UNIVERSIDAD CENTRAL DE VENEZUELA

FACULTAD DE CIENCIAS

ESCUELA DE COMPUTACIÓN

CENTRO DE ENSEÑANZA ASISTIDA POR COMPUTADOR - CENEAC

DESARROLLO DE UN SISTEMA DE INFORMACIÓN COLABORATIVO PARA EL CONTROL DE VISITAS MÉDICAS

Trabajo Especial de Grado

presentado ante la Ilustre

Universidad Central de Venezuela

Por el Bachiller

Marcel Raúl Isaac Oviedo.

para optar al título de

Licenciado en Computación

Yusneyi Y. Carballo Barrera

Caracas, 11 de Junio de 2010

Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Centro de Enseñanza Asistida por Computador - CENEAC

DESARROLLO DE UN SISTEMA DE INFORMACIÓN COLABORATIVO PARA EL CONTROL DE VISITAS MÉDICAS.

Autor: Marcel Raúl Isaac Oviedo

Tutora: Profa. Yusneyi Carballo Barrera

Fecha: 11 de Junio de 2010

RESUMEN

En este Trabajo Especial de Grado se presenta el desarrollo de un Sistema de Información Colaborativo para facilitar el control y la administración de la información de organizaciones de mercadeo de productos farmacéuticos.

Para el desarrollo del mismo se utilizan diversas tecnologías de programación, organizadas en el modelo multi-capa, entre las que podemos nombrar: *PHP Hypertext Processor* y su librería de funcionalidades *Zend Framework* para la capa lógica de negocio, Base de Datos *MySql* para la capa de persistencia de datos y *JavaScript* y su librería de funcionalidades Jquery. La arquitectura que se utiliza para construir el Sistema de Información es la llamada Modelo Vista Controlador.

Todo el desarrollo se apoya en la metodología de desarrollo Scrum tomando las historias de usuario de la Programación Extrema (XP).

Con el desarrollo de este Sistema, se logra la disminución de los tiempos de respuesta en las operaciones de la empresa, así como la automatización de diversas tareas, el control de las actividades y centralización de la información.

Palabras Claves: Sistemas de Información, aplicaciones de trabajo colaborativo, Control de visitas, controles de eventos, Lenguajes de Programación, Servidor Web, Base de Datos, Programación Extrema, *Scrum*, *MySql*, *PHP*, *Zend Framework*, *Jquery*, *JavaScript*, *AJAX*, *CSS*, *HTML*, *Web* 2.0.

Tabla de Contenido

INTRODUCCIÓN 1

Capítulo I – PROBLEMA DE INVESTIGACIÓN 3

1.1 Contexto 3

1.2 Planteamiento del problema 4

1.3 Objetivos general y específicos 5

1.4 Justificación e Importancia 6

1.5 Límites y alcances 6

Capítulo II - MARCO TEÓRICO 8

2.1 Sistemas de Información 8

2.1.1 Sistemas de Procesamiento de Transacciones (TPS) 8

2.1.2 Sistemas de Trabajo del Conocimiento (WKS) y los 2.1.3 Sistemas de Oficina 9

2.1.4 Sistemas de Información Gerencial (MIS) 9

2.1.5 Sistemas de Apoyo a la Toma de Decisiones (DSS) 9

2.1.6 Sistemas de Información Personales (PIS) 10

2.1.7 Sistemas de Información Colaborativos (CIS) 10

2.2 Arquitecturas y Tecnologías para el desarrollo Web 11

2.2.1 Patrón de Arquitectura Modelo-Vista-Controlador (MVC) 11

2.2.2 Tecnologías para el desarrollo Web 13

2.3 Metodología de Desarrollo Ágil 21

2.3.1 Programación Extrema XP 22

2.3.2 Scrum 25

Capítulo III - MARCO APLICATIVO 37

3.1 Definición de Roles del Proyecto 37

3.2 Escribiendo el Inventario de Funcionalidades 38

3.3 Diseño de la Aplicación 47

3.3.1 La Base de Datos 47

3.3.2 Tecnologías involucradas en el desarrollo de la Aplicación 51

3.3.3 Interfaces de Usuario..... 54

3.4 Las Iteraciones 60

3.4.1 Primera Iteración: configuraciones básicas del sistema..... 61

3.4.2 Segunda Iteración: Manejo de ciclos y actividades del sistema 61

3.4.3 Tercera Iteración: Gestión de territorios..... 62

3.4.4 Cuarta Iteración: Gestión de listado de médicos..... 63

3.4.5 Quinta Iteración: Planificación de vistas y reporte..... 64

CONCLUSIONES.....65

RECOMENDACIONES.....67

BIBLIOGRAFÍA.....68

Índice de Ilustraciones y Tablas

Figura 1: Diagrama de Clases de la Arquitectura Modelo Vista Controlador (Buschman, y otros, 1996) . 12
Figura 2: Fases del proceso Scrum (Palacio, 2008) 27
Figura 3: Modelo Entidad Relación de la Aplicación (E/R) 47
Figura 4: Estructura del Directorio de la Aplicación 54
Figura 5: Pantalla de Gestión de Ciclos 55
Figura 6: Elemento Acordeón 56
Figura 7: Elemento Pestañas 56
Figura 8: Pantalla de Gestión de Territorios 57
Figura 9: Despliegue de Líneas y Distritos 57
Figura 10: Uso de Ventanas en acciones 58
Figura 11: Gráfico de Progreso del Ciclo 58
Figura 12: Pantalla de Planificación de Ciclo 59
Figura 13: Pantalla de Reporte de Visitas 60

INTRODUCCIÓN

Los Sistemas de Información y las Tecnologías de Información cambiaron la forma en que las organizaciones operan. La automatización de los procesos operativos, la implantación de plataformas de información necesarias para la toma de decisiones y el incremento en las ventajas competitivas son importantes mejoras que estas organizaciones han logrando.

Las Tecnologías de Información integran diversas soluciones de computación, utilizando componentes de equipamiento, software e infraestructura; las telecomunicaciones con sus diversos mecanismos de intercambio de información y técnicas para el procesamiento de datos. Muchas soluciones también integran al factor humano, los contenidos de la información, los elementos externos a la organización y regulaciones, además de los recursos financieros.

Estas Tecnologías son el núcleo central de una transformación multidimensional que experimentan los mecanismos de manejo de información, esquemas de trabajo de las organizaciones y la forma de operar de sus empleados. Impulsan el crecimiento de las organizaciones, haciéndolas más eficientes y modificando los hábitos, patrones de conducta y la forma de pensar y trabajar de los empleados.

La información ha pasado a ser uno de los principales recursos que poseen las empresas. Esto ha motivado a que las organizaciones hayan reconocido la importancia de administrar, no solamente los principales recursos como la mano de obra y las materias primas, sino también la información como recurso crítico.

Para las organizaciones, la información ya no es subproducto de la conducción empresarial, sino uno de los tantos factores necesarios para la determinación del éxito o fracaso de los negocios, sustentando en forma considerable la toma de decisiones.

El manejo de información generada por una computadora difiere en forma significativa del manejo de datos producidos manualmente. Adicionalmente, la fácil disponibilidad de manejo de este insumo, permiten que la sociedad y los negocios se mantengan evolucionando constantemente de forma más eficiente.

El crecimiento de la Internet y sus nuevas tecnologías de desarrollo, ha mejorado drásticamente la manera en que los Sistemas de Información se comunican y comparten la información. Convirtiendo a la Internet en una de las primeras consideraciones a tomar por toda organización, a la hora de desarrollar un nuevo sistema que le permita manejar su información permitiendo acceder de forma instantánea a datos que son generados desde lugares remotos como puede ser el territorio completo de una nación, e inclusive del mundo entero, representando una ventaja obvia sobre las antiguas maneras tradicionales de consulta de información.

Toda esta infraestructura está montada sobre lo que actualmente se conoce como *World Wide Web*, ("La Web"). Mediante la implantación de tecnologías para el desarrollo de aplicaciones, esta plataforma ha evolucionado hasta convertirse en lo que actualmente se conoce como Web 2.0. Expresado en resumen en las palabras de Ribes (Ribes, 2007): …*"podemos considerar como Web 2.0 todas aquellas utilidades y servicios de Internet que se sustentan en una base de datos, la cual puede ser modificada por los usuarios del servicio, ya sea en su contenido (añadiendo, cambiando, borrando información o asociando metadatos a la información existente), bien en la forma de presentarlo o en contenido y forma simultáneamente."*

La popularidad de los Sistemas de Información Colaborativos, se ha incrementado significativamente conforme se desarrolla la Internet, surgiendo varias aplicaciones que han cambiado la forma de interactuar de los estructuras de una sociedad. Ejemplos típicos de estos sistemas son los ambientes de enseñanza a distancia (*e-learning*, *Moodle*), las comunidades virtuales (*Facebook*, *Twitter*, *digg*, *Wikipedia*), los foros de discusión (*PHPbb*), los sistemas de realidad virtual (*Second Live*, *The Sims*) o los sistemas de comunicación instantánea (*ICQ, Skype, Windows Live Messenger, Net Meeting*), entre otros.

El objetivo de este Trabajo Especial de Grado es la implementación de las necesidades de una organización que centra sus actividades en la distribución y comercialización indirecta de productos farmacéuticos mediante su promoción en la población de médicos y así llegar al público general.

Este documento está estructurado en los siguientes capítulos:

Capítulo 1 – Planteamiento del Problema: se define el contexto en el cual se desenvuelve la organización que sirve de caso de estudio, para describir cómo se realizan sus operaciones y la forma actual de llevar el control de su información. Este conocimiento permitió identificar las principales deficiencias en el manejo de la información y esbozar los requerimientos del cliente que fueron considerados en la solución.

Capítulo 2 – Marco Teórico: presenta las definiciones y términos referentes a los Sistemas de Información, tipos, características, generalidades y cómo pueden ser implantados en una organización, las Tecnologías de Desarrollo Web necesarias incluidas en el modelo multicapa. *PHP* y *Zend Framework* en la capa lógica de negocio, MySql en la capa de datos y *JavaScript*, con su librería de trabajo *Jquery* y la tecnología de comunicación asíncrona *AJAX*, en la capa de interfaz. Se describe de igual forma el proceso de desarrollo ágil *Scrum*.

Capítulo 3 – Marco Aplicativo: se describen las diferentes estructuras diseñadas en la construcción de la herramienta. Se define la estructura de la Base de Datos, las principales actividades incluidas en el Inventario de Funcionalidades del sistema y se indican los diferentes pasos seguidos para el cumplimiento del objetivo principal.

Este documento finaliza presentando las conclusiones del Trabajo Especial de Grado, recomendaciones que permitirán la evolución de la aplicación, la bibliografía y referencias consultadas.

Capítulo I – PROBLEMA DE INVESTIGACIÓN

1.1 Contexto

La organización seleccionada como caso de estudio cuenta con un personal dedicado a realizar visitas periódicas a la comunidad médica. Promocionando y haciendo seguimiento de los productos que son recetados por los médicos en sus consultas. La misión de esta organización es *"Poner a la disposición de la comunidad médica y de la colectividad una amplia gama de productos, de altísima calidad y de cualidades y formas farmacéuticas innovadoras, destinadas y orientadas a proporcionar una mejor calidad de vida"*.

Es primordial que dichas visitas periódicas sean constantes y seguidas, para garantizar que los médicos comiencen a recetar o recomendar los productos promovidos por la organización, asegurando su introducción al público en general.

La empresa ha definido para esto, un período con una duración aproximada de 20 días hábiles en el cual cada visitador realiza al menos una visita. Este período es llamado "Ciclo". En un año se realizan un máximo de 10 Ciclos, debido a la necesidad de cumplir con otras actividades propias del ambiente laboral como reuniones con supervisores, gerentes e inclusive reuniones a nivel nacional.

La empresa presta sus servicios para diferentes Líneas de Producto a nivel nacional. Una línea de producto agrupa diferentes artículos relacionados por la especialidad médica a la cual están dirigidos, por ejemplo, cardiología, obstetricia y oftalmología. Las líneas son dirigidas por el Gerente de Línea y estructuran sus operaciones en zonas geográficas llamadas: Distritos, Territorios y *Bricks*.

Un Distrito se define como una región que abarca uno o más estados del país. A los distritos se le asignan Territorios. El Gerente de Distrito es el encargado de velar por el desarrollo de las operaciones en su zona. La empresa ha definido los siguientes distritos y los estados que abarca:

- 01 - Maracaibo: Zulia y Falcón.
- 02 - Andes: Mérida, Táchira y Trujillo.
- 03 - Barquisimeto: Lara, Yaracuy, Portuguesa y Barinas.
- 04 - Caracas: Caracas, Vargas y Miranda.
- 05 - Valencia: Aragua, Carabobo, Guárico, Apure, parte del *brick* Tucacas.
- 06 - Oriente: Anzoátegui, Bolívar, Monagas, Sucre, Nueva Esparta y Delta Amacuro.
- 07- Resto del país.

Los Territorios son subregiones de los distritos, definen el área de acción de un visitador o representante, para buscar los potenciales médicos a incluir en el Ciclo. Un territorio se compone de zonas llamadas *Bricks*. Un *Brick* es la región más pequeña de la estructura y fueron definidos por la empresa tomando los códigos postales de área. Por ejemplo, algunos de los *Bricks* de un territorio en la región capital podrían ser:

- 1021 – El Paraíso.
- 1030 – Catia.
- 1041 – Colinas de Bello Monte.
- 1051 – Ciudad Universitaria.

Se lleva el control de estos territorios mediante una numeración que se compone por 3 pares de números que indican lo siguiente:

- El primer par indica el número de Distrito.
- El segundo par indica el número de Línea.
- El tercer par indica el número de Territorios.

Por lo tanto si queremos definir un segundo (02) territorio para la Línea Número 03, Distrito Maracaibo (01), el código con el que se refieren a esa región es 010302.

Para la organización es importante tener la información de los médicos a visitar, consideran importante conocer el potencial que tiene un médico para aceptar y recetar los productos. También desean conocer los intervalos de tiempo que se producen entre visitas para evitar el hostigamiento al realizar una visita muy pronto, o por el contrario el olvido al realizarla en una fecha muy alejada desde la última vez en la que se visitó.

En la planificación, un visitador debe cumplir con una cuota diaria de visitas y generar un reporte de las novedades. Esto permite llevar un control de las visitas realizadas, el rendimiento del visitador y verificar que las metas se están cumpliendo.

1.2 Planteamiento del problema

Los procesos anteriormente descritos, se realizan con herramientas que no son las más adecuadas, por ejemplo hojas de cálculo creadas por los mismos empleados de la empresa, los cuales deben ser enviadas constantemente a los supervisores, ocasionando las siguientes desventajas:

Las hojas de cálculo pueden ser modificadas a conveniencia. Generando formatos variados, dificultando la unificación de criterios entre los datos representados, generando dificultad a la hora de procesar los datos para hacer los reportes.

Para obtener la información de los visitadores, se debe esperar a que cada empleado suministre la hoja correspondiente al período solicitado. Esto debe ocurrir al final de cada Ciclo, pero en general se produce varios días después de la fecha de cierre, ocasionando retrasos y burocratización de los procesos.

La cantidad de hojas de cálculo generadas al final de cada ciclo por cada uno de los visitadores es excesiva y resulta muy difícil hacer seguimiento de la información almacenada en ellas.

Cuando se requiere un reporte tabulado con la información de rendimiento se debe utilizar cada una de las hojas de cálculo para obtener los datos. No se logran reportes de forma sencilla y oportuna,

pues es necesario que los visitadores envíen la información a tiempo y en su totalidad al final de cada período o ciclo.

No se tiene control de los datos de la población de médicos y de sus formas de contacto, ya que cada visitador puede agregar la cantidad de médicos que le eran solicitados, sin tener la garantía de que otro visitador también lo había tomado en cuenta, generando datos cruzados a la hora de hacer los reportes y duplicidad en los registros.

Considerando entonces las deficiencias que se presentan en el manejo de la información, su falta de uniformidad, los retrasos e incumplimiento de entregas de reportes, duplicación y pérdida de datos, además de la compleja estructura de información que la organización maneja, se observa como problema principal la carencia de un Sistema de Información, preferiblemente Colaborativo y sobre ambiente Web, que permita controlar la información de las operaciones de la empresa involucradas en la planificación de actividades, manejo de personal, asignación de tareas y responsabilidades a los empleados, gestión de zonas geográficas, planificación y reporte de visitas y generación de informes para facilitar el análisis de la información.

1.3 Objetivos general y específicos

Desarrollar un Sistema de Información Colaborativo en ambiente Web que permita controlar y planificar la información asociada a visitas médicas, así como agilizar y simplificar las actividades operativas para la configuración de territorios, definición de ciclos, asignación de personal, planificación de visitas, reporte de visitas, manejo de listados de médicos y generación de informes.

A fin de lograr el cumplimiento del objetivo general, se plantearon los siguientes objetivos específicos:

1. Realizar un levantamiento de la información manejada por la empresa en relación con sus actividades, procesos, perfiles, responsabilidades, organización de territorios, entre otras.
2. Realizar un análisis de la información obtenida para establecer los principales requerimientos de la organización y funcionalidades de la herramienta que se desea desarrollar.
3. Realizar el diseño preliminar de un Sistema de Información Colaborativo de manera de establecer los lineamientos de los principales procesos e interfaces o pantallas.
4. Determinar la prioridad de desarrollo de los procesos que forman parte de la aplicación.
5. Construir la base de datos del sistema partiendo de los datos recopilados durante el proceso de levantamiento de información.
6. Instalar y configurar el servidor de pruebas del sistema.
7. Diseñar y desarrollar funcionalidades de seguridad para identificar y autenticar a cada usuario del sistema.
8. Utilizar el proceso de Desarrollo *Scrum* para llevar el control de las actividades de construcción de la aplicación.
9. Cargar la Base de datos con información de prueba para verificar su funcionamiento.

10. Realizar pruebas para verificar el funcionamiento del sistema y las mejoras necesarias para responder a las expectativas del usuario.
11. Instalar el sistema en un servidor de aplicaciones y realizar los ajustes finales de configuración que sean necesarios.

1.4 Justificación e Importancia

Las organizaciones que distribuyen productos farmacéuticos y hospitalarios abarcan regiones a nivel nacional, lo cual aunado al crecimiento de sus operaciones, complica el manejo de sus actividades y la administración de su información.

Para ello fue imprescindible la implantación de un Sistema de Información Colaborativo, el cual les permite tener la información necesaria para apoyar la toma de decisiones, llevar el control de su personal y enfocar su fuerza de trabajo.

También les permite enfocar los esfuerzos de los empleados para lograr llevar a la comunidad médica y hospitalaria la mayor cantidad de productos que benefician al consumidor, evitando grandes pérdidas de tiempo en llenar tediosos y obsoletos formatos a ser remitidos a la oficina central.

Cabe destacar que la gerencia de la empresa puede conocer de forma inmediata las regiones en donde sus productos tienen mayor receptividad, permitiendo tomar decisiones de mercadeo y distribución, además de conocer aquellos empleados que se destacan sobre los demás realizando un trabajo cumplido y eficiente.

Uno de los principales beneficios para la empresa es la automatización y el control de procesos y de personal, pues mediante un sistema de información como el propuesto, se pueden realizar las asignaciones de personal a territorio, tener una base de datos de los médicos a los que se les recomiendan los productos y principalmente controlar la información que maneja la empresa.

1.5 Límites y alcances

El alcance que tuvo la aplicación desarrollada como propuesta de solución abarcó el desarrollo de los siguientes módulos:

- Control y permisos de acceso e identificación de los usuarios.
- Ingreso, clasificación y modificación de médicos.
- Creación, modificación, eliminación y consulta de los usuarios que interactúan en el sistema, en los niveles: Visitador y Administrador.
- Manejo de territorios incluyendo la asignación y manejo del personal a cargo del mismo, así como modificación de la configuración de las áreas que abarca.
- Módulos de gestión de ciclos y días hábiles para crear, eliminar, modificar y consultar las actividades que rigen las operaciones de los visitadores de la empresa.
- Planificación de visitas que incluye visitas a médicos y hospitales y la reprogramación de visitas.

- Reporte de vistas.
- Consultas y Reportes de informaciones varias: Reporte de médicos, Reporte de frecuencias de visitas, Reporte de indicadores de visitas, entre otros.

