PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

ESCUELA DE SISTEMAS

DISERTACIÓN PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS Y COMPUTACIÓN

ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE INVENTARIO EN LA NUBE, CON NOTIFICACIONES VÍA SMS, EMAIL E INTEGRACIÓN A FACEBOOK

AUTORES:

FRANKLIN ALEXIS PAULA AGUIRRE

XAVIER ALEJANDRO RIVAS ESPINOZA

QUITO, 2014

ÍNDICE GENERAL:

ÍNDICE DE TABLAS:

ÍNDICE DE ILUSTRACIONES:

Capítulo 1 Definiciones Básicas

1.1 Objetivo del sistema a ser desarrollado

El proyecto busca demostrar la utilidad y las capacidades del uso de la tecnología con software de uso masivo, concretamente en el caso del manejo de inventario.

1.2 Inventario

Definición:

Podemos definir el inventario como el orden, documentación, clasificación que podemos dar a los bienes de cualquier clase de entidad (persona/empresa).

Todos estos bienes pueden tener variables que pueden hacer que afecte a este inventario, tales como el tiempo, etc.

Un inventario nos ayuda a llevar un mejor control de los activos de una empresa.

"El inventario se define como el registro documental de los bienes y demás cosas pertenecientes a una persona o comunidad, hecho con orden y precisión.

En una entidad o empresa, es la relación ordenada de bienes y existencias, a una fecha determinada. Contablemente, es una cuenta de activo circulante que representa el valor de las mercancías existentes en un almacén. En contabilidad, el inventario es una relación detallada de las existencias materiales comprendidas en el activo, la cual debe mostrar el número de unidades en existencia, la descripción de los artículos, los precios unitarios, el importe de cada renglón, las sumas parciales por grupos y clasificaciones y el total del inventario."[1]

1.3 Cloud Computing

Se puede empezar definiendo que todo sistema que brinda servicios a través de un servidor que se encuentra en Internet puede ser considerado Cloud Computing.

[1] http://es.wikipedia.org/wiki/Inventario - 2014-10-14

Cuando el servicio almacena archivos, información, etc., y estos pueden ser accedidos de manera remota desde cualquier lugar, debido a que están en un servidor en Internet, podemos decir que se trata de esta tecnología.

Las bondades que nos da esta tecnología es que a través del Internet somos capaces de acceder a cualquier tipo de información;además,nos ayuda a administrar mejor nuestro espacio de uso si nuestro computador estálleno;otro beneficio es que nos brinda la capacidad de tener nuestra informaciónen cualquier parte del mundo, siempre y cuando tengamos acceso a Internet.

Los modelos más importantes de esta tecnología son los SaaS, PaaS, IaaS, los cuales serán explicados a continuación:

SaaS(Software As A Service): Se brinda servicios y aplicaciones para los usuarios. Los usuarios no pueden cambiar esta aplicación pero si pueden hacer uso de ella. El usuario final no tiene ninguna clase de control sobre ellas.

PaaS(Platform As A Service): Se permite alojar y desarrollar aplicaciones al usuario final. En este caso el usuario tiene control sobre las aplicaciones que maneja.

IaaS (Infraestructures as a Service): Este servicio es aquel que brinda control sobre la infraestructura(capacidad de procesamiento, de almacenamiento y/o de comunicaciones).

Estos últimos servicios han ganado popularidad en los últimos años debido a su fácil acceso y a la seguridad que brindan. Entre los más conocidos ejemplos de estos servicios, podemos mencionar a Google Drive e iCloud.

Podríamos mencionar como un ejemplo de Cloud Computing a la administración de cualquier servicio de correo electrónico. Aquí el usuario final tiene acceso a la aplicación, en este caso, el gestor de emails, donde los administra y puede acceder a ellos desde cualquier lugar que cuente con conexión a Internet.

"La computación en la nube, concepto conocido también bajo los términos servicios en la nube, informática en la nube, nube de cómputo o nube de conceptos, del inglésCloud Computing, es un paradigma que permite ofrecer servicios de computación a través de Internet.

