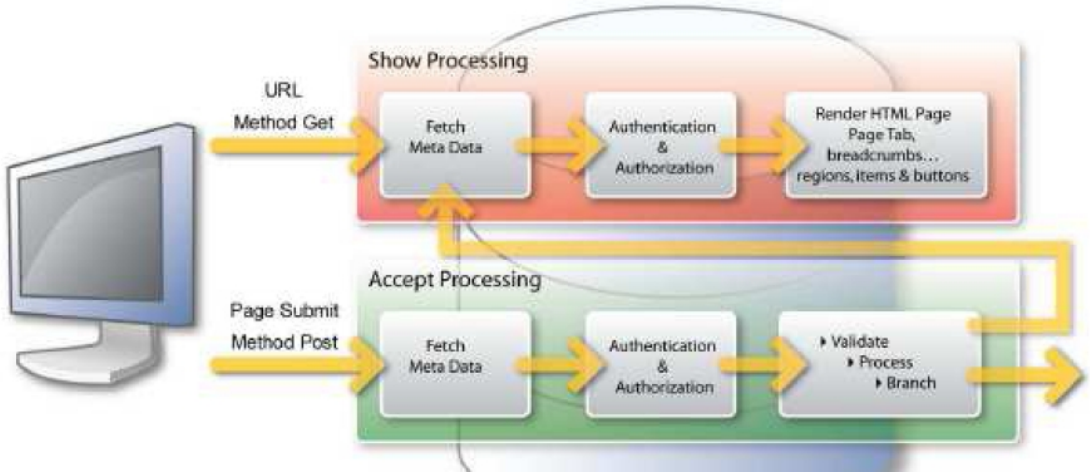

Ilustración 21. Procesamiento de una página en APEX
Fuente: Universidad de Almería

11. CONCLUSIONES

Las conclusiones que se pueden obtener alrededor del proyecto, se pueden diferenciar en dos aspectos principales: el análisis realizado a la fundación en cuanto al proceso actual y el desarrollo de un modelo conceptual donde la elaboración de un prototipo evolutivo en el control de la información en la corporación.

Puede decirse que las tecnologías que se usaran para el desarrollo de la solución son las adecuadas, pues se logra construir una herramienta con los propósitos y requisitos de la corporación teniendo como característica relevante una visión innovadora utilizando un análisis estructurado en todo el proceso.

Actualmente se utiliza el desarrollo de softwares para la elaboración y aplicación de la eficiencia de un sistema que se puede considerar se encuentra en estado crítico tomándolos en los momentos en que se encuentra a punto de colapsar y en el extremo del medio externo.

Con la realización de este trabajo se ha pretendido demostrar que Oracle Application Express es una opción a tener muy en cuenta para el desarrollo de Aplicaciones Web para la Base de Datos Oracle, además es posible realizar conexiones con otro tipo de Base de Datos mediante dblink.

Cada una de las aplicaciones realizadas con Apex cuenta con una estabilidad y eficacia garantizada la cual se puede evidenciar en el momento de realizar un análisis en el campo visual.

La ventaja de estas aplicaciones es que son un complemento total con las nuevas tecnologías implementadas lo que permite un acceso total en cualquier momento, solo basta contar con un dispositivo que permita el acceso a la web y una red de conexión, contando además con una

sistema de seguridad que solo le permite acceder a los usuarios autorizados para extraer o ingresar la información.

Con el proyecto Apex se puede garantizar un producto de calidad, altas tecnologías, velocidad en el acceso y a costos muy bajos comparados con los actualmente existentes. El ciclo de vida contempla la noción de fases generales que constituyen un marco de la situación de la corporación Elena y Juan.

Todos los objetivos específicos se llevaron a cabo satisfactoriamente, sin embargo esto no hubiera sido posible sin que el diseño del sistema adoptará una esctrucutra algebraica lógica como lo es la metodología RUP. Esta metodología, inteligentemente diseñada, permite anteponerse a los errores y facilitar su corrección a través de la estructuración modular y el flujo coherente de requisitos.

12. CRONOGRAMA

(Por favor remitirse al cronograma anexo).

13. GLOSARIO

- A -

Actor: Es una entidad externa al sistema que se modela y que puede interactuar con él.

Atributo de requerimiento: Cada uno de los atributos que se especifican para un requerimiento de software.

- B -

Blueprint: es entender los objetivos de la corporación de Elena y Juan y determinar los procesos de negocio requeridos para apoyar tales objetivos

- D -

Diagrama de Actividad: Diagrama que hace parte del lenguaje UML. Permite modelar el flujo entre un conjunto de objetos que cooperan entre sí. Son similares a los diagramas de flujo de otras metodologías diferentes a la orientada a objetos.