Capítulo II - MARCO TEÓRICO

2.1 Sistemas de Información

(Fernández Alarcón, 2006) Un Sistema de Información es un conjunto de personas, datos, procesos y tecnologías de la información que interactúan para recoger, procesar, almacenar y proveer la información necesaria para el correcto funcionamiento de la organización.

Con el fin de construir un sistema de información eficaz y eficiente, los responsables de su desarrollo deben ser capaces de combinar de forma eficaz los distintos componentes que constituyen dichos sistemas. Algunos de ellos pueden ser:

- **Personas:** directivos, usuarios, diseñadores, analistas, etc. Forman parte de un subgrupo de empleados que pertenecen a una organización con un conocimiento específico sobre temas relacionados al Sistema de Información al que pertenecen.
- **Datos:** materia prima para crear información útil. Representan la información necesaria para alcanzar las funciones básicas y los objetivos de un Sistema de Información.
- **Procesos:** Son las actividades propias de la organización que transforman los datos en información.
- **Tecnologías:** el hardware y el software necesario para el procesamiento de la información y que sostienen a los tres componentes anteriores.

(Laudon, y otros, 2008) Los sistemas de Información se pueden clasificar en función del nivel organizacional en donde son necesarios. Se identifican entonces cuatro niveles: el nivel estratégico, el nivel administrativo, el nivel de conocimiento y el nivel operativo. A continuación se analizan algunos de los Sistemas de Información diseñados a cubrir las necesidades en los distintos niveles organizativos:

2.1.1 Sistemas de Procesamiento de Transacciones (TPS)

Los Sistemas de Procesamiento de Transacciones *(Transaction Processing System)* son aquellos que se encargan de capturar y procesar datos sobre las transacciones de negocios que se realizan diariamente en una empresa a nivel operativo. Una transacción es un hecho o actividad que se registra en una empresa.

Los *TPS* poseen las siguientes características:

- Procedimientos definidos y rutinarios
- Registran una cantidad enorme de información
- Son utilizados para registrar el día a día de una empresa
- Pueden ser utilizados en más de un área en la misma organización

- Generalmente son los sistemas encargados de recopilar los datos primarios utilizados en otros sistemas de información más especializados.

2.1.2 Sistemas de Trabajo del Conocimiento (WKS) y los 2.1.3 Sistemas de Oficina

Los sistemas de trabajo de conocimiento *(Working Knowledge System)* promueven la creación de nuevo conocimiento, permitiendo su integración con la experiencia adquirida durante su creación. Ofrecen sus servicios en el nivel de conocimiento de la empresa.

Las principales características de estos sistemas son:

- Son utilizados principalmente por usuarios especializados en las operaciones de la empresa.
- Generalmente son utilizados en operaciones de ingeniería o creación.
- Son sistemas que proporcionan un grado perfeccionado de comunicación entre los empleados especializados de una organización.
- Manejan y administran documentos a través de procesamiento de texto, digitalización de documentos, programación mediante calendarios electrónicos, y comunicación a través de correo electrónico, correo o videoconferencia.

2.1.4 Sistemas de Información Gerencial (MIS)

Un sistema de información gerencial *(Management Information System)* proporciona informes orientados a la gestión basados en el procesamiento de transacciones y operaciones de la organización.

Los *MIS* poseen las siguientes características:

- Son utilizados por los líderes de área o supervisores a nivel administrativo.
- Se encargan de resumir las transacciones registradas por los TPS.
- Son utilizados en períodos regulares de tiempo.
- Proporcionan informes estructurados y poco flexibles.
- Dichos informes se basan en información del pasado de la empresa.
- Apoyan servicios internos de la organización.

2.1.5 Sistemas de Apoyo a la Toma de Decisiones (DSS)

Un Sistema de Apoyo a la Toma de Decisiones *(Decision Support System)* está orientado a identificar oportunidades o proporciona la información necesaria en la toma de decisiones.

Sus principales características son:

- Proporcionan servicio a nivel gerencial.

- Son utilizados para resolver problemas no estructurados, es decir, problemas que no se pueden prever.
- Debe permitir y disponer de una gran flexibilidad para adaptarse a cualquier tipo de situación.
- Proveen un gran número de herramientas de análisis que permitan un estudio analítico profundo.
- Toman los datos de los TPS, MIS y utilizan fuentes externas de la organización que les proporcionan información sobre competidores, clientes, mercados, proveedores, etc.
- Permite simular resultados en base a los acontecimientos presentes y pasados de la organización y del entorno.
- Pueden ser utilizados para la evaluación de estrategias para el lanzamiento de nuevos productos, o la evaluación de diversas alternativas en un largo periodo de tiempo.

2.1.6 Sistemas de Información Personales (PIS)

(Cárdenas Rosado, y otros, 2002) Un Sistema de Información Personal (Personal Information System) se enfoca en incrementar la productividad, mediante el manejo de la información de un individuo. Entre los PIS que podemos nombrar como ejemplos tenemos: las hojas de cálculo, sistemas de procesamiento de palabras, agendas electrónicas, calendarios, entre otros.

Las características de los Sistemas de Información Personales son:

- Sincronizan la información que manejan entre los diversos dispositivos que los pueden contener como computadoras personales, dispositivos móviles, portales de internet, entre otros.
- Su principal uso es el de almacenar y recuperar la información manejada por un individuo, que por lo general incluye: Notas personales y diario, direcciones, tareas, fechas importantes de calendario como cumpleaños, aniversarios y reuniones, recordatorios, correo electrónico personal, lectores de noticias, etc.

2.1.7 Sistemas de Información Colaborativos (CIS)

Se define como Software Colaborativo (*Groupware*)[1] al conjunto de programas informáticos que integran el trabajo en un sólo proyecto con muchos usuarios concurrentes que se encuentran en diversas estaciones de trabajo, conectadas a través de una red (intranet o Internet).

(FCEA) Las características del Sistema de Información Colaborativo son:

- Son herramientas que incrementan la productividad al reducir trámites y el procesamiento de papel.
- Reducen el tiempo de respuesta de los procesos.

[1] Obtenido de: http://es.wikipedia.org/wiki/Groupware. Mayo 2010

- Monitorean el estado de avance de los procesos.
- Permiten asignar roles a cada empleado dentro del sistema, modelando sus responsabilidades dentro de la organización.
- Generan reportes para identificar oportunidades o deficiencias en los procesos.
- Reducen el consumo de papel al implementar procesos automatizados y eficientes.

Existen varios tipos de Sistemas Colaborativos, entre los que encontramos:

- **Herramientas de comunicación electrónica**: envían mensajes, archivos, datos o documentos entre personas y facilitan el envío de información (colaboración asíncrona), como por ejemplo:
  - Correo electrónico.
  - Correo de voz.
  - Publicación en Web.
- **Herramientas de conferencia:** facilitan el flujo de información, de forma interactiva e inmediata (colaboración síncrona), como por ejemplo:
  - Conferencia de datos.
  - Conferencias de voz.
  - Conferencias de video.
  - Salas de chat o mensajería instantánea.
  - Sistemas para facilitar reuniones.
- **Herramientas de gestión colaborativa:** facilitan las actividades del grupo, como por ejemplo:
  - Calendarios electrónicos.
  - Sistemas de gestión de proyectos.
  - Sistemas de control de flujo de actividad.
  - Sistemas de gestión del conocimiento.
  - Sistemas de soporte a redes sociales.

Todos estos sistemas de información están apoyados sobre un compendio de tecnologías (*Software* y *Hardware*) que le permiten su funcionamiento óptimo y eficiente. Si se desarrolla una aplicación basada en un sistema de información, se deben utilizar arquitecturas y tecnologías que soporten las funcionalidades descritas anteriormente. A continuación se describen algunas de ellas.

2.2 Arquitecturas y Tecnologías para el desarrollo Web

2.2.1 Patrón de Arquitectura Modelo-Vista-Controlador (MVC)

(Battlez, 2007) El patrón de Arquitectura Modelo-Vista-Controlador (MVC), divide una aplicación interactiva en tres áreas o componentes: procesamiento, salida y entrada. Fue ideado por Trygve Reenskaug entre 1978 y 1979 mientras trabajaba en la compañía Xerox y fueron desarrolladas para ser implementadas en SmallTalk-80. El objetivo principal de la utilización de esta estructura es la de reducir la

distancia entre el modelo mental del usuario y el modelo computacional. Esta estructura utiliza las siguientes abstracciones:

- **Modelo:** Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada. Es la representación específica de los datos con la cuales opera la aplicación. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
- **Vista:** obtiene información del modelo y la presenta al usuario. Generalmente es conocida como la interfaz de usuario y les permite visualizar e interactuar con el sistema.
- **Controlador:** recibe las entradas del usuario, generalmente en forma de eventos que se traduce en solicitudes de servicio al modelo o a las vistas.

En la Figura 1 se muestra el diagrama de clases de esta arquitectura. Cuando el modelo cambia, vía el controlador de una vista, todas las vistas que dependan del cambio deben reflejarlo. El modelo notifica a todas sus vistas y controladores registrados cuando ocurre un cambio. Las visitas recuperan los datos del módulo y actualizan la información que presentan. Los controladores registrados recuperan los datos del modelo para activar o desactivar ciertas funciones al usuario.

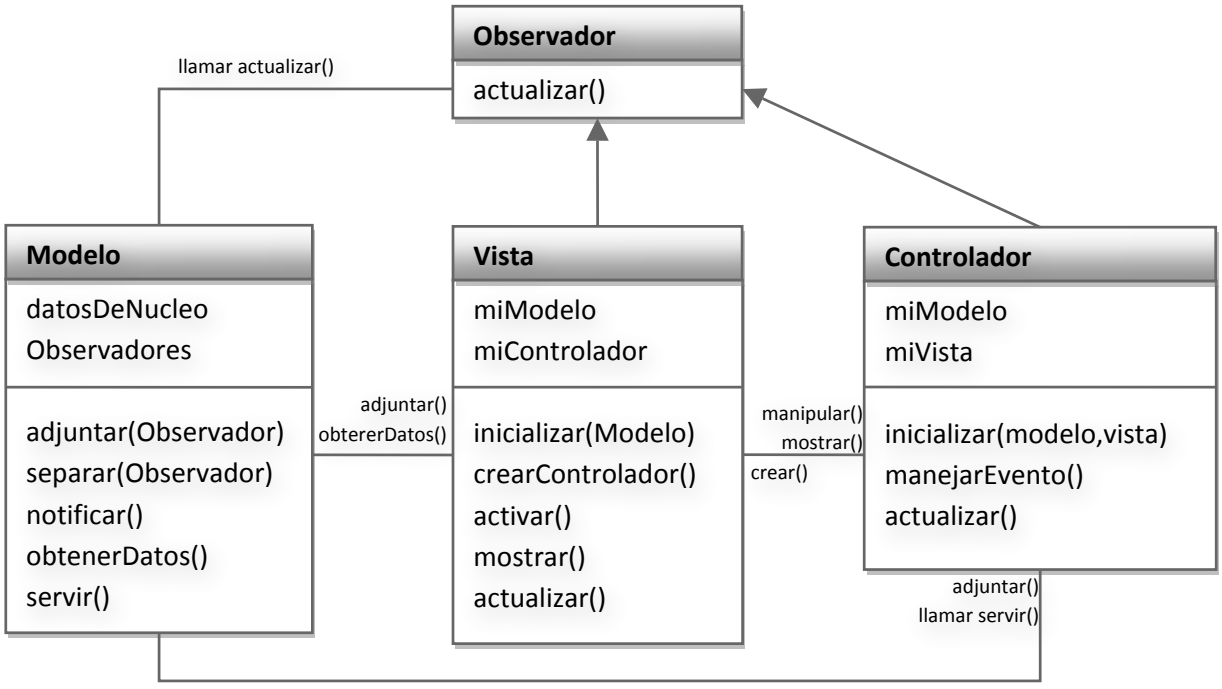

Figura 1: Diagrama de Clases de la Arquitectura Modelo Vista Controlador (Buschman, y otros, 1996)

Esta separación está inspirada en un modelo de procesamiento de información donde el Controlador representa las entradas, el Modelo el procesamiento y la vista las salidas.

El uso de este patrón de arquitectura brinda las siguientes ventajas:

- Implementación por separado de los componentes de la aplicación: permitiendo identificar y reparar errores sin afectar el resto del programa o componentes.
- Vistas y controladores conectables: se pueden incorporar nuevas y/o cambiar vistas y controladores de un modelo para ser representado de diferentes formas y con diferentes acciones. Por ejemplo: un modelo puede ser representado en una versión de programa nativo del sistema operativo, en una interfaz *WEB* o en dispositivos móviles
- Vistas Sincronizadas: la propagación del efecto que produce un cambio en el modelo es inmediata en todas las vistas y controladores del sistema. Por ser el modelo la pieza central de la aplicación, las vistas y controladores dependen de este.
- Facilita el escalamiento de la aplicación: nuevos módulos pueden ser agregados al sistema sin afectar los existentes.

Las desventajas del uso de este patrón son:

- El tiempo de desarrollo de una aplicación que usa el patrón de arquitectura MVC es mayor, al menos en sus primeras etapas, que el tiempo de desarrollo de una aplicación que no lo implementa, ya que MVC requiere que el programador desarrolle una mayor cantidad de clases que en un entorno de desarrollo común no son necesarias. Sin embargo, en la etapa de mantenimiento de la aplicación, una aplicación MVC es más mantenible, extensible y modificable que una aplicación que no lo implementa.
- MVC requiere la existencia de una arquitectura inicial sobre la que se deben construir clases e interfaces para modificar y comunicar los módulos de una aplicación. Esta arquitectura inicial debe incluir, por lo menos: un mecanismo de eventos para poder proporcionar las notificaciones que genera el modelo de aplicación; una clase Modelo, otra clase Vista y una clase Controlador genéricas que realicen todas las tareas de comunicación, notificación y actualización que serán luego transparentes para el desarrollo de la aplicación. Con el uso de paquetes de programación (Framework) y de navegadores de Internet, que facilitan el manejo de eventos, este tipo de implementaciones ya han sido tomados en cuenta.
- MVC es un patrón de arquitectura orientado a objetos por lo que su implementación es sumamente costosa y difícil en lenguajes que no siguen este paradigma.

Por lo general, las librerías de desarrollo de aplicaciones basadas en ambiente *WEB* como *Struts* (The Apache Software Fundation, 2010) en lenguaje Java y *Zend Framework* (Zend Technologies Ltd., 2010) en PHP, implementan el patrón de arquitectura MVC.

2.2.2 Tecnologías para el desarrollo Web

El desarrollo de aplicaciones de escritorio suele generar importantes problemas de escalabilidad, disponibilidad, seguridad, integración y distribución en los sistemas de gestión distribuidos utilizados por las empresas. Esto sobre todo ocurre cuando la empresa no concentra sus operaciones en una misma red u oficina y se depende de la disponibilidad de conexión y de la distribución de la aplicación en las distintas terminales y plataformas.

En los últimos años, se ha optado por el desarrollo de aplicaciones utilizando las ventajas que ofrecen la Web, ya que la distribución de un sistema se realiza de forma inmediata, permite que las actualizaciones o mejoras del sistema no dependan de la terminal en la que se corra y permite que sea accedido prácticamente utilizando cualquier dispositivo con una conexión a Internet.

Uno de los enfoques utilizados actualmente en el desarrollo de aplicaciones, es el llamado arquitectura multicapa *(multi-tier)*, el cual permite que las éstas evolucionen de forma sencilla y eficiente. Esta Arquitectura divide las aplicaciones en dos, tres o más capas. La arquitectura de dos capas es el conocido como arquitectura Cliente-Servidor en el que uno o varios programas (clientes), que se pueden encontrar generalmente en una ubicación remota, hacen peticiones o solicitan servicios a otro programa centralizado llamado Servidor. El servidor se encargará de responder las solicitudes e interconectar a los clientes entre sí. La arquitectura de tres capas, es la más popular y es utilizada en aplicaciones desarrolladas en web, implementa la solicitud de servicio en tres niveles. A continuación se describen las tecnologías utilizadas en cada una de estas:

Capa de persistencia o almacenamiento

Esta capa se encarga de almacenar y recuperar los datos persistentes de la aplicación. Generalmente está conformada por un servidor de Base de Datos, el cual permite que la información se mantenga neutral e independiente al modelo e interfaz que la utilice, permitiendo la escalabilidad de información, apoyando la modularidad del sistema, facilitando su actualización y la administración de su información. Uno de los servidores de Base de Datos utilizados en esta capa de persistencia es el *MySql.*

El servidor de Base de Datos *MySql* es un sistema manejador de base de datos relacional, multihilo, multiusuario y de código abierto desarrollado por la empresa *MySql AB* subsidiaria de *Sun Microsystems* (MySql AB, 2010) y de distribución gratuita con fines educativos. Está desarrollado en su mayor parte en *ANSI C*.

Entre las ventajas que ofrece este manejador de base de datos relacional tenemos:

- Soporte a multiplataforma y multihilos.
- Soporte a campos de texto que permiten ingresar valores alfanuméricos de longitudes variables, evitando el almacenaje de caracteres vacíos.
- Incorpora una base de datos que almacena información acerca de todas las otras bases de datos que existen en el servidor.
- Motores de almacenamiento independientes que permiten optimizar la estructura dependiendo del contenido de cada tabla.
- Consultas optimizadas gracias a la implementación de un sistema de memoria temporal.
- Soporte para consultas anidadas.
- Búsqueda indexada de campos de texto completos.
- Soporte completo para distintos set de caracteres.
- Soporte de transacciones atómicas, consistentes, aisladas y durables.
- Completo soporte para cláusulas del lenguaje SQL.

- Soporta una capacidad de datos considerable.
- Los clientes se conectan al servidor utilizando la red y es independiente de la plataforma.
- Se distribuye bajo licenciamiento Software Libre por lo que no se asumen costos de adquisición y uso.

Sus desventajas más significativas:

- En comparación con otros servidores de Bases de Datos, es un sistema primitivo, ya que no ofrece la misma cantidad de funciones.
- Si no se tiene cuidado con el manejo de la información, se puede incurrir en la inconsistencia de información (Barwick, 2009).
- Es una Base de Datos que no ofrece un cliente gráfico de manejo de la Base de Datos. Su única herramienta nativa es la consola de comandos, por lo que hay que recurrir a software de terceros.

Capa de Negocio

Esta segunda capa se encarga de validar los datos de entrada del usuario y ejecutar los procesos de negocios, coordinar la aplicación, procesar comandos, tomar decisiones lógicas y evaluaciones, y realizar cálculos. También se encarga de mover y procesar información entre las capas de persistencia y la de presentación.

En una arquitectura de tres capas implementada en una aplicación *WEB*, la capa de negocio se localiza en el servidor y se utilizan lenguajes de programación como *JSP (Java Server Pages)*, *ASP (Active Server Pages)* y *PHP (PHP Hypertext Processor)*.

(Achour, y otros, 2010) ***PHP*** es un lenguaje de programación interpretado para el desarrollo de tecnologías *Web*. No es necesario generar un objeto ejecutable o compilarlo para correr un programa desarrollado en él. Se utiliza principalmente para la generación de páginas Web dinámicas, ejecución de scripts desde la línea de comando o la generación dinámica de archivos.

(Stewart, 2006) Entre las ventajas del uso de este lenguaje tenemos:

- Es uno de los más populares lenguajes de programación en funcionamiento hoy en día.
- Es rápido, estable, seguro, fácil de usar y de código abierto.
- El código *PHP* se inserta directamente en el código *HTML* de un sitio Web.
- El usuario no necesita ningún navegador especial o componente de instalación adicional para ejecutar un programa.
- Es fácil de entender y aprender.
- Fue concebido para funcionar en diversas plataformas.
- *PHP* no requiere mucho uso del sistema o recursos
- Su estabilidad está sustentada gracias al aporte de la comunidad de programación que lo utiliza.