En este tipo de computación todo lo que puede ofrecer un sistema informático se ofrece como servicio, de modo que los usuarios puedan acceder a los servicios disponibles "en la nube de Internet" sin conocimientos (o, al menos sin ser expertos) en la gestión de los recursos que usan. Según el IEEE, es un paradigma en el que la información se almacena de manera permanente en servidores de Internet y se envía a cachés temporales de cliente, lo que incluye equipos de escritorio, centros de ocio, portátiles, etc."[2]Cloud Computing

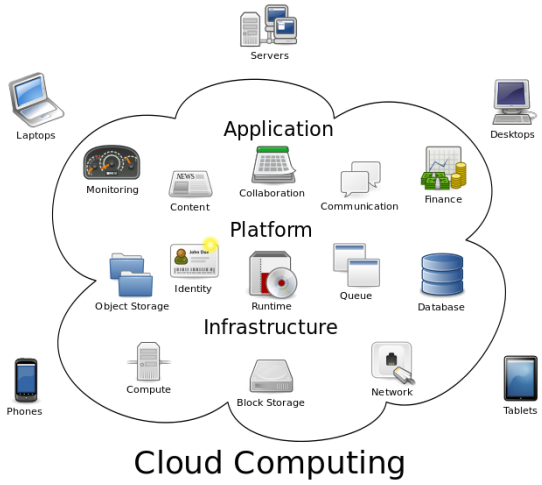

Ilustración 1 - Cloud Computing

Fuente: http://es.wikipedia.org/wiki/Computaci%C3%B3n_en_la_nube#mediaviewer/File:Cloud_computing-es.svg

[2]http://es.wikipedia.org/wiki/Computaci%C3%B3n_en_la_nube - 2014-10-14

Capítulo 2 Herramientas de Desarrollo

2.1 Lenguaje de programación PHP

2.1.1 Definición

El lenguaje de código abiertoPHP es dedicado y especializado para el desarrollo en ambiente web.

Es un lenguaje interpretado y no compilado.

Es un lenguaje ejecutado desde el lado del servidor, por lo cual no ocupael lado del cliente como Javascript, etc., o cualquier tipo de lenguaje que se ejecuta de lado del cliente. Con esto depende de la máquina del cliente el correcto funcionamiento de los sistemas y se garantiza que el resto de usuarios no sufran ningún tipo de problema relacionado con el rendimiento.

PHP es interpretado por el servidor como se mencionó antes y envía al lado del cliente solo HTML, esto quiere decir que el usuario no observa nada de la programación en PHP, solo recibe el resultado de la programación interpretada y escrita en HTML.

La curva de aprendizaje de PHP es corta si ya se tiene nociones de programación, debido a la simplicidad de su sintaxis.

Por otro lado, podemos decir que la seguridad en PHP depende netamente de la calidad de la programación, debido a que es de código abierto, cualquier persona que tenga conocimientos de programación puede entender PHP, volverse un experto y lograr acceder a páginas o sistemas vulnerables que no manejen los controles necesarios.

Este problema de seguridad no tiene por ejemplo .Net, debido a que es código cerrado, y que tiene una compañía atrás que respalda su continuo desarrollo y evolución. No se basa en una comunidad como PHP.

Lo interesante es que se puede entrar a esta comunidad, reportar errores, crear librerías y así mejorar el lenguaje.

Básicamente lo que hace es crear páginas dinámicas, o convertir páginas estáticas en dinámicas aunque va mucho más allá que eso.

PHP puede ser combinado con cualquier otra tecnología de desarrollo web, manejar varios lenguajes, varias bases de datos, etc. Generalmente PHP va de la mano con MySQL, pero también puede ser usado con otros gestores de base de datos.