Definición de requerimientos de Software: Documento, o conjunto de Documentos en el cual se consignan de manera preliminar a la fase de especificación los requerimientos de software de un sistema.

Documento de casos de uso: Documento que contiene la especificación de casos de uso definidos para un sistema. Asimismo, puede contener un diagrama de casos de uso.

- G -

Grafo: Representación gráfica y matemática de los datos de una situación particular, a través de elementos particulares llamados nodos y enlaces.

- M -

Modelo conceptual: Modelo que define vistas que representan la organización de los componentes, agentes o elementos de software que participan para lograr la Funcionalidad requerida por el sistema.

- O -

OpenSource: es una expresión de la lengua inglesa. Aunque puede traducirse como “fuente abierta”, suele emplearse en castellano directamente en su versión original, sin su traducción correspondiente.

- P -

Plugins: Corresponde a la inserción de funcionalidades del sistema por medio de procedimiento específicos, que son usados dentro de la aplicación, con el fin de realizar validaciones.

- S -

Scrum: Es una Metodología Ágil de Gestión de Proyectos que se basa en la adaptación continua a las circunstancias evolutivas del Proyecto apoyándose en iteraciones cortas conocidas como s*prints* a través del siguiente ciclo:Productbacklog, Sprint Plannig, sprint Backlog, Duración de 2 a 4 Semanas, Dayly Sprint Meeting, Demostración y Retrospectiva, Nueva Versión del producto.

Stakeholder: Persona interesadas o involucradas en el desarrollo de un sistema, bajo Una perspectiva. Esta puede ser económica o relacionada otro beneficio por el Desarrollo del sistema.

- U -

UML (UnifiedModelingLanguage): Lenguaje De Modelamiento Unificado. Es un lenguaje para especificar, construir, visualizar y documentar los artefactos o ítems de Un sistema o software orientado a objetos (OO).

- T -

Team building: construcción de equipo de trabajo término usado para clarificar el objetivo, y construir un sentimiento de propiedad en el grupo.

14. REFERENCIAS

Barrera Fuentes, W. E. (2002). Ingeniería de requerimientos desde una perspectiva social. Bogotá: Universidad de los Andes.

Berszal, F. (2014). *Diseño de Bases de Datos.* iKor Consulting.

Boehm, B. (1981). *Software Engineering Economics.* New Jersey: Prentice Hall.

Booch, G. (1996). *Object Solutions: Managing the Object-oriented Project.* Addison-Wesley Publishing Company.

Campderrich Falgueras, B. (2003). *Ingeniería de Software* (Primera ed.). Barcelona, Barcelona, España: Gráficas Rey .

Cerrada, J. A. (1997). Ingeniería de Software. En M. Ortega Cantero, J. Bravo Rodríguez, & J. Ruíz Hernández, *Informática Industrial* (págs. 41-66). Cuenca, Castilla-La Mancha, España: Publicaciones Universidad Castilla-La Mancha.

Cortés B., G. (2003). *Los retos actuales para nuestra industria de software.* Bogotá: 189.

D. G. (2002). *La problemática de las PYMES en Colombia.* Bogotá.

Elena Y Juan Corporación. (s.f.). Obtenido de ppadrino.medellin@elenayjuan.org

Fedesoft. (5 de Septiembre de 2013). *fedesoft.* Obtenido de www.fedesoft.com

Meyer, B. (2009). *Touch of Class: Learning to Program Well with Objects and Contracts.* Nueva York: Springer Science & Business Media.

Naranjo, I. (2001). *Aprende a Programar con APEX.* Almería, España: Universidad de Almería.

Perens, B. (3 de Febrero de 1998). Obtenido de opensource.org

Sommerville, I. (2005). Ingeniería del Software. Madrid: PEARSON EDUCACIÓN, S.A.

Villegas, & Eduado, J. (2007). *Análisis y Diseño de Algotimos.* Bogotá D.C.: Universidad Nacional de Colombia.

15. CIBERGRAFÍA

A continuación la lista de la documentación *Web* que han servido de referencia bibliográfica complementaria durante el desarrollo del proyecto.

- Oracle Application Express, http://apex.oracle.com/ (último acceso: Julio 2014)
- Oracle Learning Library, http://www.oracle.com/technetwork/tutorials/index.html (ultimo acceso: Agosto 2014)
- El blog de Oracle APEX, http://www.oracleapex.es/ (último acceso: Agosto 2014)
- Oracle APEX World, http://www.oracleapexworld.com/ (último acceso: Agosto 2014)
- AJBD Soft, http://www.ajpdsoft.com/ (último acceso: Septiembre 2014)
- Expertos en Oracle APEX, http://www.ieskem.com/wordpress/ (ultimo acceso: Julio 2014)
- OTN Community,
- https://forums.oracle.com/community/developer/english/oracle_database/application_express (último acceso: Septiembre 2014)