- Posee niveles y configuraciones de seguridad que pueden configurarse dependiendo de las necesidades de uso
- Gran variedad de librerías modulares que permiten un número considerable de acciones.
- Capacidad de conexión con diversas plataformas de Bases de Datos

(Spolsky, 2006) Algunas de sus desventajas:

- Código redundante y desordenado. Un programador puede encontrar la solución a un problema utilizando diferentes vías
- Nombres de funciones o procedimientos no estandarizados
- Carece de un buen soporte y estabilidad en la programación orientada a objetos

Por lo general, en la programación de aplicaciones en ambiente Web, se utilizan librerías de desarrollo que facilitan e implementan tareas de comunes y frecuentes, brindando una base sólida sobre la cual evoluciona la aplicación. Entre estas librerías de desarrollo se pueden nombrar, *Struts* para desarrollar en ambiente *Java*, *Rails* para el desarrollo de aplicaciones en lenguaje *Ruby*, *.NET* utilizado por *Microsoft* y finalmente *Zend Framework* y *Symfony* para el desarrollo en PHP

(Zend Technologies Ltd., 2010) El ***Zend Framework***, es un conjunto de herramientas o librerías (*Framework*) de código abierto para *PHP*, que implementa el patrón de Arquitectura MVC (Modelo Vista Controlador). Cada librería puede ser usada de forma independiente, lo cual evita tener que instalar el *framework* en su totalidad. Se utilizan para implementar la estructura de una aplicación y permite reutilizar el código de funcionalidades provistas.

El uso de esta librería ofrece las siguientes ventajas

- Simplicidad: No requiere de configuraciones, instalaciones o cualquier otro tipo de funcionalidad para comenzar a utilizarlo. Simplemente se obtiene el código de la o las librerías en su sitio oficial y se agrega como un directorio más en el código de la aplicación.
- Escalabilidad: por haber sido desarrollado orientado a objetos, cualquier actualización que se le haga a la librería le agregará nuevas funcionalidades o corregirá fallas sin tener que modificar el código que las utiliza.
- Implementa el Modelo-Vista-Controlador: posee las herramientas necesarias para desarrollar una aplicación utilizando el Modelo-Vista-Controlador
- Indican ellos en su web oficial que 4 de cada 5 funcionalidades que a diario se necesitan implementar en un desarrollo han sido tomadas en cuenta por lo que permite al programador desarrollar con mayor flexibilidad y rapidez.
- Facilita la organización del código si se siguen los parámetros básicos del Modelo-Vista-Controlador que implementan.

Sus desventajas más destacables son:

- Se debe contar con conocimientos avanzados de programación para su uso

- La documentación ofrecida por este *framework* no puede conseguirse en idioma español, lo que para muchos puede llegar a ser un impedimento
- Sumado a eso, la documentación ofrecida se basa en la explicación de algunos de los componentes con casos de uso, dejando por fuera todo lo necesario para conocer el uso de la totalidad la librería. No obstante, posee la documentación completa de la definición de su librería.
- Por ser un *framework* de desarrollo conocido y su código fuente está al acceso del público general, una aplicación desarrollada con esta herramienta es vulnerable a un ataque externo.
- La librería completa instalada en un sistema web ocupa un espacio significativo en comparación con el código utilizado para el desarrollo, debido a que ofrece un número considerable de herramientas.

Capa de Presentación

Es la capa más superficial del sistema. La responsabilidad principal es recibir las acciones y comandos de los usuarios e interpretar los resultados lógicos y transformarlos en tablas, gráficos, imágenes ó listados para su presentación al usuario.

En esta capa, que se localiza en el navegador del usuario, se utilizan tecnologías para la manipulación de objetos y elementos de la interfaz, entre las que podemos nombrar *JavaScript* y su librería de desarrollo *Jquery*, *Ajax* y *Json* son tecnologías utilizadas para la comunicación entre esta capa y la de negocio.

(Sharma, 2000) ***JavaScript*** es un lenguaje de programación que se ejecuta en los navegadores de Internet. Se encarga de la interactividad de la interfaz de usuario del sistema tales como cambios en imágenes, color de texto, menús interactivos, eventos del cursor del ratón. El lenguaje es utilizado para hacer validaciones de formularios y darle a las páginas web el dinamismo que la aplicación requiere.

(Valdelli, 2006) Algunas ventajas en el uso de esta tecnología son:

- *JavaScript* es un lenguaje robusto y seguro que permite la manipulación total de un sistema o de la información que contiene una página Web.
- Al ser interpretado, permite la ejecución de ciertas operaciones a pesar de haber ocurrido un error.
- Los usuarios pueden tener una mejor experiencia al navegar entre pantallas y realizar acciones o tareas con el sistema.
- Permite que la carga de trabajo del servidor disminuya al trasladar el procesamiento de muchas operaciones al computador del usuario.

Sus desventajas:

- Es un lenguaje que puede ser desactivado en el computador del usuario, afectando la ejecución del sistema.

- Un código desarrollado en este lenguaje debe ser descargado al computador del usuario para poder ser ejecutado, por lo que aquellos códigos de gran tamaño o pesados podrían traer lentitud y esperas injustificadas.
- Es un lenguaje cuyo código es visto por el usuario, lo que lo hace potencialmente inseguro.

Jquery Framework[2] es una biblioteca o *framework* de *JavaScript* que permite simplificar la manera de interactuar con los documentos *HTML*, permitiendo manejar eventos, desarrollar animaciones y agregar interacción con la tecnología *AJAX* a páginas *Web*. Permite reutilizar código de funcionalidades basadas en *JavaScript*.

Algunas de sus principales características:

- Simplificación y aumento de posibilidades para la selección de elementos *DOM*[3] a la hora de realizar operaciones.
- Interactividad y modificaciones del árbol *DOM*, incluyendo soporte para hojas de estilo en cascada *CSS 1-3*.
- Manejo de Eventos.
- Manipulación de la hoja de estilos *CSS*.
- Efectos y animaciones.
- Simplificación de funciones *AJAX*.
- Soporta extensiones.
- Utilidades varias como obtener información del navegador, operar con Objetos y Arreglos, funciones complementarias al *JavaScript*, etc.
- Compatible con navegadores *Firefox 1.5+, Internet Explorer 6+, Safari 2.0.2+ y Opera 9+*, entre otros.

Las ventajas al usar esta tecnología:

- Extensa Documentación: de igual manera que otras librerías de trabajo de *JavaScript*, *Jquery* ofrece una nutrida documentación, pero a diferencia de ellas, incluye ejemplos y demostraciones.
- Simplificación de código: una de las principales ventajas de esta librería es la de reducir la cantidad de líneas de código necesarias para la ejecución de una acción. Esto porque en *JavaScript* son necesarias varias líneas para definir objetos, seleccionarlos, y operar con ellos por separado, *Jquery* entre permite seleccionar y operar con un elemento u objeto de manera más directa acortando los formalismos y encadenando las operaciones.
- Desarrollos de extensiones por terceros: en la documentación de esta librería, se incluyen extensiones desarrolladas por terceros, de libre acceso y de fácil uso, las cuales agregan funcionalidades para el desarrollo del sistema.

2 Obtenido de: http://es.wikipedia.org/wiki/jquery Mayo 2010

3 *Document Model Object* (Modelo de Objetos del Documento) es una interfaz de programación que permite el acceso y modificación del contenido, estructura y estilo de los documentos *HTML* y *XML*.

- Interfaz de usuario: los desarrolladores de *Jquery* han lanzado recientemente una extensión que permite el uso de herramientas para la interacción entre los usuarios y el sistema, de manera tal que se puede hacer uso de herramientas como el ratón, elementos no soportados por el navegador nativo para solicitarle información al usuario y efectos animados, lo que permite incluir nuevas características al desarrollo de un sistema haciéndolo atractivo y novedoso.

Algunas desventajas:

- Es un sistema que no ofrece documentación en español, lo que puede llegar a ser un impedimento para su uso y comprensión.
- Por ser una librería de acceso público y conocido, una aplicación desarrollada en esta tecnología podría ser la herramienta de un ataque de seguridad externo a la aplicación.
- Si en algún momento es abandonado este proyecto, no se podrán recibir más actualizaciones o soporte
- Es un código que depende exclusivamente de la operatividad del *JavaScript* si este es desactivado, la librería no funciona.
- Debe ser descargada en conjunto con el resto del código para su funcionamiento, lo que agrega tiempo de carga de la aplicación
- El uso excesivo de sus beneficios, pueden incidir en la ejecución de las funciones del sistema.

(Charte Ojeda, 2007) ***Asyncrhonous JavaScript And XML (AJAX)*** Es la implementación y uso del objeto XMLHttpRequest, un objeto de la interfaz del lenguaje JavaScript, empleada para realizar peticiones HTTP y HTTPS a servidores Web sin que se actualice completamente la página ya cargada. Para los datos a transferir en ambas direcciones, cliente al servidor o viceversa, se usa cualquier codificación basada en texto, incluyendo: texto plano, XML, JSON, HTML y codificaciones particulares específicas.

El uso más popular, si bien no el único, de esta interfaz es proporcionar contenido dinámico y actualizaciones asíncronas en páginas *Web* mediante tecnologías construidas sobre ella como por ejemplo *JavaScript*.

Las ventajas del uso de *AJAX* en la programación de una aplicación Web:

- No es necesario instalar una aplicación *AJAX*, basta con introducir el *URL* de la aplicación que se va a utilizar en un navegador para obtener su interfaz y poder comenzar a trabajar.
- El usuario siempre accede a la última versión de la aplicación, basta con actualizarla en el servidor *Web* para que todos los clientes tengan esa actualización, sin instalaciones individuales.
- Una aplicación *AJAX* precisa muchos menos recursos, espacio en disco y memoria, que un programa nativo del sistema operativo para hacer el mismo trabajo.

- Cuando es preciso obtener información del servidor, una aplicación *AJAX* solicita únicamente los datos que necesita y no una nueva página completa, resultando más ágil que una aplicación Web clásica.
- La funcionalidad de una aplicación *AJAX* es similar a la de una aplicación *Web* que utilice *applets Java* o *Flash*, pero sin necesidad de instalar complementos ni descargar programas hasta el cliente.
- La forma en que el usuario interactúa con una aplicación *AJAX* resulta mucho más fluida, más cercana a la de un programa nativo, que la que puede ofrecer una aplicación *Web* clásica.
- Para responder a las acciones del usuario una aplicación *AJAX* actualiza partes de la página que actúa como interfaz, frente a la actualización completa de la página con el consiguiente parpadeo de las aplicaciones *Web* clásicas.

Algunas desventajas con respecto a páginas Web clásica

- El navegador notifica al usuario, visualmente y de manera automática, cuándo una aplicación clásica está esperando la recepción de una nueva página. Esa notificación no tiene lugar para las aplicaciones *AJAX*.
- Un usuario puede avanzar y retroceder por las páginas de una aplicación *Web* clásica, una posibilidad que no se tiene automáticamente con *AJAX* puesto que no hay una integración con los botones de navegación.
- Para que una aplicación *AJAX* funcione es imprescindible que el cliente, el navegador, permita la ejecución de código *JavaScript*. Si el usuario ha desactivado el motor *JavaScript* del navegador las aplicaciones *AJAX* no funcionarán.
- Las aplicaciones *Web* clásicas son relativamente más fáciles de desarrollar que las aplicaciones *AJAX*, al no tener que ejecutar una lógica compleja en el cliente, usar *DOM*, *XML*, *JSON*, etc.

JavaScript Object Notation (JSON)[4] es un formato ligero para el intercambio de datos. *JSON* es un subconjunto de la notación literal de objetos de *JavaScript* que no requiere el uso de *XML*.

La simplicidad de *JSON* ha dado lugar a la generalización de su uso, especialmente como alternativa a *XML* en *AJAX*. Una de las ventajas de *JSON* sobre *XML* como formato de intercambio de datos en este contexto es que es mucho más sencillo escribir un analizador semántico de *JSON*. En *JavaScript*, *JSON* puede ser analizado trivialmente usando el procedimiento *eval(),* lo cual ha sido fundamental para la aceptación de *JSON* por parte de la comunidad de desarrolladores *AJAX*, debido a la presencia de *JavaScript* en casi cualquier navegador *Web*.

Cada vez hay más lenguajes de programación que soportan el uso de *JSON,* mediante el uso de paquetes escritos por terceras partes que facilitan su comprensión, entre los que se incluyen: *ActionScript, C, C++, ColdFusion, Common Lisp, Delphi, Java, JavaScript, ML, Objective CAML, Perl, PHP,*

[4] Obtenido de: http://www.json.org/json-es.html Mayo 2010

Python, Rebol, Ruby y *Lua.* En el caso de PHP, en su última versión fue incluida una función nativa que evalúa la información contenida en un archivo de este tipo y la traduce a objetos que el lenguaje puede manejar.

Básicamente esta Notación de Objetos *JavaScript* se compone de una serie de elementos que siguen una sintaxis y sirve para plasmar en un texto plano la información que se desea transmitir, la cual puede estar contenida en Objetos, Arreglos, y tipos de datos variados.

2.3 Metodología de Desarrollo Ágil

(Brito Acuña, 2009) Los procesos de Desarrollo Ágiles surgen para evitar los retrasos en la entrega del software, exceso de los costos presupuestados y problemas de calidad del producto. Un proceso de desarrollo ágil se centra en satisfacer as necesidades del cliente, implementando los requerimientos del cliente de forma rápida, económica y con alta calidad del producto que se desarrolla. Los procesos de desarrollo ágil generan poca documentación pues se considera el código fuente como tal. Los procesos de desarrollo tradicionales (RUP *Rational Unified Procces,* MSF *Microsoft Solution Framework, Win-Win Spiral Model, Iconix*) hacen énfasis en la planificación detallada total de todo el trabajo a realizar para posteriormente comenzar el ciclo de desarrollo del producto software. Estos procesos de desarrollo tradicionales, llamados también procesos de desarrollo pesados, se centran en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos de software, herramientas, diagramas para el modelado y documentación detallada. Esto evita la adaptabilidad rápida a los cambios que pueden surgir durante el desarrollo, ya que requiere modificar todos los diagramas y artefactos de software que están involucrados.

Los procesos de desarrollo ágiles son más adecuados o convenientes cuando los requerimientos de los clientes son cambiantes. En estas situaciones, un proceso adaptativo será mucho más efectivo que un proceso predictivo. Por otra parte, los procesos de desarrollo adaptativos también facilitan la generación rápida de prototipos y de versiones previos a la entrega final, lo cual agradará al cliente.

(Beck, y otros, 2001)Los procesos ágiles se fundamentan en los siguientes principios:

- La prioridad más alta es satisfacer al cliente con entregas rápidas y continuas del producto de software.
- Los cambios a los requerimientos son aceptados, inclusive durante el desarrollo. Un proceso ágil permite incorporar los cambios a fin de que el cliente posea ventaja competitiva.
- El software se entrega frecuentemente.
- Los clientes y los desarrolladores deben trabajar juntos diariamente hasta la finalización del proyecto.
- El producto de software se desarrolla con personas motivadas, creando un ambiente propicio que de soporte y confianza a los desarrolladores en el éxito del proyecto.
- La manera más eficiente y efectiva de transmitir la información dentro de un equipo de desarrollo es la interacción personal.

- El software que funciona es la principal medida del progreso.
- Un proceso ágil promueve el desarrollo. Los clientes, desarrolladores y usuarios deben mantener un ritmo constante de trabajo.
- Prestar continua atención a la excelencia técnica y realizar buenos diseños.
- La simplicidad permitirá abarcar la cantidad de trabajo del proyecto.
- Una buena arquitectura, los requerimientos y diseños, son producto de un equipo bien organizado.
- En intervalos de tiempo regulares, el equipo de trabajo reflexiona en cómo serian más efectivos y ajustan su comportamiento adaptativamente.

Existen varios procesos de desarrollo ágil, entre los que podemos enumerar:

- Scrum
- Programación Extrema (XP)
- Crystal
- Evolutionary Project Management (Evo)
- Feature Driven Development (FDD)
- Adaptive Software Development (ASD)
- Lean Development (LD) y Lean Software Development (LSD)
- RUP Ágil, existiendo dos variantes AUP y EUP

2.3.1 Programación Extrema XP

(Wells, 2009) La Programación Extrema *(Extreme Programming XP)* es un proceso de desarrollo de software, ágil y ligero propuesto por Kent Beck. XP está basado en los principios de simplicidad, comunicación, retroalimentación, coraje y respeto y su principal objetivo es obtener código operativo lo antes posible durante el ciclo de desarrollo.

XP se centra en trabajar estrechamente con el cliente y entregar versiones pequeñas del sistema con mucha frecuencia, idealmente cada dos semanas. En cada versión se debe desarrollar lo mínimo y de la manera más simple posible para que funcione correctamente y no agregar complejidad al código. El diseño evoluciona conjuntamente sobre la marcha, haciendo una entrega de diseño para cada una de las entregas del sistema y luego modificándolo en las siguientes versiones. En algunos casos se considera que la documentación es el código fuente.

Para desarrollar un proyecto siguiendo el proceso de desarrollo de software XP, se aplican un conjunto mínimo de reglas y prácticas agrupadas en los segmentos de planificación, gerencia, diseño, codificación y pruebas. (Jeffries, 2001) Las reglas que se aplican en cada uno de estos segmentos son:

1. **Equipo completo:** Forman parte del equipo todas las personas que tienen algo que ver con el proyecto, incluido el cliente y el responsable del proyecto.
2. **Planificación:** Se hacen las historias de usuario y se planifica en qué orden se van a hacer y las versiones. La planificación se revisa continuamente.

3. **Test del cliente:** El cliente, con la ayuda de los desarrolladores, propone sus propias pruebas para validar las versiones.
4. **Versiones pequeñas:** Las versiones deben ser lo suficientemente pequeñas como para poder hacer una cada pocas semanas. Deben ser versiones que ofrezcan algo útil al usuario final y no trozos de código que no pueda ver funcionando.
5. **Diseño simple:** Hacer siempre lo mínimo imprescindible de la forma más sencilla posible. Mantener siempre sencillo el código.
6. **Pareja de programadores:** Los programadores trabajan por parejas y se intercambian las parejas con frecuencia.
7. **Desarrollo guiado por las pruebas automáticas:** Se deben realizar programas de prueba automática y deben ejecutarse con mucha frecuencia. Cuantas más pruebas se hagan, mejor.
8. **Mejora del diseño:** Mientras se codifica, se debe mejorar el código existente si se considera conveniente con el fin de extraer funcionalidades comunes, eliminar líneas de código innecesarias y de alta complejidad.
9. **Integración continua:** Deben tenerse siempre un ejecutable del proyecto que funcione y en cuanto se tenga una nueva pequeña funcionalidad, debe recompilarse y probarse. Es un error mantener una versión congelada dos meses mientras se hacen mejoras y luego integrarlas todas de golpe. Cuando falle algo, no se sabe qué es lo que falla de todo lo que hemos metido.
10. **El código es de todos:** Cualquiera puede y debe tocar y conocer cualquier parte del código. Para eso se hacen las pruebas automáticas.
11. **Normas de codificación:** Debe haber un estilo común de codificación, de forma que parezca que ha sido realizado por una única persona.
12. **Metáforas:** Hay que buscar unas frases o nombres que definan cómo funcionan las distintas partes del programa, de forma que sólo con los nombres se pueda uno hacer una idea de qué es lo que hace cada parte del programa. Ayuda a que todos los programadores y el cliente sepan de qué se está hablando y que no haya mal entendidos.
13. **Ritmo sostenible:** Se debe trabajar a un ritmo que se pueda mantener indefinidamente. Esto quiere decir que no debe haber días muertos en que no se sabe qué hacer y que no se deben hacer un exceso de horas otros días. Al tener claro semana a semana lo que debe hacerse, hay que trabajar duro en ello para conseguir el objetivo cercano de terminar una historia de usuario o versión.

Historias de Usuario

(Cohn, 2004) Una historia de usuario es una representación de un requerimiento de software escrito en una o dos frases utilizando un lenguaje fácil de comprender para el usuario. Las historias de usuario son utilizadas en las metodologías de desarrollo ágiles para la especificación de requerimientos (acompañadas de las discusiones con los usuarios y las pruebas de validación). Cada historia de usuario debe ser limitada, esta debería poderse escribir sobre una nota adhesiva pequeña.

Las historias de usuario son una forma rápida de administrar los requerimientos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Las historias de usuario permiten responder rápidamente a los requerimientos cambiantes.