"PHP (acrónimo recursivo de PHP: HypertextPreprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

Lo que distingue a PHP de algo como Javascript del lado del cliente es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabría el código subyacente que era. El servidor web puede ser incluso configurado para que procese todos los ficheros HTML con PHP, por lo que no hay manera de que los usuarios puedan saber qué se tiene debajo de la manga."[3]

2.1.2 Historia

PHP nació de un producto anterior llamado PHP/FI, su creador fue RasmusLerdorf; los primeros inicios de PHP fueron una agrupación de ficheros binariosCommon Gateway Interface (CGI).

En un principio fue usado para rastrear visitas de un currículum online, a este conjunto de scripts, su creador los llamó "Personal Home Page Tools" (PHP Tools). Al pasar el tiempo RasmusLerdorf decidió que dichas librerías debían tener mucha más funcionalidad, así que decidió reescribir PHP Tools, generando una nueva implementación con un mayor potencial.Este nuevo desarrollo generó nuevas formas de trabajar con PHP Tools, tales como interacción con bases de datos, creación de webs dinámicas, entre otras.

Rasmus volvió código abierto PHP Tools en junio de 1995, así el propio código fue mejorado y utilizado de diferentes maneras por desarrolladores, que con el paso del tiempo fue depurado y corregido.

En septiembre del mismo año RasmusLerdorf decidió ampliar PHP y abandonar el nombre PHP Tools.

Esta nueva ampliación incluía funciones básicas de PHP que todavía se encuentran vigentes hasta ahora.

[3] http://www.PHP.net/manual/es/intro-whatis.PHP - 2014-10-14

Tenía mucha similitud con el lenguaje de programación PERL, pero era un poco más sencillo. Tenía incrustaciones de HTML e interpretación de formularios.

En abril de 1996 el código fue completamente renovado, mezclando los nombres de las versiones anteriores y así nació PHP/FI.

Dicha versión tenía soporte para bases de datos DBM, MSQL, Postgres95 además de varias características tales como:

- Cookies
- Soporte de funciones definidas por el usuario, entre otras.

El mismo mes salió la versión 2.0 de PHP/FI, pero algo interesante que acotar es que solo existía una única versión completa de PHP/FI 2.0, y cuando paso de la versión beta ya se estaba reescribiendo por completo.Aunque tuvo una corta vida de desarrollo, fue bien acogida por usuarios alrededor del mundo.

Netcraftrealizó una encuesta en mayo de 1998,indicando que cerca de 60,000 dominios tenían de alguna manera PHP, cifra que ha ido aumentando con el pasar de los años.

2.1.3 Evolución

Durante el transcurso de los años PHP ha ido evolucionando desde PHP/FI para así llegar a lo que es ahora PHP y sus versiones siguientes.

RasmusLedford en 1994 desarrolló PHP empezando como un CGI realizado en Perl este a su vez interpretaba una serie de comandos limitados. A este sistema se le denominó "Personal Home Page Tools", otras personas pidieron utilizar una variedad de programas en sus páginas a Rasmus, el cual tuvo un gran éxito.

AÑO	VERSION	DESCRIPCION
1995	PHP/FI	Rasmus tuvo la idea de crear páginas dinámicas que manejen formularios, creando así etiquetas denominadas "FormInterpreters"
1996	PHP/FI 2.0	Una versión mejorada y consideradas como una herramienta más rápida y simple para el desarrollo de páginas web dinámicas

1998	PHP 3.0	ZeevSuraski y AndiGutmans incluyen dentro del analizador el soporte a nuevos protocolos de Internet y bases de datos como: MySQL y Postgres SQL, dando lugar también a Apache
2000	PHP 4.0	Se define la sintáxis y semántica dentro de dos fases anlásis y ejecución utilizando como motor Zend (desarrollado por Zeev y Andi)
2001	PHP 4.1	Aquí se llega a introducir variables globales ($_GET, $_SESSION, etc.)
2002	PHP 4.2 / PHP 4.3	Por defecto se deshabilita register_globals. Meses después se introduce CLI, en adición al CGI
2004	PHP 5.0	Se utiliza como mejora el motor Zend II, presentando mejoras y un entorno de programación orientada a objetos mucho más completo, proporcionando un alto rendimiento a aplicaciones Web empresariales
2006	PHP 5.2	Se habilita por defecto un filtro de extensiones
2007	PHP 5.2.5	Una versión con mejor estabilidad (más de 60 errores encontrados y solucionados)
Hoy día	PHP 6	Versión mejorada hasta el día actual