Las historias de usuario deben ser:

- **Independientes unas de otras:** De ser necesario, combinar las historias dependientes o buscar otra forma de dividir las historias de manera que resulten independientes.
- **Negociables:** La historia en si misma no es lo suficientemente explícita como para considerarse un contrato, la discusión con los usuarios debe permitir esclarecer su alcance y éste debe dejarse explícito bajo la forma de pruebas de validación.
- **Valoradas por los clientes o usuarios:** Los intereses de los clientes y de los usuarios no siempre coinciden, pero en todo caso, cada historia debe ser importante para alguno de ellos más que para el desarrollador.
- **Estimables:** Un resultado de la discusión de una historia de usuario es la estimación del tiempo que tomará completarla. Esto permite estimar el tiempo total del proyecto.
- **Pequeñas:** Las historias muy largas son difíciles de estimar e imponen restricciones sobre la planificación de un desarrollo iterativo. Generalmente se recomienda la consolidación de historias muy cortas en una sola historia.
- **Verificables:** Las historias de usuario cubren requerimientos funcionales, por lo que generalmente son verificables. Cuando sea posible, la verificación debe automatizarse, de manera que pueda ser verificada en cada entrega del proyecto.

Las historias de usuario conforman la parte central de muchas metodologías de desarrollo ágil, tales como XP; Estas definen lo que se debe construir en el proyecto de software, tienen una prioridad asociada definida por el cliente de manera de indicar cuales son las más importantes para el resultado final, serán divididas en tareas y su tiempo será estimado por los desarrolladores. Generalmente se espera que la estimación de tiempo de cada historia de usuario se sitúe entre unas 10 horas y un par de semanas. Estimaciones mayores a dos semanas son indicativos de que la historia es muy compleja y debe ser dividida en varias historias.

Al momento de implementar las historias, los desarrolladores deben tener la posibilidad de discutirlas con los clientes. El estilo sucinto de las historias podría dificultar su interpretación, podría requerir conocimientos de base sobre el modelo o podría haber cambiado desde que fue escrita.

Cada historia de usuario debe tener en algún momento pruebas de validación asociadas, lo que permitirá al desarrollador, y más tarde al cliente, verificar si la historia ha sido completada. Como no se dispone de una formulación de requerimientos precisa, la ausencia de pruebas de validación concertadas abre la posibilidad de discusiones largas y no constructivas al momento de la entrega del producto.

Si bien el estilo puede ser libre, la historia de usuario debe responder a tres preguntas: ¿Quién se beneficia?, ¿qué se quiere? y ¿cuál es el beneficio? Por ello, algunos autores (Cohn, 2004) recomiendan redactar las historias de usuario según el formato:

Como (rol) quiero (algo) para poder (beneficio).

Los beneficios obtenidos al representar un requerimiento como una historia son:

- Al ser muy corta esta representa requisitos del modelo de negocio que pueden implementarse rápidamente (días o semanas).
- Necesitan poco mantenimiento.
- Mantienen una relación cercana con el cliente.
- Permite dividir los proyectos en pequeñas entregas.
- Permite estimar fácilmente el esfuerzo de desarrollo.
- Es ideal para proyectos con requerimientos volátiles o no muy claros.

Se pueden presentar algunas limitaciones en el momento de utilizar las historias de usuario:

- Sin pruebas de validación pueden quedar abiertas a distintas interpretaciones haciendo difícil utilizarlas como base para un contrato.
- Se requiere un contacto permanente con el cliente durante el proyecto lo cual puede ser difícil o costoso.
- Podría resultar difícil escalar a proyectos grandes.
- Requiere desarrolladores muy competentes.

2.3.2 Scrum

(Palacio, 2008) *Scrum* es un proceso de desarrollo de software Ágil que basa la gestión en la adaptación continua de las circunstancias de la evolución del proyecto. Toma su nombre y principios de los estudios realizados sobre nuevas prácticas de producción por *Hirotaka Takeuchi* e *Ikujijo Nonaka* (1985) a mediados de los años 80 (Imai, y otros, 1985). Está orientado a las personas (cliente y equipo de desarrollo) antes que al proceso en si mismo y es iterativo e incremental.

En la etapa de planificación inicial, el equipo del proyecto define la estructura y un plan. El arquitecto define la visión del desarrollo en base a esa estructura y mantiene la visión durante las etapas de desarrollo permitiendo hacer cambios.

El desarrollo se inicia desde la visión general de producto, dando detalle solo a las funcionalidades que, por ser las de mayor prioridad para el negocio, se van a desarrollar en primer lugar, y pueden llevarse a cabo en un periodo de tiempo breve (entre 15 y 60 días).

Cada uno de los ciclos de desarrollo es una iteración (*sprint*) que produce el producto incrementalmente. La duración de tiempo de una iteración puede ser de 1 a 4 semanas. Estas iteraciones son la base del desarrollo ágil, y *Scrum* gestiona su evolución a través de reuniones breves de seguimiento en las que todo el equipo revisa el trabajo realizado desde la reunión anterior y el previsto hasta la reunión siguiente. Las reuniones de seguimiento de la iteración son diarias con una duración de 15 a 30 minutos.

En una lista de funcionalidades denominada *Product Backlog* se escriben las tareas identificadas por ser desarrolladas y se le asignan prioridades. Antes de cada iteración, el equipo de desarrollo actualiza la lista reasignando prioridades y tareas.

Scrum controla de forma empírica y adaptable la evolución del proyecto, con las siguientes prácticas de la gestión ágil:

Revisión de las Iteraciones: al final de cada iteración, se realiza una revisión con todas las personas implicadas en el proyecto. Este es el periodo máximo que se puede tardar en reconducir una desviación del proyecto o de las circunstancias del producto

Desarrollo incremental: En el proyecto, no se trabaja con diseños o abstracciones. El desarrollo incremental implica que al final de cada iteración se dispone de una parte del producto operativa que se puede inspeccionar y evaluar.

Desarrollo evolutivo: Como modelo ágil, es útil en entornos con incertidumbre e inestabilidad de requisitos. Intentar predecir en las fases iniciales cómo será el resultado final, y sobre dicha predicción desarrollar el diseño y la estructura del producto no es realista, porque las circunstancias obligarán a remodelarlo muchas veces.

Scrum toma a la inestabilidad como premisa; por eso el protocolo de las prácticas de trabajo que se diseñen tiene que permitir la evolución continua sin degradar la calidad de la arquitectura, que se irá generando durante el desarrollo. Con *Scrum*, el diseño y la estructura del resultado se construyen de forma evolutiva. No se considera que la descripción detallada del producto, del servicio, de la estrategia o de la arquitectura del software (según el caso) deban realizarse definitiva y completa en la primera etapa del proyecto. El desarrollo ágil no sigue un modelo de desarrollo en cascada sino un modelo en espiral.

En la aplicación de *Scrum* para software, para evitar los problemas de degradación del sistema o de la arquitectura por la evolución continua del producto se deben incluir prácticas de refactorización en las tareas de diseño y codificación.

Auto-organización: Durante el desarrollo de un proyecto surgen circunstancias impredecibles en todas las áreas y niveles. La gestión predictiva confía la responsabilidad de su resolución al gestor de proyectos. En Scrum los equipos son auto-organizados, con margen de decisión suficiente para tomar las decisiones que consideren oportunas.

Colaboración: Las prácticas y el entorno de trabajo ágiles facilitan la colaboración del equipo, que es necesario y debe basarse en la colaboración abierta entre todos según los conocimientos y capacidades de cada persona, y no según su rol o puesto

Visión general del proceso

El resultado final se construye de forma iterativa e incremental. Al comenzar cada iteración se determina qué partes se van a construir, tomando como criterios la prioridad para el negocio, y la cantidad de trabajo que se podrá abordar durante la iteración.

En la Figura 2 se muestra el listado de funcionalidades ordenado según la prioridad. En una primera iteración, se selecciona una funcionalidad y se establecen las responsabilidades y tareas a desarrollar por cada uno de los integrantes del equipo. Estas tareas pasan a desarrollo en lo que se ha denominado iteración que tiene un período de duración de 15 a 60 días. Dentro de cada iteración se realizan diferentes ciclos de trabajo diario, en los que se realizan reuniones de revisión. Una vez finalizada la iteración, se obtiene el denominado Incremento. El proceso se reinicia al volver al inventario de funcionalidades y seleccionar una nueva funcionalidad.

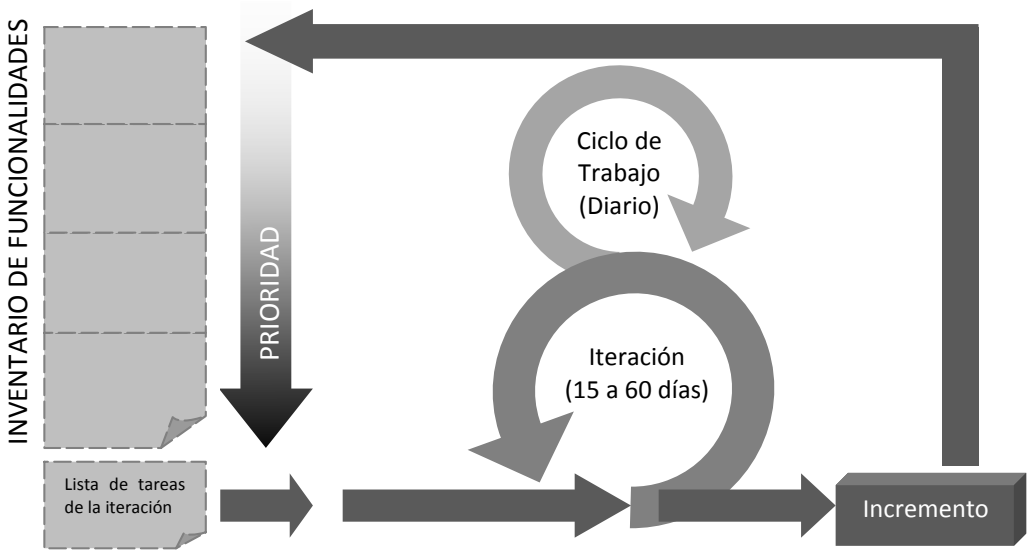

Figura 2: Fases del proceso Scrum (Palacio, 2008)

El propietario del producto: Product Manager

En el proyecto hay una persona, y sólo una, conocedora del entorno de negocio del cliente y de la visión del producto. Representa a todos los interesados en el producto final y es el responsable del Inventario de funcionalidades. Se denomina "propietario del producto" y es el responsable de obtener el resultado de mayor valor posible para los usuarios o clientes y de la financiación necesaria para el proyecto, de decidir cómo debe ser el resultado final, del lanzamiento y del retorno de la inversión. En desarrollos internos puede ser el Propietario del Producto, o responsable de Mercadeo quien asume este rol.

En desarrollos para clientes externos lo más aconsejable es que sea el responsable del proceso de adquisición del software de la empresa, que por lo general lo conforma el equipo de compras.

El Equipo: Team

Todo el equipo de desarrollo, incluido el propietario del producto conoce el proceso de desarrollo Scrum, y son los auténticos responsables del resultado. Es un equipo multidisciplinario que cubre todas las habilidades necesarias para generar el resultado. Se auto-gestiona y auto-organiza, y

dispone de atribuciones suficientes en la organización para tomar decisiones sobre cómo realizar su trabajo.

El Facilitador Scrum: Scrum Master

La organización debe garantizar el funcionamiento de los procesos y metodologías que emplea, y en este aspecto *Scrum* no es una excepción. (Schwaber, y otros, 1996) Un facilitador es responsable de garantizar el cumplimiento del proceso desarrollo. Considerando que las realidades de unas y otras empresas pueden ser muy diferentes, y que siempre que sea posible es mejor optar por adaptar las prácticas de trabajo a la empresa, y no al revés, en ocasiones puede resultar más aconsejable:

- Que en lugar de una persona con el rol de Facilitador, sean las personas y puestos más adecuados en cada organización los que reciban la formación adecuada y asuman las funciones correspondientes para cubrir esta responsabilidad.
- Que al compromiso de funcionamiento del proceso se sume también la dirección de la empresa, con el conocimiento de gestión y desarrollo ágil; y facilitando los recursos necesarios.

Facilitador Scrum designa por tanto, más que al rol, a la responsabilidad de funcionamiento del modelo. Puede ser a nivel de proyecto o a nivel de la organización; y en algunos casos resultará más apropiado un rol exclusivo (tipo *Scrum Master*) y en otros, puede ser mejor que las responsabilidades de funcionamiento las asuman los responsables del departamento de calidad o procesos, o del área de gestión de proyectos

El Inventario de Funcionalidades: Product Backlog

El *Product Backlog* es una lista que contiene las funcionalidades, mejoras, tecnología y corrección de errores que deben incorporarse al producto a través de las sucesivas iteraciones de desarrollo.

Representa todo aquello que esperan los clientes, usuarios, y en general los interesados en el producto. Todo lo que suponga un trabajo que debe realizar el equipo tiene que estar reflejado en el Inventario. Estos son algunos ejemplos de posibles elementos en el listado:

- Permitir a los usuarios la consulta de las obras publicadas por un determinado autor.
- Reducir el tiempo de instalación del programa.
- Mejorar la escalabilidad del sistema.
- Permitir la consulta de una obra a través de un API web.

A diferencia de un documento de requisitos del sistema, el Inventario de Funcionalidades nunca se da por completo; está en continuo crecimiento y evolución.

Habitualmente se comienza a elaborar con el resultado de una reunión de Lluvia de Ideas donde colabora todo el equipo partiendo de la visión del Propietario del Producto. Ésta según los casos, puede

ser una presentación informal, un informe de requisitos del departamento de marketing, etc. Es importante sin embargo disponer de una visión real, comprendida y compartida por todo el equipo.

El Inventario de Funcionalidades evolucionará de forma continua mientras el producto esté en el mercado, para darle valor de forma continua y mantenerlo útil y competitivo. Para comenzar el desarrollo se necesita una visión de los objetivos que se quieren conseguir con el producto, comprendida y conocida por todo el equipo, y elementos suficientes en el Inventario de Funcionalidades para llevar a cabo la primera Iteración.

Lista de Tareas de la Iteración: Sprint Backlog

El *sprint backlog* es la lista que descompone las funcionalidades del Inventario en las tareas necesarias para construir un incremento: una parte completa y operativa del producto.

En esta Lista de Tareas se asigna a cada tarea la persona que la va a llevar a cabo, y se estima el esfuerzo para terminarla.

Una lista de tareas de la iteración es útil porque descompone el proyecto en tareas de tamaño adecuado para determinar el progreso diario; e identificar riesgos y problemas sin necesidad de procesos complejos de gestión. Es también una herramienta de soporte para la comunicación directa del equipo.

Esta lista debe cumplir las siguientes condiciones:

- Realizado de forma conjunta por todos los miembros del equipo.
- Cubre todas las tareas identificadas por el equipo para conseguir el objetivo de la iteración.
- Sólo el equipo lo puede modificar durante la Iteración.
  El tamaño de cada tarea está en un rango de 4 a 16 horas de trabajo.
- Es visible para todo el equipo. Idealmente en una pizarra o pared en el mismo espacio físico donde trabaja el equipo.

La Iteración: Sprint

Un *Sprint* en *Scrum* es el término que denomina a una iteración que está acotada en el tiempo, usualmente entre 2 y 4 semanas, durante la cual el Equipo trabaja para convertir los elementos del Inventario de Funcionalidades en un Incremento potencialmente productivo.

La duración de la iteración debería ser lo suficientemente larga para crear algo de valor y con la suficiente calidad como para ser demostrado frente al Propietario del Producto.

En base a previas experiencias de otros autores, una iteración de 2 semanas es la duración ideal por los siguientes motivos:

- Fuerza al equipo a operar de una manera ágil.
- Brinda una retroalimentación más frecuente en lo que se está construyendo

- Reduce el riesgo de equivocar la asignación de tareas
- Incrementa la respuesta del equipo ante cambios en el negocio
- Le brinda al equipo más oportunidades de analizar y adaptarse a la forma que trabajan

El Incremento

Incremento es la parte de producto desarrollada en una Iteración. El incremento es la parte de producto producida en una Iteración, y tiene como características que está completamente terminada y operativa: en condiciones de ser entregada al cliente final. No se trata por tanto de módulos o partes a falta de pruebas, o documentación o cualquier otro producto.

Idealmente en el desarrollo ágil:

- Cada funcionalidad del Inventario se refiere a funcionalidades entregables, no a trabajos internas del tipo "diseño de la base de datos"
- Se produce un "incremento" en cada iteración.

Sin embargo suele ser una excepción habitual la primera Iteración. En el que objetivos del tipo "contrastar la plataforma y el diseño" pueden ser normales, e implican trabajos de diseño, desarrollo de prototipos para probar la plataforma que se va a emplear, entre otros.

Teniendo en cuenta esta excepción habitual, Incremento es:

Parte de producto realizada en una Iteración, y potencialmente entregable: TERMINADA Y PROBADA

Si el proyecto o el sistema requiere documentación, o procesos de validación y verificación documentados, o con niveles de independencia que implican procesos con terceros, éstos también tienen que estar realizados para considerar que el producto está "terminado".

Las Reuniones

En el trabajo con *Scrum*, el seguimiento y la gestión del proyecto se basa en la información de trabajo de las tres reuniones que forman parte del modelo:

- Planificación de la Iteración
- Monitoreo de la Iteración
- Revisión de la Iteración

Planificación de la Iteración

Es una reunión conducida por el responsable del funcionamiento de *Scrum*, donde asisten el propietario del producto, el equipo de desarrollo, y otras personas implicados en el proyecto. La reunión comienza con la presentación del propietario del producto del Inventario, en la que expone los resultados que por orden de prioridad necesita; especialmente los que se prevén que se podrán desarrollar en la siguiente Iteración. Si el Inventario de Funcionalidades ha tenido cambios significativos desde la reunión

anterior; explica también las causas que los han ocasionado. El objetivo es que todo el equipo conozca las razones y los detalles con el nivel necesario para poder estimar el trabajo necesario.

En esta reunión, tomando como base las prioridades y necesidades de negocio del cliente, se determinan cuáles y cómo van a ser las funcionalidades que se van a incorporar al producto en la próxima Iteración. Esta reunión tiene en dos objetivos fundamentales: se decide, en un período de una a cuatro horas, qué elementos del Inventario de Funcionalidades del producto se van a desarrollar y se desglosan las funcionalidades para determinar las tareas necesarias, estimar el esfuerzo que necesita cada una y asignarlas a las personas del equipo. La planificación de la iteración no debe durar más de un día.

Las características de la reunión son:

Pre-condiciones:

- La organización tiene determinados los recursos posibles para llevar a cabo la Iteración.
- El propietario del producto tiene preparado Inventario de Funcionalidades del producto: con su criterio de prioridad para el negocio, y un nº suficiente de tareas para desarrollar en la Iteración.
- Siempre que sea posible el propietario del producto debe haber trabajado ya previamente con el equipo. De esta forma su estimación previa de qué cantidad tareas de producto se puede desarrollar en la Iteración será bastante ajustada.
- El equipo tiene un conocimiento de las tecnologías empleadas, y del negocio del producto suficiente para realizar estimaciones basadas en "juicio de expertos", y para comprender los conceptos del negocio que expone el propietario del producto.

Entradas:

- El Inventario de Funcionalidades del producto
- El producto desarrollado hasta la fecha a través de los sucesivos incrementos (excepto si se trata de la primera Iteración)
- Circunstancias de las condiciones de negocio del cliente y del escenario tecnológico empleado.

Resultados:

- Listado de Tareas de la Iteración.
- Duración de la iteración y fecha de la reunión de revisión.
- Objetivo de la iteración.

Formato de la reunión

Esta reunión marca el inicio de cada Iteración. Una persona con la responsabilidad de procesos en la organización (*Facilitador Scrum*) es el responsable de su organización y gestión. Duración máxima: un día. Deben asistir el propietario del producto, el equipo y el Facilitador Scrum. También pueden asistir todas aquellas personas que puedan aportar información útil.

En una primera parte, con una duración de 1 a 4 horas, el Propietario del producto presenta el inventario de Funcionalidades del producto que tienen mayor prioridad y que estima se pueden realizar en la Iteración. La presentación se hace con un nivel de detalle suficiente para transmitir al equipo toda la información necesaria para realizar el trabajo. El equipo Realiza las preguntas y solicita las aclaraciones necesarias. Propone sugerencias, modificaciones y soluciones alternativas.

Los aportes del equipo deben suponer modificaciones en el inventario de Funcionalidades. Esta reunión es un punto importante del protocolo de *Scrum* para favorecer la lluvia de ideas en equipo y añadir valor a la visión del producto. Tras reordenar y replantear el inventario de funcionalidades del producto, el equipo define el "objetivo de la Iteración" o cuál es el valor que se le aportará al producto. Esto es una garantía de que todo el equipo comprende y comparte la finalidad del trabajo.

En la segunda parte, que puede alargarse hasta el final de la jornada el equipo desglosa cada funcionalidad en tareas, y estima el esfuerzo para desarrollar cada una de las tareas, determinando de esta forma los elementos de la Lista de Tareas de la Iteración. En este desglose el equipo tiene en cuenta los elementos de diseño y arquitectura que deberá incorporar el sistema. Los miembros del equipo se auto-asignan las diferentes tareas teniendo como criterios sus conocimientos, intereses y distribución homogénea del trabajo.

Esta segunda parte debe considerarse como una "reunión del equipo", en la que deben estar todos sus miembros y ser ellos quienes descomponen el trabajo en tareas, las asignan y estiman. El papel del propietario del producto en esta parte es atender a dudas y comprobar que el equipo comprende y comparte su objetivo. El Facilitador *Scrum* actúa como moderador de la reunión.

El Facilitador *Scrum* es responsable y garante de:

1. Que se realice esta reunión antes de cada Iteración.
2. Que antes de la reunión, el propietario del producto suministre el inventario de Funcionalidades para poder realizar la Iteración.
3. Que el diálogo principal de la reunión se realice entre el propietario del producto y el equipo. Otros asistentes pueden participar, pero su colaboración no puede implicar toma de decisiones ni limitar el diálogo principal.
4. Que la reunión sea un trabajo de colaboración activa entre los dos protagonistas: cliente y equipo, y concluyen con un acuerdo sobre el incremento de producto que van a realizar en la Iteración.
5. Que el equipo comprenda la visión y necesidades de negocio del cliente.
6. Que el equipo haya realizado una descomposición y estimación del trabajo realistas y se considere las posibles tareas necesarias de análisis, investigación o de apoyo.
7. Que al final de la reunión se tengan los listados, objetivos y duración de la lista de Tareas de la Iteración.

El *Facilitador Scrum* modera la reunión para que no dure más de un día. Debe evitar que el equipo comience a profundizar en trabajos de análisis o arquitectura que son propios de la iteración.

Monitoreo de la iteración

Reunión diaria breve, a lo sumo de 15 minutos en la que todos los miembros del equipo comunican las tareas en las que están trabajando, si se han encontrado o prevén encontrarse con algún impedimento y actualizan sobre el aquellas tareas ya terminadas o los tiempos de trabajo que les quedan.

Pre-condiciones

- Disponibilidad de un lugar físico en la organización para realizar diariamente la reunión.
- Listado de tareas actualizado en el soporte que emplee el equipo (dibujado en pizarra, sobre hoja de cálculo, etc.)
- Asiste todo el equipo
- Asiste un responsable con rol de Facilitador *Scrum* de la organización.
- Un miembro líder del equipo conduce y garantiza el protocolo, formato y tiempos de la reunión.

Entradas

- Listado de tareas de la iteración y gráfico de trabajo pendiente actualizados con la información de la reunión anterior. Información de las tareas realizadas por cada componente del equipo

Resultados

- Listado de tareas de la iteración y gráfico de trabajo pendiente actualizados. Identificación de necesidades e impedimentos.

Formato de la reunión

Se recomienda realizarla de pie y emplear un formato de lista de tareas en una pizarra o en la pared, para que todo el equipo pueda verlo, anotar o mover las tareas, junto con el gráfico de trabajo pendiente de la iteración.

En la reunión está presente todo el equipo, y pueden asistir también otras personas relacionadas con el proyecto o la organización, pero éstas últimas no pueden intervenir. Uno por uno, los miembros del equipo exponen estas tres cuestiones:

1. Tarea en la que trabajaron ayer.
2. Tarea o tareas en las que trabajarán hoy.
3. Si van a necesitar alguna cosa especial o prevén algún impedimento para realizar su trabajo. Actualizan sobre el Listado de Tareas el tiempo de trabajo que queda pendiente a las tareas en las que están trabajando, y/o marcan las que hayan podido completar.

Al final de la reunión:

- Con las estimaciones de tiempos actualizadas por el equipo, líder del equipo actualiza el Gráfico de Tareas Pendientes.

- El responsable de la gestión de procesos de la organización comienza a gestionar las posibles necesidades e impedimentos identificados.

Revisión de la Iteración

Reunión realizada al final de la iteración en la que, con una duración máxima de 4 horas, el equipo presenta al propietario del producto, clientes, usuarios, gestores, etc. el incremento construido en la Iteración.

Objetivos:

- El propietario del producto obtiene una revisión del progreso del sistema. Esta reunión le ofrece a intervalos regulares el ritmo de construcción del sistema y la trayectoria que va tomando la visión del producto.
- Al ver el incremento funcionando, el propietario del producto, y el equipo en general obtienen la retroalimentación clave para evolucionar y dar valor al Inventario de Funcionalidades.
- El responsable de procesos o calidad de la organización obtiene una retroalimentación sobre buenas prácticas y problemas durante la Iteración, necesaria para las prácticas que se empleen de ingeniería de procesos y mejora continua.
- Reunión que se realiza al final de cada Iteración en la que el equipo muestra el incremento construido, y se genera retro-información entre todos los participantes para preparar el Inventario de Funcionalidades para el inicio de la siguiente Iteración.

Pre-condiciones

- Se ha concluido la Iteración
- Asiste todo el equipo de desarrollo, el propietario del producto, el responsable de procesos de la empresa y todas las personas implicadas en el proyecto que lo deseen

Entrada

- Incremento terminado

Resultado

- Retroalimentación para el propietario del producto: hito de seguimiento de la construcción del sistema, e información para mejorar el valor de la visión del producto.
- Retroalimentación para el responsable de procesos (*Facilitador Scrum*): buenas prácticas y problemas durante la Iteración.
- Convocatoria de la reunión de la siguiente Iteración.

Formato de la reunión

Es una reunión informal. El objetivo es ver el incremento, trabajar en el entorno del cliente. Están prohibidas las presentaciones gráficas. El equipo no debe invertir más de una hora en preparar la reunión, y lo que se muestra es el resultado final: terminado, probado y operando en el entorno del cliente (incremento)

Según las características del proyecto puede incluir también documentación de usuario, o técnica. Es una reunión informativa. No tiene una misión orientada a tomar decisiones, ni a criticar el incremento. Con la información generada en la preparación de la siguiente Iteración se expondrán y tratarán las posibles modificaciones sobre la visión del producto. Un protocolo recomendado:

1. El líder del equipo el objetivo de la iteración, la lista de funcionalidades que se incluían y las que se han desarrollado.
2. El equipo hace una introducción general de la iteración y demuestra el funcionamiento de las partes construidas
3. Se abre un turno de preguntas y sugerencias sobre lo visto. Esta parte genera información muy valiosa para que el propietario del producto, y para el equipo en general puedan mejorar el valor de la visión del producto.
4. El responsable del proceso, de acuerdo con las agendas del propietario del producto y el equipo cierra la fecha para la reunión de preparación de la siguiente Iteración.

Conceptos y Métricas

Tiempo real o tiempo de trabajo.

Tiempo efectivo para realizar un trabajo. Se suele medir en horas o días.

Tiempo teórico o tiempo de tarea

Tiempo que sería necesario para realizar un trabajo en "condiciones ideales": si no se produjera ninguna interrupción, llamadas telefónicas, descansos, reuniones, etc.

Puntos de función o puntos de funcionalidad

Unidad de medida relativa para determinar la cantidad de trabajo necesaria para construir una funcionalidad o historia de usuario del Inventario de Funcionalidades.

Estimaciones

Cálculo del esfuerzo que se prevé necesario para desarrollar una funcionalidad. Las estimaciones se pueden calcular en unidades relativas (puntos de función) o en unidades absolutas (tiempo teórico)

Velocidad absoluta

Cantidad de producto construido en una Iteración. Se expresa en la misma unidad en la que se realizan las estimaciones (puntos de función, horas o días reales o teóricos).

Velocidad relativa

Cantidad de producto construido en una unidad de tiempo de trabajo.

P. ej.: puntos de función / semana de trabajo real; o horas teóricas / día de trabajo real.

Valores

Las prácticas de *Scrum* son una "carrocería" que permite trabajar con los principios ágiles, que son el motor del desarrollo. Son una ayuda para organizar a las personas y el flujo de trabajo; como lo pueden ser otras propuestas de formas de trabajo ágil: XP, Cristal, DSDM, etc.

La carrocería sin motor, sin los valores que den sentido al desarrollo ágil, no funciona.

- Delegación de atribuciones al equipo que le permita auto-organizarse y tomar las decisiones sobre el desarrollo.
- Respeto entre las personas. Los miembros del equipo deben confiar entre ellos y respetar sus conocimientos y capacidades.
- Responsabilidad y auto-disciplina (no disciplina impuesta).
- Trabajo centrado en el desarrollo de lo comprometido
- Información, transparencia y visibilidad del desarrollo del proyecto

Capítulo III - MARCO APLICATIVO

La creación de toda aplicación de software debe cumplir una serie de pasos en los que se analiza, diseña y construye los cuales se engloban en una metodología de desarrollo. Al igual que Scrum, la XP propone el desarrollo de sistemas o aplicaciones siguiendo el manifiesto de la programación ágil, el cual valora el desarrollo de software que funcione sobre la documentación exhaustiva, la colaboración con el cliente sobre la negociación de contratos, responder ante el cambio sobre el seguimiento de un plan y dar importancia a los individuos e interacciones sobre los procesos y herramientas.

Scrum y *XP* enfocan su proceso en las prácticas de organización, gestión y programación. Es por ello que se propone hacer uso de la metodología Scrum para gestionar el proceso de desarrollo del sistema utilizando los beneficios de las historias de usuario de XP (simplicidad, relación cercana con el cliente, planificación y adaptabilidad) para construir el Inventario de Funcionalidades del que parte cada una de las *Iteración*.

Para comenzar con el proceso de Scrum, se definen los roles del proyecto, se obtienen a partir del conocimiento recogido por los empleados de la empresa en los procesos internos y los formularios hasta ahora llevados en hojas de cálculo.

3.1 Definición de Roles del Proyecto

En un desarrollo de software utilizando la metodología Scrum, es necesario establecer los roles que intervienen en este proceso, por lo que a continuación se identifican cada uno de ellos aplicados al caso de estudio de este Trabajo Especial de Grado:

El Propietario del Producto: Es la alta gerencia de la empresa quien posee la necesidad de conocer el funcionamiento de su fuerza de trabajo y el desarrollo de sus operaciones asociadas con el de manera de poder tomar decisiones que dirijan el destino de su empresa.

El Facilitador Scrum: Esta conformado por el tutor y por el Tesista de este Trabajo Especial de Grado, en el caso del tutor su rol principal es el de mantener el ritmo de desarrollo del sistema, vigilar el cumplimiento de los lapsos establecidos en las estimaciones, en el caso del Tesista en garantizar la calidad de los procesos y de la comunicación del equipo de desarrollo con la empresa.

El Equipo de desarrollo: Está conformado por el Tesista de este Trabajo Especial de Grado, quien se encargó de los procesos de desarrollo la aplicación desde hacer el análisis preliminar de los requerimientos, definiciones de roles, historias de usuario, etc. Pasando por el diseño y estructuración de las interfaces, bases de datos, funcionalidades y terminando por el proceso de implementación.

Usuarios o Clientes: son los visitadores médicos, que serán el objetivo principal de la aplicación, ya que ellos serán los encargados de alimentar la información que maneja el sistema y son ellos los que se encargaron de retroalimentar los procesos de prueba.

3.2 Escribiendo el Inventario de Funcionalidades

(JASOSA, 2000) A continuación se describe el proceso utilizado para generar el guión de historias con el fin de identificar los requisitos funcionales de la aplicación. Se extrae el siguiente diálogo del contexto del dominio de la aplicación:

"La empresa quiere saber qué están haciendo sus visitadores durante un ciclo. Para ello, el visitador debe poder ingresar al sistema, planificar qué médicos va a visitar cada día antes del inicio del ciclo y luego, una vez realizada la visita, hacer el reporte con algunos indicadores establecidos y así cumplir con la cuota de visitas diarias que se le exige al visitador. El reporte se debe realizar en el período de la semana que transcurre más dos días, por lo que podrán hacer su reporte en el transcurso de la semana actual y hasta el martes siguiente. De no hacer el reporte a tiempo, se le penalizará indicando que el reporte ha sido hecho a destiempo".

Se pueden seleccionar del enunciado las posibles necesidades:

- Ingreso de usuarios (visitadores y administradores) al sistema.
- Planificación de visitas a realizar durante el ciclo.
- Reporte de una visita.
- Penalización por reporte a destiempo.

(Cohn, 2004) Se propone partir de estas necesidades y hacer uso de una plantilla para definir las historias de usuario, la cual contiene los siguientes elementos:

- **Numero de historia:** Número que identifica la Historia
- **Nombre de la Historia:** Texto corto que identifica la historia
- **Tipo de Historia:** Define el tipo de historia el cual puede catalogarse como técnica o funcional
- **Descripción:** texto descriptivo de la actividad con el siguiente enunciado: "Como un <Usuario de Sistema> quiero poder <Funcionalidad> para <Justificación>"
- **Importancia:** número que le otorga el nivel de importancia a esta historia utilizando una escala del 1 al 10, siendo 10 el más importante
- **Cómo Probarlo:** serie de pasos a seguir para verificar que la tarea planteada se ha desarrollado y cumple con el requerimiento o funcionalidad.
- **Horas de desarrollo:** número de horas en la que se estima se complete la actividad

Se utiliza la plantilla para proceder a definir cada una de las historias de usuarios identificadas durante todas las reuniones preliminares con los integrantes de la empresa, resultando en la lista total y final de funcionalidades que presentamos a continuación en la siguiente lista:

Número de Historia: 1

Nombre de la Historia: Ingreso de visitadores y administradores al sistema

Tipo de Historia: Funcional

Descripción: Como Visitador o administrador queremos poder ingresar al sistema para poder realizar nuestras actividades y mantener un registro personalizado de las mismas.

Importancia: 10

Cómo Probarlo:

- Se le solicita al usuario el nombre de usuario y la contraseña y se ingresa al sistema.
- Se verifica que los datos ingresados sean correctos, de lo contrario se indica un error.
- Se verifica que la pantalla de inicio, luego de ingresar en el sistema, sea la correspondiente al tipo de usuario.
- Se verifica que la la pantalla corresponda a la del usuario que ingreso al sistema.

Horas de Desarrollo: 24 horas

Número de Historia: 2

Nombre de la Historia: Inscripción de usuarios en el sistema

Tipo de Historia: Funcional

Descripción: Como Administrador quiero poder inscribir nuevos visitadores en el sistema para poder asignarle las actividades a desarrollar.

Importancia: 10

Cómo Probarlo:

- Se verifica que la información suministrada sea la correcta.
- Se verifica que el usuario pueda ingresar al sistema con la información suministrada.
- Se verifica que el usuario pueda realizar las operaciones asignadas.

Horas de Desarrollo: 24 horas

Número de Historia: 3

Nombre de la Historia: Gestión de médicos

Tipo de Historia: Funcional

Descripción: Como Visitador quiero poder ingresar, modificar o desincorporar de mi lista a un médico, para poder obtener la información de los médicos que voy a visitar, modificarle los indicadores y quitarlo para evitar tener en la lista médicos que no vayan a ser visitados o no requieran visita.

Importancia: 9

Cómo Probarlo:

- Se ingresa un nuevo médico al sistema verificando que no haya sido previamente ingresado.
- Se verifica que la información del médico sea correcta.
- Se verifica el listado de médicos incluido en la lista del visitador.
- Al modificar, se verifica que la nueva información ingresada sea correcta.
- Se verifica que el sistema haya registrado los cambios.
- Se verifica que los datos registrados no afecten otros indicadores.
- Se ingresan los nuevos indicadores verificando que la información sea correcta.

- Se verifica que el sistema haya registrado los cambios.
- Se verifica que al desincorporar, no aparezca el médico en la lista del visitador.

Horas de Desarrollo: 48 horas

Número de Historia: 4

Nombre de la Historia: Planificación de visitas a realizar durante el ciclo

Tipo de Historia: Funcional

Descripción: Como un Visitador quiero poder planificar las visitas que voy a realizar en mi próximo ciclo, antes de que éste comience, para poder reportarlas y cumplir con la cuota diaria de visitas exigida por mi supervisor.

Importancia: 10

Cómo Probarlo:

- Escoger el día en el que se realizará la planificación.
- Seleccionar los médicos a visitar ese día.
- Tratar de ingresar un día vacío para detectar un error.
- Verificación de que la planificación haya quedado registrada y pueda ser consultada por el usuario.
- Verificación de los indicadores de visita diaria.

Horas de Desarrollo: 48 horas

Número de Historia: 5

Nombre de la Historia: Carga de esquema de planificación de visitas anteriores

Tipo de Historia: Funcional

Descripción: Como un Visitador quiero poder cargar el esquema planificado en mi ciclo anterior para poder planificar mi próximo ciclo.

Importancia: 5

Cómo Probarlo:

- Verificar que la planificación cargada sea la que el visitador hizo en su último ciclo.

Horas de Desarrollo: 8 horas

Número de Historia: 6

Nombre de la Historia: Modificación de la planificación de visitas

Tipo de Historia: Funcional

Descripción: Como un Visitador quiero poder modificar la planificación de las visitas que voy a realizar en mi próximo ciclo, antes de que éste comience, para poder adecuar mis actividades a los diferentes eventos que puedan surgir.

Importancia: 5

Cómo Probarlo:

- Verificar que la información de planificación se recupere exactamente como la ingresó el usuario.
- Permitir la modificación de las visitas planificadas.
- Verificar que los indicadores hayan cambiado y que la información calculada sea la correcta.
- Verificar que los cambios realizados se hayan registrado exitosamente.

Horas de Desarrollo: 8 horas

Número de Historia: 7

Nombre de la Historia: Reporte de una visita

Tipo de Historia: Funcional

Descripción: Como un Visitador quiero poder reportar una visita realizada para poder indicar las actividades realizadas en ella.

Importancia: 8

Cómo Probarlo:

- Verificar que la información suministrada sea la correcta.
- Verificar que el reporte quede registrado en el sistema.
- Verificar que el status del reporte cambie a reportado.

Horas de Desarrollo: 16 horas

Número de Historia: 8

Nombre de la Historia: Reporte masivo de visitas

Tipo de Historia: Funcional

Descripción: Como un Visitador quiero poder reportar varias visitas al mismo tiempo para poder realizar la misma tarea en varios reportes que tengan la misma actividad o resultados.

Importancia: 2

Cómo Probarlo:

- Seleccionar varias visitas.
- Verificar que la información almacenada se haya guardado satisfactoriamente.

Horas de Desarrollo: 4 horas

Número de Historia: 9

Nombre de la Historia: Penalización por reporte a destiempo

Tipo de Historia: Funcional

Descripción: Como Administrador quiero conocer aquellos visitadores que no están realizando sus reportes a tiempo para poder tomar acciones y evitar retrasos en los procesos.

Importancia: 2

Cómo Probarlo:

- Se verifica que al reportar una visita fuera del tiempo reglamentario, sea marcada como reportada a destiempo.
- Se verifica que el tiempo reglamentario sea calculado exitosamente

- Se verifica que el tiempo reglamentario para hacer los reportes, sea desde el día lunes de la semana en curso hasta el martes de la siguiente semana a la media noche.

Horas de Desarrollo: 8 horas

Número de Historia: 10

Nombre de la Historia: Reprogramación de visitas

Tipo de Historia: Funcional

Descripción: Como un Visitador quiero poder reprogramar una visita que no haya podido realizar en la fecha planificada para poder cumplir con mis actividades.

Importancia: 2

Cómo Probarlo:

- Solicitando la nueva fecha en la que se realizará la visita.
- Verificando que la visita actual se quite de la fecha original.
- Verificando que la visita aparezca en la nueva fecha.

Horas de Desarrollo: 2 horas

Número de Historia: 11

Nombre de la Historia: Calendario de visitas

Tipo de Historia: Funcional

Descripción: Como un Visitador quiero poder ver la planificación de las visitas que voy a realizar mi ciclo para poder conocer cuales visitas tengo pendientes por ejecutar, cuáles han sido reportadas y cuáles han sido reportadas fuera de tiempo.

Importancia: 8

Cómo Probarlo:

- Realizando un calendario de visitas.
- Verificando que aparezcan las visitas del usuario planificadas para el ciclo actual.
- Verificando que las visitas que allí aparezcan cambien de estado al ser reportadas, reportadas a tiempo, reprogramadas, etc.

Horas de Desarrollo: 24 horas

Número de Historia: 12

Nombre de la Historia: Gestionar ciclos o actividades

Tipo de Historia: Funcional

Descripción: Como Administrador quiero poder agregar, modificar o eliminar los ciclos o actividades anuales para poder llevar el control de cada uno de los ciclos, las reuniones de gerentes y reuniones de ciclo que se aplican a las diferentes líneas del sistema.

Importancia: 10

Cómo Probarlo:

- Se indican la fecha de inicio y finalización de la actividad.
- Se verifican que las fechas sean válidas.
- Se verifica que la información suministrada se vea reflejada en el sistema.
- Se verifica que al modificar una actividad o ciclo se refleje en el sistema.

- Se verifica que al eliminar una actividad o ciclo no se eliminen aquellos que poseen planificaciones ya pasadas.
- Se verifica que los cambios o modificaciones se hagan independientes de la línea en donde se aplique.

Horas de Desarrollo: 32 horas

Número de Historia: 13

Nombre de la Historia: Gestionar líneas de producto

Tipo de Historia: Funcional

Descripción: Como Administrador quiero poder agregar, modificar y eliminar una línea de productos del sistema para poder iniciar o culminar las actividades comerciales de la misma en el sistema.

Importancia: 10

Cómo Probarlo:

- Se solicita el nombre de la línea.
- Se verifica que la misma haya sido creada.
- Se verifica que al modificar una línea se reflejen los cambios.
- Se verifica que al eliminar una línea se elimine toda la información asociada (distritos, territorios y *Bricks*).

Horas de Desarrollo: 2 horas

Número de Historia: 14

Nombre de la Historia: Gestionar distritos en las líneas de producto

Tipo de Historia: Funcional

Descripción: Como Administrador quiero poder agregar o quitar un distrito de una línea de productos en el sistema para poder indicar una región.

Importancia: 10

Cómo Probarlo:

- Se selecciona el distrito a agregar de una lista.
- Se verifica que la lista solo muestre los distritos que no estén en la Línea actualmente.
- Se verifica que se haya agregado exitosamente.
- Se verifica que al eliminar toda la información asociada a un distrito también sea eliminada (territorios y *Bricks*).

Horas de Desarrollo: 4 horas

Número de Historia: 15

Nombre de la Historia: Gestionar territorios

Tipo de Historia: Funcional

Descripción: Como Administrador quiero poder agregar o quitar un territorio en un distrito para poder indicar una zona.

Importancia: 10

Cómo Probarlo:

- Se verifica que el territorio agregado sea el número correlativo siguiente a los que ya existen en el distrito.
- Se verifica que al crearlo, quede reflejado en el sistema.
- Se verifica que al modificar un territorio se reflejen los cambios.
- Se verifica que al eliminar toda la información asociada a un territorio también sea eliminada (*Bricks*).

Horas de Desarrollo: 4 horas

Número de Historia: 16

Nombre de la Historia: Gestionar *Bricks*

Tipo de Historia: Funcional

Descripción: Como Administrador quiero poder agregar o quitar *Bricks* de un territorio para poder indicar las zonas en las que trabajará un visitador.

Importancia: 10

Cómo Probarlo:

- Se selecciona uno o varios Bricks de un listado
- Se verifica que solamente se muestren los Bricks que se encuentran definidos en el distrito y que no hayan sido agregados anteriormente al territorio
- Se verifica que los Bricks seleccionados queden reflejados en el territorio.
- Se verifica que al quitarlos, se eliminen del territorio.

Horas de Desarrollo: 4 horas

Número de Historia: 17

Nombre de la Historia: Gestionar gerentes de línea

Tipo de Historia: Funcional

Descripción: Como Administrador quiero poder seleccionar cual de los usuarios es el gerente de una línea.

Importancia: 10

Cómo Probarlo:

- Se verifica que los usuarios que aparecen en la lista de usuarios sean solamente los declarados como gerentes de línea.
- Se verifica que luego de ser seleccionado se refleje en la línea como gerente.
- Se verifica que no quede el gerente anterior en la línea.

Horas de Desarrollo: 2 horas

Número de Historia: 18

Nombre de la Historia: Gestionar gerentes de distrito

Tipo de Historia: funcional

Descripción: Como Administrador quiero poder seleccionar cual de los usuarios es el gerente de un distrito.

Importancia: 10

Cómo Probarlo:

- Se verifica que los usuarios que aparecen en la lista de usuarios sean solamente los declarados como gerentes de distrito.
- Se verifica que luego de ser seleccionado se reflejen el distrito como gerentes.
- Se verifica que no quede el gerente anterior en el distrito.

Horas de Desarrollo: 2 horas

Número de Historia: 19

Nombre de la Historia: Gestionar representantes

Tipo de Historia: Funcional

Descripción: Como Administrador quiero poder agregar o quitar un representante o visitador de un territorio para asignarle su zona de acción.

Importancia: 10

Cómo Probarlo:

- Se verifica que el listado de usuarios que aparecen en la lista de usuarios sean solamente los declarados como visitadores.
- Se verifica que luego de ser seleccionados se refleje en el territorio como el representante asignado.
- Se verifica que no quede el representante anterior o el puesto vacante.
- Se ingresa en la cuenta del usuario asignado y se verifica que el listado de médicos que posee sea el correcto y el rango de operaciones sea sobre el territorio asignado.

Horas de Desarrollo: 2 horas

Número de Historia: 20

Nombre de la Historia: Asignar índice de cobertura diaria solicitada

Tipo de Historia: Funcional

Descripción: Como Administrador quiero poder asignar el índice de cobertura diaria solicitada de visitas para que los visitadores tengan una meta que cumplir en cada ciclo

Importancia: 7

Cómo Probarlo:

- Se verifica que el campo solo acepte valores numéricos en formato decimal
- Se verifica que el valor ingresado quede reflejado en el sistema
- Se verifica que el valor ingresado permita el cálculo de los indicadores del sistema.

Horas de Desarrollo: 1 hora

Número de Historia: 21

Nombre de la Historia: Gráficos de progreso

Tipo de Historia: Funcional

Descripción: Como Administrador quiero poder tener un gráfico que muestre el progreso de las visitas que se realizan diariamente para poder contar con una herramienta que me indique en tiempo real, el progreso de las actividades del ciclo y poder tomar acciones a tiempo.

Importancia: 10

Cómo Probarlo:
- Se verifica que el cumplimiento de las actividades de los visitadores haga variar los valores calculados por el reporte.
- Se verifica que el gráfico refleje correctamente el avance de los visitadores durante el ciclo.

Horas de Desarrollo: 6 horas

Número de Historia: 22

Nombre de la Historia: Gestión general de los datos del sistema

Tipo de Historia: Funcional

Descripción: Como Administrador quiero poder tener acceso a los indicadores del sistema para poder modificar, agregar o eliminar aquellos valores que son utilizados por el sistema.

Importancia: 10

Cómo Probarlo:
- Se crean diferentes formularios de gestión del sistema para permitir la modificación, inclusión o eliminación de indicadores.
- Se verifica que cada una de las operaciones refleje los cambios hechos en los registros de las tablas de indicadores.

Horas de Desarrollo: 24 horas

Número de Historia: 23

Nombre de la Historia: Instalación del servidor de desarrollo

Tipo de Historia: Técnica

Descripción: Como equipo de desarrollo queremos tener un ambiente de aplicaciones y tecnologías para la implementación de la aplicación.

Importancia: 10

Cómo Probarlo:
- Se instala el servidor de base de datos y se comprueba su funcionamiento logrando su conexión.
- Se instala el servidor de aplicaciones, se verifica su funcionamiento al lograr establecer una conexión
- Se configura el directorio de la aplicación

Horas de Desarrollo: 2 horas

3.3 Diseño de la Aplicación

A continuación se explican los criterios utilizados para el diseño de la aplicación, partiendo de la especificación de la estructura de la base de datos implementada y la diagramación de los procesos descritos en el inventario de funcionalidades.

3.3.1 La Base de Datos

Para describir la estructura de la Base de Datos de la aplicación, se utilizan un Diagrama Entidad-Relación (E/R), para modelar los datos de un sistema, las entidades y las relaciones entre ellas. El diagrama que se muestra en la figura3 es utilizado para facilitar la conversión del diseño hacia la Base de datos.

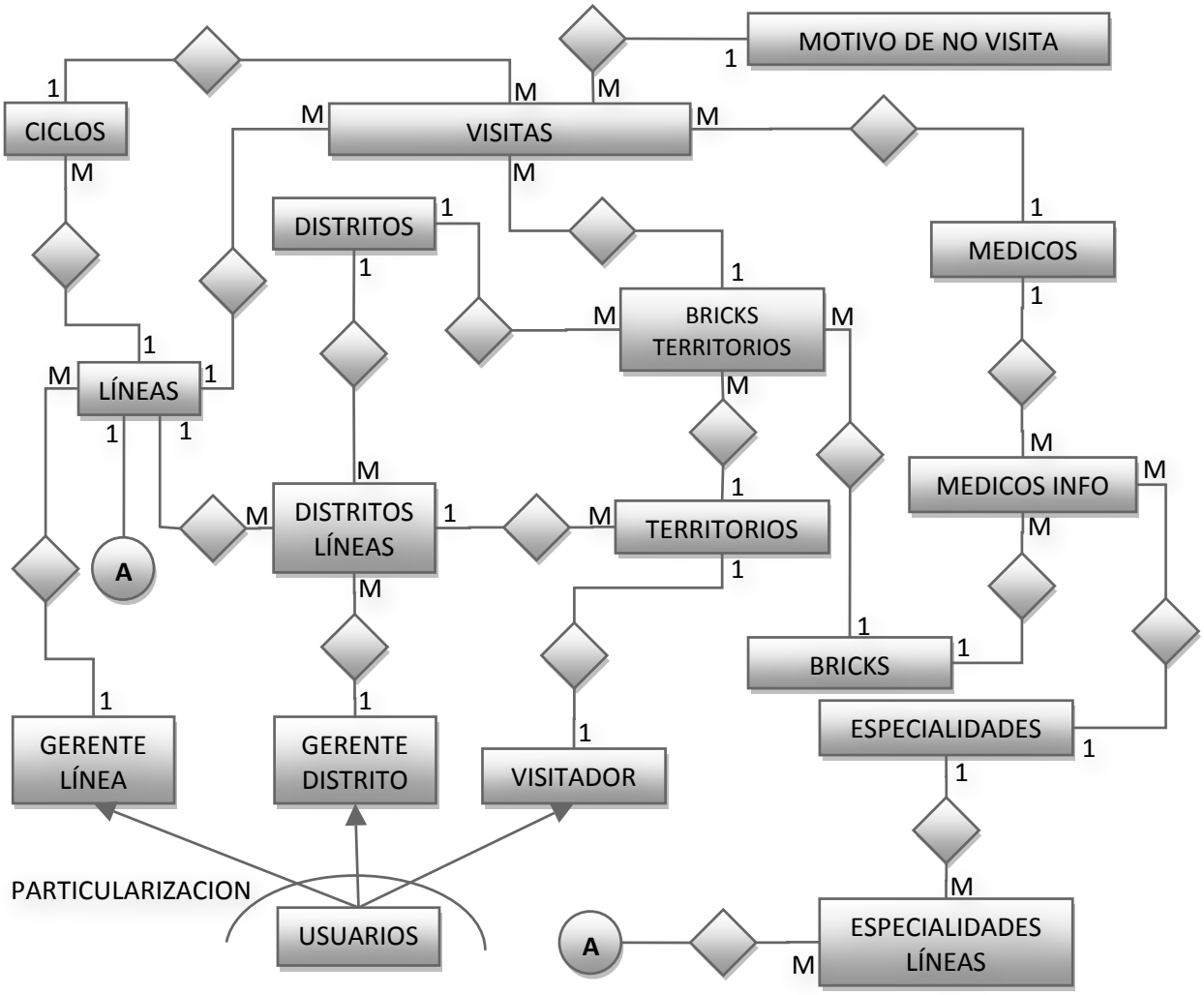

Figura 3: Modelo Entidad Relación de la Aplicación (E/R)

En la Tabla 1 se establece la transformación del modelo Entidad-Relación (E/R) de la Figura3, en donde las entidades se transforman en tablas de la Base de Datos, cada atributo se transforma en campos y sus relaciones, por ser todas de uno a muchos, son absorbidas como campos en la tabla, creándose una restricción:

Tabla 1: Definición de tablas de la Base de Datos

Tabla	Campo	Tipo de Dato	Descripción
Usuarios	**user_id**	**mediumint**	**Identificador de usuario**
	user_active	tinyint	Indica si un usuario esta activo o no
	user_name	varchar(25)	Nombre de usuario
	user_password	char(32)	Contraseña codificada con MD5
	hash	char(32)	Campo utilizado para recordar contraseña
	user_level	tinyint	Nivel de Usuario (0 = Administrador, 1 = Gerente de Línea, 2 = Gerente de Distrito, 3 = Visitador)
	user_fullname	varchar(50)	Nombres y apellidos del usuario
	user_email	varchar(100)	Correo Electrónico
	user_direction	text	Dirección completa del usuario
	user_phone	varchar(20)	Teléfono del usuario
	birth_date	date	Fecha de Nacimiento del Usuario
Líneas	**id_linea**	**tinyint**	**Identificador de línea**
	gte_linea	**mediumint**	**Usuario gerente de línea**
	nombre_linea	varchar(30)	Nombre de la línea
Distritos	**id_distrito**	**smallint**	**Identificador del Distrito**
	descripción	varchar(30)	Nombre del Distrito
Distritos_Lineas	**id_distrito**	**smallint**	**Identificador de Distrito**
	id_linea	**tinyint**	**Identificador de Línea**
	gte_distrito	**mediumint**	**Usuario Gerente de Distrito**
Territorios	**id_distrito**	**smallint**	**Identificador de Distrito**
	id_linea	**tinyint**	**Identificador de Línea**
	id_territorio	**mediumint**	**Identificador de territorio**
	id_visitador	**smallint**	**Usuario visitador**
	cobertura	float	Cobertura de visitas diarias solicitada
Bricks	**id_brick**	**smallint**	**Identificador de brick**
	id_distrito	**smallint**	**Identificador de Distrito**
	descripción	varchar(50)	Descripción del brick
Bricks_territorios	**id_distrito**	**smallint**	**Identificador de Distrito**
	id_linea	**tinyint**	**Identificador de Línea**
	id_territorio	**mediumint**	**Identificador de territorio**
	id_brick	**smallint**	**Identificador de brick**

Tabla	Campo	Tipo de Dato	Descripción
Ciclos	**id_ciclo**	**int**	**Identificador del ciclo**
	ano	int	Año del ciclo
	id_linea	tinyint	Identificador de Línea
	inicio	date	Inicio del Ciclo
	fin	date	Fin del Ciclo
	planificación	date	Fecha tope para planificar el ciclo
Médicos	**id_medico**	**bigint**	**Identificador del medico**
	nombre1	varchar(50)	Primer Nombre
	nombre2	varchar(50)	Segundo Nombre
	apellido1	varchar(50)	Primer Apellido
	apellido2	varchar(50)	Segundo Apellido
	fecha_nac	date	Fecha de Nacimiento
	correo	varchar(50)	Correo Electrónico
	móvil	varchar(50)	Teléfono Móvil
Médicos_info	**id_medico**	**bigint**	**Identificador del medico**
	id_linea	**tinyint**	**Identificador de Línea**
	id_brick	**smallint**	**Identificador de *brick***
	activo	tinyint	Indica si un usuario esta activo o no
	fecha_nac_asist	date	Fecha de Nacimiento de la asistente
	cupo	tinyint	Cantidad de visitadores que puede atender
	asistente	varchar(50)	Nombre de la asistente
	telefono1	varchar(20)	Primer Teléfono
	telefono2	varchar(20)	Segundo Teléfono
	atiende	tinyint	Preferencia de Atención 0 = Antes de los Pacientes, 1 = Durante los Pacientes, 2 = Después de los Pacientes)
	id_especialidad	smallint	Id de la Especialidad
	estado	int	Número del estado
	municipio	int	Número del municipio
	ciudad	varchar(30)	
	urbanización	varchar(30)	
	calle	varchar(30)	
	edificio	varchar(30)	
	piso	varchar(30)	
	consultorio	varchar(30)	
	referencia	text	
	adopción	tinyint	Indicador de adopción de los productos (0 = No le interesa, 1 = De 7 a 9 Visitas, 2 = De 4 a 6 Visitas, 3 = De 1 a 3 Visitas)

Tabla	Campo	Tipo de Dato	Descripción
Médicos_info	costo	tinyint	Costo de la Consulta (0 = Menos de 100, 1 = de 101 a 150, 2 = de 151 a 200, 3 = Más de 200)
	pacientes	tinyint	Cantidad de Pacientes (0 = menos de 20, 1 = de 21 a 40, 2 = de 41 a 60, 3 = más de 60)
	liderazgo	set	Banderas que identifican el liderazgo de un médico (0 = No posee ningún cargo 1 = Jefe de Departamento 2 = Sub-Jefe de Departamento 3 = Presidente de Asociación 4= Director de Hospital 5= Miembro de Sociedad 6 = Director Médico 7 = Investigador/Autor 8 = Catedrático 9 = Otro)
	zona	char(1)	Zona en la que se encuentra el consultorio (B = Zona Base, P= Zona Periferia)
	horario	text	Horario de Visita
	valor	float	Valor potencial del Médico
Especialidades	**id_especialidad**	**int**	**Identificador de la especialidad**
	descripción	varchar(50)	Nombre de la Especialidad
Especialidades_lineas	**id_especialidad**	**int**	**Identificador de la especialidad**
	id_linea	**tinyint**	**Identificador de Línea**
Actividades	id_actividad	**int**	**Identificador de la actividad**
	id_linea	**tinyint**	**Identificador de Línea**
	fechainicio	date	Inicio de la actividad
	fechafin	date	Fin de la actividad
	tipo	varchar(30)	Tipo de Actividad (Reunión de gerentes, Reunión de ciclo, Día no hábil, Feriado, convención)
	descripción	varchar(30)	Descripción de la actividad

Tabla	Campo	Tipo de Dato	Descripción
Visitas	**fecha_planificada**	**date**	**Fecha de la visita**
	id_medico	**bigint**	**Identificador del medico**
	id_distrito	**smallint**	**Identificador de Distrito**
	id_linea	**tinyint**	**Identificador de Línea**
	id_territorio	**mediumint**	**Identificador de territorio**
	id_brick	**smallint**	**Identificador de *brick***
	id_ciclo	**int**	**Identificador del ciclo**
	id_visitador	**smallint**	**Usuario visitador**
	dia_ciclo	**tinyint**	Día del ciclo en el cual se planificó la visita
	potencial_medico	float	Potencial del médico visitado
	cobertura_solicitada	float	Cobertura solicitada para esa visita
	fecha_tope_reporte	date	Día tope para reportar
	fecha_reportada	date	Fecha en la que se hizo el reporte
	fecha_reprogramada	date	Fecha en la que se reprogramó la cita
	visita_efectiva	tinyint	Indica si la visita fue efectiva o no
	id_motivo	tinyint	Motivo de la visita no efectiva
	especifique	varchar(50)	Motivo especifico
	id_acomp	mediumint	Usuario acompañante
	comentarios	text	Comentarios del reporte
	estudio_cientifico	tinyint	Entrega de estudio científico
Motivos_no_visita	**id_motivo**	**tinyint**	**Identificador del motivo**
	descripción	varchar(50)	Nombre del motivo
	abreviatura	char(3)	Abreviatura del motivo de la no visita

3.3.2 Tecnologías involucradas en el desarrollo de la Aplicación

Por ser un desarrollo basado en Web, es necesario contar con un servidor HTTP, un servidor de aplicaciones y un servidor de Base de Datos. Se decidió hacer uso de las herramientas Apache Web Server como servidor *HTTP*, *PHP Hypertext Processor* como servidor de aplicaciones en la capa de negocio y *MySql Database Server* como servidor de Base de Datos en la capa de persistencia. Los tres servicios han sido instalados en el servidor de desarrollo utilizando un paquete que se conoce como *MAMP*.

El ambiente de desarrollo de la aplicación escogido es el *Zend Framework (ZF)* en su versión 1.7. Para hacer uso de este ambiente es necesario obtener los archivos del código fuente de la librería en la Sitio *Web* Oficial y copiarlos en un directorio dentro de la estructura del servidor *HTTP*. No es necesaria la ejecución de algún proceso que instale algún servicio, ya que esta librería es el conjunto de muchas clases que solamente van a ser incluidas en el programa a desarrollar.

Para la programación de las interfaces de usuario para la capa de presentación, se hace uso de la librería *Jquery* y de su código para interfaces llamado *Jquery UI (Jquery User Interface).* De igual modo

que con el *Zend Framework*, esta librería es obtenida directamente en su sitio *Web* oficial y sus archivos posteriormente son copiados en uno de los directorios que a continuación describiremos.

Esta selección de tecnologías y servicios, se basa en los siguientes beneficios:

- Las plataformas conocidas por la colectividad como LAMP, WAMP o MAMP[5], abarcan más de dos tercios de los servidores, bases de datos y lenguajes de programación Web a la fecha.
- *MySql, PHP, Zend Framework* y *Jquery* se rigen por el principio de código abierto haciéndolas de adquisición gratuita, robustas, de fácil instalación y configuración.
- El desarrollo de nuevas características y mejoras sugeridas por la comunidad de programadores es constante y pueden ser instaladas, configuradas y mantenidas con un mínimo esfuerzo.
- Los paquetes de desarrollo de aplicaciones *LAMP, WAMP* o *MAMP* y las librerías de desarrollo *Zend Framework* y *Jquery*, permiten a los desarrolladores hacer su trabajo sin utilizar grandes cantidades de horas en detalles administrativos.
- Los lenguajes de programación para el cliente como *JavaScript* y *AJAX* son interpretados automáticamente por los navegadores de los usuarios, por lo cual no se deben hacer instalaciones de complementos adicionales para su uso.
- El uso de la herramienta *Jquery* como apoyo para el desarrollo de funcionalidades de interfaz de usuario de la aplicación en el cliente, permite el ahorro de código, implementación de novedosas interfaces de usuario para agilizar procesos y evitar el rechazo hacia la aplicación a desarrollar.

Estructura de los directorios

La librería ZF fue concebida para albergar un directorio público, un directorio de aplicación y un directorio de librería.

- **Directorio de Librería:** contiene toda la librería ZF, en la cual se definen todas las clases y funciones que esta ofrece. Si se desea agregar una definición de clase adicional, el lugar conveniente para colocarlo es en este directorio, ya que posteriormente se va a definir un proceso que carga automáticamente el contenido localizado en este directorio. La ventaja de tener en un solo directorio la librería ZF es que en una posterior actualización, simplemente se reemplaza el directorio librería por el nuevo, separándola del programa y evitando actualizaciones o daños en el código.
- **Directorio Público:** representa el directorio raíz del servidor *HTTP*. Cuando un usuario externo intenta ingresar en la aplicación, no tendrá visibilidad ni acceso a los archivos contenidos en los directorios de aplicación y de librería. Esto garantiza que no se hagan accesos no autorizados a la aplicación intentando colocar direcciones en el navegador diferentes a las manejadas por el sistema.

[5] *LAMP (Linux, Apache Web Server, Mysql y PHP), WAMP (Windows, Apache Web Server, Mysql y PHP), MAMP (Macintosh, Apache Web Server, Mysql y PHP)*

En este directorio se colocan todos aquellos archivos que por su naturaleza deben poder ser direccionados desde el navegador del usuario, entre otros, las imágenes, archivos de estilo en cascada (CSS), archivos de código *JavaScript* incluyendo la librería *Jquery* y *Jquery User Interface*.

Los dos archivos más importantes que se encuentran en este directorio son el llamado *index.php* y *.htaccess*. El primero es el archivo de inicio del sistema, toma todas las peticiones que no tengan que ver con archivos de imágenes, *JavaScript* o CSS y las dirige al Controlador correspondiente. El segundo se encarga de la redirección de todas las peticiones al *index* a menos que no sea uno de los tipos descritos anteriormente.

La ventaja de esta redirección al archivo *index.php* es que no es necesaria la colocación de algún otro archivo en este directorio y todo se maneja desde la aplicación.

- **Directorio de Aplicación:** se encuentra la lógica del sistema organizando los archivos de la siguiente manera:
  - **Archivos de Configuración (*config*):** archivo que define los parámetros básicos de configuración, en los que se puede diferenciar bajo qué términos se realiza la conexión con el servidor de Base de Datos, cual es la ruta de los archivos en el servidor, etc. Aquí se puede diferenciar estos términos dependiendo del servidor en el que se encuentre instalada la aplicación, desarrollo o producción.
  - **Disposiciones de Interfaz *(layouts)*:** directorio que contiene la definición base de la imagen del sistema, así como todos aquellos encabezados de página e inclusiones de código. Básicamente en este directorio se coloca el encabezado y pie de página del sistema. Se pueden definir también diferentes *Layouts* dependiendo de la funcionalidad para la que se requiera, por ejemplo, un *Layout* sin estilo para las llamadas de *AJAX*.
  - **Controladores (*controllers*):** directorio en el cual se definen todos los controladores que se van a encargar de manejar las peticiones de acción en un sistema. En el caso específico de la aplicación, se definieron controladores para los ciclos, uno diferente para los territorios, otro para la planificación, entre otros.
  - **Modelos (*models*):** directorio donde se colocan las clases encargadas de interactuar con la Base de Datos, que forman parte del modelo en la estructura MVC.
  - **Vistas (*views*):** directorio donde se definen las interfaces utilizadas por los controladores.
  - **Tablas (*tables*):** archivos en donde se definen las tablas que existen en la Base de Datos y son utilizadas por los modelos para almacenar, modificar, consultar o eliminar información de ellas.
  - **Formularios (*forms*):** directorio en el cual se colocan todos los formularios que se definen en el sistema.

En la Figura 4 se muestra la estructura de los directorios descritos:

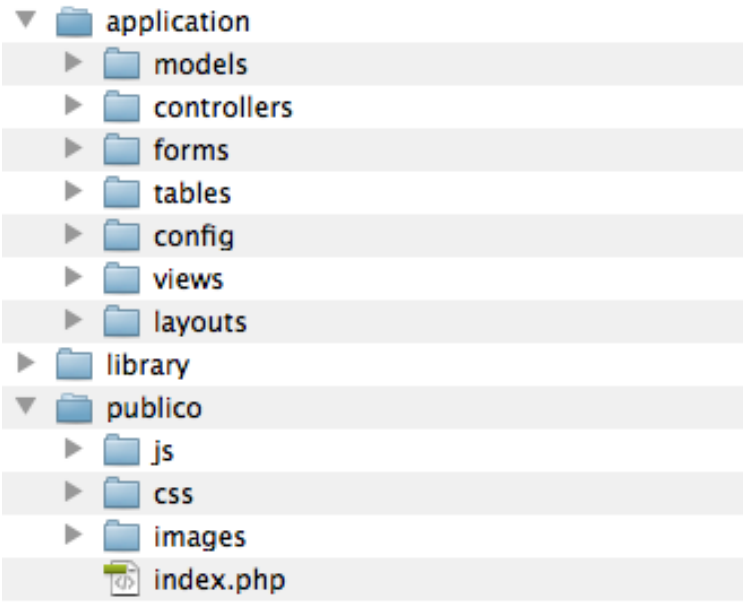

Figura 4: Estructura del Directorio de la Aplicación

Otra de las principales ventajas al organizar los directorios utilizando esta estructura es la de evitar que se acceda código no autorizado o peticiones no autorizadas, aumentando el nivel de seguridad que ofrece la aplicación.

3.3.3 Interfaces de Usuario

Uno de los principales requerimientos del cliente fue el de proveerle a sus visitadores un sistema fácil de usar y con funcionalidades novedosas pero intuitivas.

Para cumplir con este requerimiento se hizo uso de la funcionalidad de interfaz de usuario de la librería *Jquery* (*Jquery User Interface*). Con ella se pudo incorporar en las pantallas de usuario, de elementos y acciones que, dada su complejidad, hubiesen sido difíciles o complejos de programar. A continuación se presentan algunas de las pantallas ofrecidas en el sistema y las funcionalidades incorporadas gracias al uso de esta librería.

Gestión de Ciclos

En la Figura 5 se muestra la pantalla diseñada para que el Administrador del sistema pueda declarar todas las actividades a desarrollar en el año. En esta interfaz se pueden declarar el inicio y el final de un ciclo, se visualizan los días feriados, los días declarados como no hábiles por la empresa y las diferentes reuniones de ciclo o convención que se realizan en el año.

Figura 5: Pantalla de Gestión de Ciclos

Del lado izquierdo de la pantalla se muestran todos los meses del año en curso y en color se resaltan los diferentes días dependiendo de la leyenda ubicada a la derecha de la misma. Por ser una lista considerable de eventos o actividades, se ha decidido agruparlos en un elemento llamado Acordeón (ver Figura 6) que permite mostrar, a conveniencia del usuario, las listas de eventos de forma progresiva y utilizando el mismo espacio en la pantalla. Esto evita que el usuario tenga que desplazar la pantalla hacia arriba y hacia abajo para ver los eventos y su declaración en el calendario.

Se observa en la Figura 6 como en un primer estado, la sección 1 (*Section 1*) se encuentra activa y muestra su contenido, cuando un usuario procede entonces a seleccionar una nueva sección, en este caso la sección 2 (*Section 2*), se repliega la primera y se expande el contenido de la segunda. Con esto evitamos el uso de un mayor espacio y se permite agrupar el listado.

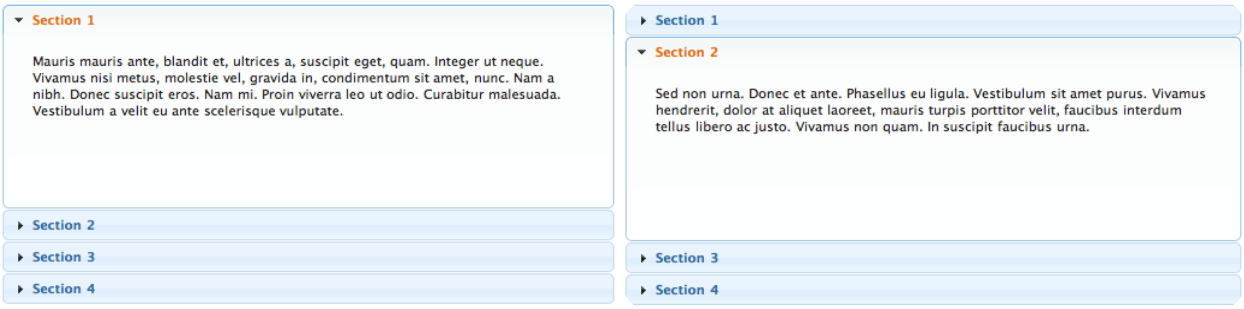

Figura 6: Elemento Acordeón[6]

En la historia de usuario número 12, se especifica que la gestión de los ciclos, reuniones o cualquier actividad puede ser independiente de la línea en la que se desea aplicar, es por ello que se decidió que la pantalla descrita anteriormente, con el calendario y las actividades se repita con la información específica para cada una de las líneas.

Nuevamente en la Figura 5 se observa que en la parte superior de la pantalla, se ha colocado lo que se conoce como Pestañas, que es otra de las funcionalidades de la librería. Estas pestañas permiten también agrupar la información desplegada para cada una de las líneas sin tener que recurrir a la programación compleja.

Del lado izquierdo de la Figura 7, se pueden visualizar las diferentes actividades programadas para la línea número 1, las cuales se encuentran resaltadas en su correspondiente pestaña con diferentes colores. Se observa que hay dos reuniones de ciclo (el 15 y del 21 al 25 de Enero), un día no hábil (el 14 de Enero) y una reunión de Gerentes (el 28 y 29 de Enero).

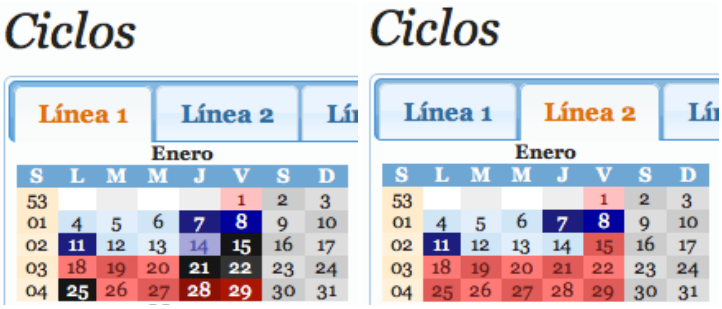

Figura 7: Elemento Pestañas

El lado derecho de la Figura 7, se muestran las actividades para la Línea 2. Se observa que en este caso no hay programación para las reuniones de ciclo, ni para día no hábil ni la reunión de gerente, sin embargo aparece el 1er ciclo y la conferencia anual del 7 al 11 de Enero.

Gestión de Territorios

En la Figura 8 se muestra la pantalla de gestión de territorios, se le ofrece al usuario administrador la posibilidad de crear, eliminar, modificar líneas, distritos y territorios; asignar

[6] http://jqueryui.com/demos/accordion/

responsables en cada nivel de división de las zonas en las que se realizan las actividades del sistema y asignar el indicador de cobertura diaria que deben cubrir los visitadores en cada ciclo.

Figura 8: Pantalla de Gestión de Territorios

Para evitar la sobrecarga de información, se ha decidido hacer un despliegue progresivo, de manera que en una primera instancia se obtenga el primer nivel en donde se encuentran las líneas, al acceder al segundo nivel, se muestran los distritos, en un tercer nivel se muestran los territorios y en un último nivel se muestran los *Bricks*. En la Figura 10, se muestra la configuración de la Línea 1, Distrito Maracaibo, Territorio 010101, en la que se despliega la información de los Gerentes y Visitadores respectivamente de cada uno de los niveles y los *Bricks* sobre los cuales tiene acción el visitador asignado.

Figura 9: Despliegue de Líneas y Distritos

La mayoría de las actividades desarrolladas por el administrador en esta pantalla son puntuales, como asignar un nuevo gerente o representante, crear una línea, asignar un distrito, etc. Es por ello que se decidió utilizar ventanas para que el administrador construya paso a paso la configuración completa de una línea, asignándole el nombre del gerente que se va a hacer cargo, los distritos donde ella actúa, los territorios, etc.

En la Figura 10 se muestra el uso de dos ventanas, a la izquierda se muestra la opción para modificar la línea y a la derecha se muestra la opción para asignar al gerente de una línea. La ventaja principal del uso de este elemento es la de evitar la recarga innecesaria de la pantalla para hacer actividades cortas y sencillas y así se puede tener de forma inmediata el resultado.

Figura 10: Uso de Ventanas en acciones

Se le ha agregado a la ventana de territorios un gráfico, como el que se muestra en la Figura 11, en donde se muestra el progreso de los reportes del ciclo actual, en el que el administrador puede monitorear el progreso individual de sus visitadores de forma granularizada. En una primera instancia, podrá ver los gráficos totales de la línea, que serán la sumatoria de todos los distritos, en un segundo nivel, podrá ver los gráficos totalizados de los territorios en el distrito y así sucesivamente en cada uno de los subniveles, hasta llegar al más detallado que es el del *Brick*.

Figura 11: Gráfico de Progreso del Ciclo

Este gráfico está dividido en cuatro sectores. Un ciclo por lo general posee 20 días hábiles, por lo que cada uno de los sectores representa cinco días del ciclo. Con esta información, el administrador podrá conocer cuál es el progreso de una línea específica durante el desarrollo del ciclo. En la parte superior del gráfico, se muestra el progreso total del ciclo. Este progreso es calculado tomando en cuenta el total de visitas planificadas para el ciclo y de ellas cuantas han sido reportadas como visitas exitosas.

Planificación de Visitas

Esta es una de las principales tareas a realizar por un visitador. En ella, cada uno de los visitadores de la empresa deberá seleccionar cuáles médicos visitará en el ciclo y en qué día del ciclo realizará dicha visita. Esta planificación irá acorde al número de visitas diarias asignadas por su supervisor. En la Figura 12 se muestra la pantalla utilizada por un visitador para seleccionar los médicos a visitar y la fecha en la que lo va a realizar.

Figura 12: Pantalla de Planificación de Ciclo

A la izquierda de la pantalla está el listado de médicos que el visitador ha agregado a los *Bricks* que le han sido asignados, a la derecha de la pantalla se encuentra separado en semanas los 20 días hábiles del Ciclo que el visitador debe planificar. El listado de médicos ofrece en su información, el número de identificación del Médico llamado MSDS, el nombre completo del médico, el número de *Brick* al cual pertenece el médico, su valor potencial y por último, el número de visitas que le han sido programadas en este ciclo.

Para agregar un médico en un día en específico de la semana en la que se desea realizar su visita, el representante haciendo uso del ratón, pulsa el botón principal del ratón sobre el médico que desea visitar y sin soltar dicho botón lo arrastra al día planificado. Este proceso trata de usar la manipulación directa de los objetos de manera de ajustarse al modelo mental del representante.

Reporte de Visitas

En la Figura 13 se muestra el proceso base de todo el sistema, el Reporte de Visitas. Esta pantalla ofrece al Visitador un calendario de los días del ciclo de operaciones actual y en cada uno de ellos se muestra el listado de médicos que fueron planificados a visitar. El día 08 de Abril se muestra resaltada la fecha actual.

Figura 13: Pantalla de Reporte de Visitas

La pantalla muestra el estatus de las visitas según se vayan ejecutando acciones sobre ellas a la hora de hacer el reporte, por ejemplo:

- El día lunes, 05 de Abril de 2010 se muestran en gris oscuro visitas reportadas dentro del tiempo establecido para ello.
- El día martes que le sigue, se muestran en rojo visitas que han sido reportadas después del tiempo establecido por los gerentes. Este tiempo vence los martes de la semana siguiente a la fecha planificada de la visita. En el caso de las visitas del 06 de Abril, el tiempo venció el martes 13 de Abril y de igual forma para todas las visitas de esa semana (05/04 al 09/04).
- Los días miércoles 07, jueves 08 y viernes 09 se muestran las visitas que han sido planificadas pero no han sido reportadas aún.
- El caso particular del médico "Yoli Vitale", programado para el día miércoles 07, se ha resaltado en color gris claro para indicar que se ha reprogramado la visita para otro día y no será contada como reportada, sin embargo queda la muestra del cambio de fecha.
- En el ejemplo se muestra la acción de reportar la visita del médico "Alexander Hernández", programado para el 08 de Abril y donde se despliega la información solicitada a los visitadores para que se realice el reporte.

3.4 Las Iteraciones

Después de un análisis del Inventario de Funcionalidades se determinó que se podía realizar la elaboración del sistema en cinco iteraciones. Cada una de ellas con un tiempo de 1 semana en la que se realizarían diversas actividades. A continuación se describen las iteraciones realizadas, las actividades, los entregables y las reuniones realizadas durante el proceso.

3.4.1 Primera Iteración: configuraciones básicas del sistema

En esta primera iteración fueron realizadas todas las funcionalidades básicas del sistema en la que se creó la estructura de directorios, la base de datos y el sistema principal de manejo de autenticación y autorización de usuarios. Las principales actividades realizadas fueron:

- Instalación del ambiente que alojará el servidor de desarrollo.
- Instalación del ambiente que aloja el servidor de pruebas.
- Creación de la tabla de usuarios del sistema.
- Configuración de la librería de desarrollo y de la estructura de directorios.
- Conexión entre el servidor de desarrollo y la Base de Datos.
- Diseño del tema principal de la aplicación.
- Implementación de la pantalla de inicio.
- Proceso de verificación de credenciales.
- Creación de la pantalla de inicio del administrador con el acceso al módulo de creación de usuarios.
- Creación de la pantalla de inicio de los visitadores, por ahora en blanco.

Al finalizar esta iteración el producto entregado cumplía con las siguientes funcionalidades:

- Creación, modificación y eliminación de cuentas de usuario del sistema
- Ingreso de un usuario autenticado en el sistema mediante la verificación de un nombre de usuario y su contraseña asociada.
- Dependiendo del perfil de usuario que ingresa al sistema, se muestra la pantalla de inicio de cada uno de ellos.

3.4.2 Segunda Iteración: Manejo de ciclos y actividades del sistema

La segunda iteración se enfocó en desarrollar lo relacionado con la configuración principal del sistema y uno de los más importantes fue el módulo de manejo de ciclos y actividades. Esto principalmente porque se deben definirse los ciclos de trabajo de los representantes para que se puedan planificar las visitas. En tal sentido, se incluyeron en esta segunda iteración las siguientes actividades:

- Creación de la pantalla principal de los ciclos que incluye el calendario del año en curso y el listado de actividades y ciclos.
- Creación de las tablas de ciclos y actividades.
- Implementación de las actividades de creación de ciclo que permite seleccionar el día de comienzo de un ciclo, día de finalización y fecha máxima para que los visitadores realicen su planificación.
- Implementación de la creación de actividad en donde se puede seleccionar el inicio y el final de la actividad. Estas actividades se pueden clasificar en: reuniones de ciclos, reuniones de gerentes, días no hábiles, días feriados y reunión anual.

- Calculo automático de los días feriados que comprenden el carnaval y la semana santa.
- Carga de la información previamente almacenada.

Los entregables de la segunda iteración comprenden las siguientes funcionalidades:

- Creación, modificación y eliminación de ciclos.
- Creación, modificación y eliminación de actividades de diferentes tipos.
- Calculo de días feriados de carnavales y semana santa.
- Capacidad de seleccionar el año en el que se aplicará la planificación.

3.4.3 Tercera Iteración: Gestión de territorios

En la reunión de planificación correspondiente a la tercera iteración, se discutieron diferentes temas acerca de los entregables de la segunda iteración y se detectó que no se había tomado en cuenta que cada una de las líneas podía tener diferencias en la aplicación de sus ciclos y actividades, debido a lineamientos en su comercialización. Por otro lado se comenzaron a desarrollar los módulos para la configuración de la división territorial de las actividades. En consecuencia se planificaron para esta iteración las siguientes actividades:

- Separación de configuración de ciclos y de actividades permitiendo la independencia de líneas.
- Creación de las tablas de líneas, distritos, territorios, *Bricks*, distritos_lineas y *bricks_territorios*.
- Proceso de creación de nuevas líneas.
- Proceso de asignación del gerente de línea.
- Proceso de asignación de distritos a la línea.
- Proceso de asignación del gerente de distrito.
- Creación de territorios.
- Proceso de asignación del visitador.
- Proceso de asignación de la cobertura diaria.
- Proceso de asignación de *Bricks* al territorio.
- Creación del despliegue progresivo de la información

Los entregables para esta iteración cubren las siguientes funcionalidades:

- La independencia del calendario para las diferentes líneas.
- Creación de nuevas líneas y territorios.
- Asignación de distritos a las líneas y *Bricks* a los territorios.
- Asignación de los distintos cargos de gerentes de línea y distrito y los representantes.
- Asignación de la carga de trabajo diaria de un representante.

- Despliegue progresivo de la información.

Después de la entrega y las pruebas realizadas a las nuevas funcionalidades del sistema, se determinó en la reunión de revisión que el despliegue progresivo funciona correctamente pero debería mantener desplegada la información previa y así evitar perder noción del territorio en el que se trabaja. No obstante, el indicador de cobertura diaria del visitador no debería ser considerado un número entero sino como un número decimal, por cuanto se debía hacer el cambio de tipo de datos en la base de datos y en la información ingresada.

3.4.4 Cuarta Iteración: Gestión de listado de médicos

En esta iteración se planificó el ingreso de los médicos al sistema y por esto se comenzó a desarrollar la pantalla de los visitadores, ya que serían los visitadores quienes cargaran los médicos al sistema. En un primer momento se determinó que cada médico podía ser visitado por un representante, por lo que se hizo una única tabla en la que se almacenaba la información asociada, pero en discusiones posteriores se determinó que un médico podía prestar consulta en diferentes clínicas u hospitales ubicados en distintas zonas por lo tanto podía recibir la visita de más de un representante de la misma o de diferentes líneas. Cada visitador debe asignarle al médico un número de calificación dependiendo de la cantidad de pacientes, influencia en su entorno, costo de su consulta y especialidad. Lo importante a destacar es que solo puede haber un médico en el sistema con diferentes calificaciones en información asociada.

Se determinaron entonces las siguientes tareas de desarrollo en esta iteración:

- Manejo de *cookies*[7] para almacenar los pasos utilizados para desplegar la información de territorios y así se toma en cuenta la consideración de la reunión de revisión de la anterior iteración.
- Creación de la tabla de médicos y médicos_info para almacenar la información
- Creación del formulario para ingresar un nuevo médico
- Creación del proceso de ingreso del médico en el que se determinará si un médico ya existe en la Base de Datos. Si un medico existe en la base de datos, se recupera su información personal como nombres, apellidos y teléfonos, entre otros. De lo contrario se crea.
- Creación del listado de médicos.

Con la entrega de este módulo se completó la implementación de la historia de usuario número 3, se realizaron jornadas de entrenamiento a un grupo reducido de visitadores para enseñarles a ingresar los médicos que ellos diariamente visitan.

[7] Una cookie (pronunciado ['ku.ki]; literalmente galleta en inglés) es un fragmento de información que se almacena en el disco duro del visitante de una página web a través de su navegador, a petición del servidor de la página. Esta información puede ser luego recuperada por el servidor en posteriores visitas. En ocasiones también se le llama "huella". Obtenido de: http://es.wikipedia.org/wiki/Cookie

Se realizaron distintas pruebas con una población reducida de representantes, específicamente con los de un distrito en específico en el que se podían presentar casos de médicos cruzados. Las pruebas desarrolladas dieron como resultado el exitoso modelado de la información.

3.4.5 Quinta Iteración: Planificación de vistas y reporte

En esta última iteración fueron planificadas las actividades principales de los representantes. Estas incluyen la planificación de visitas del ciclo y reporte. En un primer término la planificación debía incluir el listado de médicos cargados de ese visitador y los días de la semana del ciclo que deben planificar. Simplemente se deben colocar en cada día el número de visitas que se requieren y se finaliza el proceso. Se solicitó en estas reuniones que suministraran la información de la planificación del último ciclo del representante y a partir de la última planificación se pudiera planificar el próximo ciclo\.

En un segundo proceso se analizó la creación de las actividades de reporte de visitas y se establecieron distintas reglas que se aplican a este proceso para garantizar el reporte oportuno de la información.

Se incluyó en una última instancia y a petición del propietario del producto varios reportes e indicadores de progreso de visitas que incluye un listado general de médicos y un reporte detallado de las actividades de los representantes. El listado de tareas de esta iteración incluyó

- Creación de las tablas de visitas que incluye toda la información de planificación y reporte.
- Creación de la pantalla y del proceso de planificación.
- Creación de la pantalla del reporte de visitas.
- Implementación de las reglas de reporte.
- Implementación de las diferentes actividades que pueden ser aplicadas a una visita ya planificada como la de incluir visitas fuera de la planificación y reprogramación de la visita.
- Reporte de la visita
- Creación de un gráfico que indica el progreso de las visitas
- Creación de un reporte que lista los médicos del sistema y muestra la información relevante a la empresa
- Creación de un reporte detallado de las todas las actividades visitadores.

Con esta última iteración se completan las actividades ingresadas en el Inventario de Funcionalidades, por lo que se considera completado el objetivo principal de este Trabajo Especial de Grado. El entregable por tanto en esta iteración es el sistema en su totalidad.

Las pruebas hasta ahora realizadas de la aplicación, han sido aplicadas sobre un grupo reducido de visitadores y con el usuario administrador del sistema. Estas pruebas han demostrado que el modelo utilizado para representar la información y las funcionalidades del sistema cumplen con los requerimientos del cliente, sin embargo se deben aplicar estas pruebas en la totalidad de población de visitadores para medir su impacto y aceptación.

CONCLUSIONES

El objetivo general y los objetivos específicos del presente Trabajo Especial de Grado se lograron al una herramienta que permite llevar el control de las operaciones de las visitas médicas de una organización..

No se evaluó el impacto de uso que tiene la nueva aplicación en los visitadores de la organización ya que no ha sido utilizada el tiempo suficiente para determinar su aceptación. Sin embargo, la automatización de los procesos permitió la reducción de formularios y de hojas de cálculo, la obtención de los resultados inmediatos de las operaciones y el monitoreo de las actividades de los visitadores, lo cual representa una mejora significativa en las operaciones de la organización tales como:

- Facilidad en la declaración de los tiempos de ejecución de las actividades de la empresa a lo largo del año en sus diferentes líneas, lo que les permite la planificación adelantada de los ciclos y reuniones.
- Incremento de la productividad de los visitadores al tener un sistema que les permite registrar sus actividades y les evita tener que realizar formularios y enviarlos a sus directivos, focalizando sus esfuerzos en las actividades de promoción de productos.
- Disminución del tiempo de respuesta.
- Reducción en los costos de realización de los procesos.
- Centralización de la información manejada por los directivos.
- Ganancias sustanciales del tiempo invertido en la búsqueda y manipulación de la información requerida.
- Acceso directo a la información.
- Innovación e incremento de la calidad del servicio.
- Generación automatizada de reportes.

Otro punto importante a tomar en cuenta ha sido la implementación de una metodología de desarrollo de aplicaciones en la elaboración de la herramienta. Esta permitió establecer los lineamientos de desarrollo con la empresa al definir puntualmente las actividades y estimar el tiempo en el que se desarrollaría la aplicación. Este tiempo puede ser utilizado para calcular el costo horas/hombre del desarrollo del sistema y la inversión por parte de la organización. No obstante, permitió tener el control de las actividades a desarrollar y con la determinación de los entregables, se establecieron puntos de control en el tiempo que ejercían presión en la velocidad de desarrollo de las funcionalidades, estableciendo los alcances totales del proceso de desarrollo.

Esta metodología es una buena estrategia a seguir por las empresas desarrolladoras de productos de software y, en combinación con otras metodologías de desarrollo ágil, se pueden establecer reglas y hábitos de trabajo que les permitan ser competitivos.

El desarrollo de la aplicación utilizando el Modelo Vista Controlador, permitió la modularidad y organización del sistema. El beneficio que puede traer esto es la facilidad que se tiene a la hora de

desarrollar nuevos módulos o hacer actualizaciones de procesos, ya que se aísla de otros procesos que se encuentren en la aplicación. Otro beneficio del uso de MVC es que se pueden implementar los procesos definidos en este documento en nuevas plataformas simplemente con la programación de nuevas vistas sin modificar la lógica del negocio.

La librería escogida para el desarrollo de la aplicación a nivel de la capa de negocio y el lenguaje utilizado para la programación de los procesos, facilitaron su construcción. El lenguaje PHP posee una cantidad importante de documentación al alcance, que en numerosas oportunidades fue fácil de comprender y de implementar. Cabe destacar que la librería de desarrollo *Zend Framework* permitió la agilización de la programación para mucha de las necesidades, ya que varias de ellas ya se encontraban implementadas y al hacer uso de ellas, la abstracción era sencilla.

Es importante decir que todo nuevo desarrollo de una aplicación basado en tecnologías *Web* no limite sus desarrollos solamente al uso de tecnologías en la capa de negocio o en el servidor, sino que implemente nuevos desarrollos en la presentación. Todo esto porque la explotación del *JavaScript* y de su librería de desarrollo, sumado con la librería gráfica, es un buen elemento de trabajo que permite la ejecución de acciones que en otras circunstancias no hubieran sido posibles, sumado a esto, muchas de estas acciones hubieran requerido la implementación excesiva de líneas de código en su programación.

Se resalta el hecho de que a pesar de tener una buena librería para el desarrollo de interfaces novedosas y funcionales, estas siempre van a depender del navegador que posea el usuario por lo que es importante hacer las pruebas de compatibilidad necesarias para garantizar el correcto funcionamiento de la aplicación y evitar que el exceso de codificación perjudique rendimiento de los procesos y se debe garantizar que el desarrollo de nuevas funcionalidades para navegadores en dispositivos móviles cumplan con los requerimientos del cliente.

El uso de la tecnología *AJAX como* herramientas de desarrollo Web, permitió que muchas de las operaciones de la empresa en el proceso de planificación, reporte, asignación de territorios y ciclos que son procesos cortos pero repetitivos, no se hizo necesaria la recarga completa de la pantalla.

RECOMENDACIONES

Se podrían incorporar nuevas funcionalidades al sistema tales como:

- Sistema de mensajería interna en el que se soliciten y se dé respuesta a diferentes necesidades de los visitadores así como solicitar cambios en la distribución territorial y carga de trabajo
- Manejo de inventario de muestras médicas entregadas por los representantes en cada visita, para conocer con exactitud la distribución de los productos y el nivel de aceptación obtenido.
- Incorporación de nuevos usuarios y roles que controlen de forma descentralizada las operaciones hechas por el administrador y que tengan rango de acción solamente en las líneas, distritos o territorios en los que fueron asignados
- Incorporación de un sistema manejador de archivos, que le permita a los directivos que se encuentran en la oficina principal hacerles llegar a sus representantes ubicados a lo largo del territorio nacional documentos y formularios que deben utilizar en el manejo diario de sus operaciones
- La empresa de igual forma está expandiendo sus operaciones a nuevos objetivos de distribución y promoción como son las farmacias y los hospitales, por lo que se requerirá el desarrollo de un módulo para incorporarlos a la planificación.
- Se pueden desarrollar módulos para ser utilizados en dispositivos móviles que permitan la sincronización remota de los contactos médicos del sistema así como la incorporación de la planificación para que el visitador pueda revisar en cualquier momento sus actividades y hacer el reporte sin necesidad de llegar a la oficina para ello.

Por ser una herramienta de acceso remoto y con rango de acción nivel nacional, debe resultar necesario adquirir un sistema de alojamiento *Web* que permita la ejecución de la herramienta desarrollada y que facilite el acceso a ella sin la necesidad de invertir en un centro de datos propio y en una conexión a internet dedicada.

Este sistema está disponible al uso de cualquier empresa que necesite monitorear el proceso de mercadeo de sus productos. Solo sería necesario cambiar los conceptos utilizados durante el desarrollo para que sean adaptados a los productos y referencias utilizados en cada caso.

BIBLIOGRAFÍA

Achour, Mehdi; Betz, Friedhelm y Dovgal, Antony. *Manual de PHP [En línea] // PHP.net. - 28 de Mayo de 2010. - 30 de Mayo de 2010. - http://www.php.net/manual/es/.*

Barwick, Ian. *MySql Gotchas [En línea] // sql-info.de. - 25 de Mayo de 2009. - 30 de Mayo de 1020. - http://sql-info.de/mysql/gotchas.html.*

Battlez *Model View Controller [En línea] // Web Application Component Toolkit. - 06 de Octubre de 2007. - 30 de Mayo de 2010. - http://www.phpwact.org/pattern/model_view_controller.*

Beck, Kent; Beedle, Mike y Van Bennekum, Arie. *Agile Manifesto [En línea] // Manifesto for Agile Software Development. - 2001. - 30 de Mayo de 2010. - http://www.agilemanifesto.org/.*

Brito Acuña, Karenny. *Selección de Metodologías de Desarrollo para Aplicaciones Web en la Facultad de Informática de la Universidad de Cienfuegos [Informe] / Facultad de Informática Carrera de Ingeniería Informática ; Universidad de Cienfuegos "Carlos Rafael Rodríguez" . - Cienfuegos, Cuba : [s.n.], 2009.*

Buschman, Frank [y otros]. *Pattern Oriented Software Architecture. A system of Patterns [Libro]. - Inglaterra : John Wiley & Sons ltd., 1996. - ISBN: 978-04-719-5869-7.*

Cárdenas Rosado, Raúl F y Quintal García, Nancy A. *Sistemas de Apoyo a Desiciones Mercadológicas [Libro]. - Mexico : Plaza y Valdez, 2002. - ISBN 978-97-072-2163-5.*

Charte Ojeda, Francisco. *Ajax, Guía Práctica para Usuarios. [Libro]. - Madrid : ANAYA Multimedia, 2007. - ISBN 978-84-415-2134-6.*

Cohn, Michael. *Agile Software Development Book: User Stories Applied: For Agile Software Development [Libro]. - Boston : Adison-Wesley Professional, 2004. - ISBN: 978-03-212-0568-1.*

FCEA. *Workflow [En línea] // Facultad de Ciencias Económicas y de Administración. - 30 de Mayo de 2010. - http://www.ccee.edu.uy/ensenian/catsistc/docs/Workflow.pdf.*

Fernández Alarcón, Vicenç. *Desarrollo de Sistemas de Información [Libro]. - Barcelona : Ediciones UPC, 2006. - Una metodología Basada en el modelado. - ISBN 978-84-830-1862-0.*

Imai, Nonaka y Hirotaka, Takenuchi. *Managing the new Product Development Process: How Japanese Companies Learn and Unlearn [Libro]. - Boston : Harvard Business School, 1985.*

JASOSA. *DEVNETTIPS [En línea] // Definiendo Historias de Usuario en Scrum. - 10 de Marzo de 2000. - 30 de Mayo de 2010. - http://devnettips.blogspot.com/2009/03/definiendo-historias-de-usuario-en.html.*

Jeffries, Ronald E. *XProgramming [En línea] // What is Extreme Programming?. - 2001. - 30 de Mayo de 2010. - http://www.xprogramming.com/xpmag/whatisxp.htm.*

Laudon, Kenneth C. y Laudon, Jane P. *Sistemas de Información Gerencial [Libro]. - Estados Unidos : Prentice Hall/Pearson, 2008. - Décima. - ISBN 978-97-026-1191-2.*

MySql AB *Mysql 5.5 Reference Manual [En línea] // Mysql. - 2010. - 30 de Mayo de 2010. - http://dev.mysql.com/doc/refman/5.5/en/introduction.html.*

Palacio, Juan. *Flexibilidad con Scrum [En línea]. - 2008. - 30 de Mayo de 2010. - http://www.safecreative.org/work/0710210187520.*

Ribes, Xabier. *La Web 2.0. El Valor de los Metadatos y de la Inteligencia Colectiva [Publicación periódica]. - España : Revista Telos, 2007. - 73.*

Schwaber, Ken y Shutterland, Jeff. *Control Chaos [En línea] // Controlled Chaos: Living on the Edge. - 1996. - 30 de Mayo de 2010. - http://controlchaos.squarespace.com/storage/scrum-articles/Living%20on%20the%20Edge.pdf.*

Sharma, Manish. *WebDevelopersNotes.com [En línea] // Programming languages on the internet. - 2000. - 30 de Mayo de 2010. - http://www.webdevelopersnotes.com/basics/languages_on_the_internet.php3.*

Spolsky, Joel. *Joel on Software [En línea] // PHP disadvantages for enterprise applications?. - 30 de Marzo de 2006. - 30 de Mayo de 2010. - http://discuss.joelonsoftware.com/default.asp?joel.3.326588.21.*

Stewart, Celeste. *Designer's Playground [En línea] // The Advantages of PHP. - 03 de Enero de 2006. - 30 de Mayo de 2010. - http://www.designersplayground.com/articles/118/1/The-Advantages-of-PHP/Page1.html.*

The Apache Software Fundation *Apache Struts 2 Documentation [En línea] // The Apache Software Fundation. - 2010. - 30 de Mayo de 2010. - http://struts.apache.org/2.1.8.1/docs/home.html.*

Valdelli, Ilario. *HTMLPoint.com [En línea] // Aspéctos y características generales. - 2006. - 30 de Mayo de 2010. - http://www.htmlpoint.com/javascript/corso/js_02.htm.*

Wells, Don. *[En línea] // Extreme Programming: A gentle introduction. - 28 de Septiembre de 2009. - 30 de Mayo de 2010. - http://www.extremeprogramming.org/.*

Zend Technologies Ltd. *Programmer's Reference Guide [En línea] // Zend Framework. - 2010. - 30 de Mayo de 2010. - http://framework.zend.com/manual/en/learning.quickstart.intro.html.*