Tabla 1 - Evolución PHP

Creado Por: Franklin Paula / Xavier Rivas (2014-09-20)

PHP principalmente ha sido desarrollado para entorno UNIX, aquí es donde se visualiza mejor su desarrollo consiguiendo así un alto rendimiento. La tecnología Microsoft, ASP, ha sido orientada para sistemas Windows, como es el caso de NT.

2.2 Gestor de Base de DatosMYSQL

MySQL es un gestor de base de datos relacional, multi-hilo y multi-usuario, en enero de 2008 fue parte de Sun Microsystems y ésta luego paso a formar parte de Oracle Corporation desde abril de 2009. Ahora Oracle desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Por lo tanto tiene dos distribuciones, la primera bajo la licenciatura GNU GPL, pero para cualquier tipo de empresa que requiera incorporar en sus productos de forma privada, deben adquirir o comprar una licencia específica, que le permita el uso de la misma.

La mayor parte de MySQL está desarrollada en ANSI.

MySQL es patrocinado por una empresa privada que en este caso es Oracle, que tiene es propietaria de la mayoría del código. Es por esta razón que el licenciamiento tiene sus dos formas ya mencionadas.

MySQL fue fundado por David Axmark, Allan Larsson y Michael Widenius.

Características:

Característica	Descripción
escrito	C y C++
probado	amplia gama compiladores
plataformas soportadas	funciona en una cantidad múltiple de plataformas
portabilidad	GNU Automake, Autoconf, Libtool
API's disponibles	C, C++, Eiffel, Java, Perl, PHP, Python, Ruby y Tcl
manejo de procesos	multi-threaded mediante threads del kernel, múltiplesCPUs si están disponibles
Sistema de almacenamiento	transaccional y no transaccional
Tipo de tablas	la principal característica es la tabla en disco B-tree(MyISAM)
sistema reserva de memoria	basado en threads
tipo de joins	joins optimizados usando multi-joins
tablas hash	en memoria usadas como tablas temporales
código MySQL	probado con Purify, Valgrind y una herramienta GPL

servidor	disponible como un programa separado para ser usado en un ambiente cliente/servidor, también disponible como biblioteca
seguridad	privilegios, contraseñas, verificación basada en host. Todo el tráfico de contraseñas es cifrado

Tabla 2 - Características de PHP

Creado Por: Franklin Paula / Xavier Rivas (2014-09-20)

2.3 KohanaFramework

Kohanaes un framework para PHP5 que implementa el patrón de Modelo Vista Controlador Jerárquico (HMVC). Sus principales objetivos se basan en ser seguro, ligero, y fácil de utilizar.

Características:

- Extremadamente seguro
- Extremadamente ligero
- Mínima curva de aprendizaje
- Utiliza el patrón MVC y HMVC
- Compatibilidad UTF-8 100%
- Arquitectura con bajo acoplamiento
- Extremadamente sencilla de extender

Tecnología:

- PHP5orientado a objetos estricto.
- Sencilla abstracción de base de datos mediante librerías SQL
- Múltiples drivers de sesión (nativo, base de datos, y cookie)
- Un Poderoso gestor de eventos que permite pequeñas modificaciones dinámicamente
- Originalmente basado en CodeIgniter

2.4 Laravel Framework 4.1

Laraveles un framework para aplicaciones web para PHP5.4+. Utiliza una arquitectura MVC e implementa una interfaz expresiva, amigable y entendible. Además, implementa nativamente funcionalidades generalmente tediosas para el desarrollo, como son el manejo de rutas, autenticación, sesiones, caché, conexión SSH, entre otras.

Características:

- Seguridad nativa con el manejo propio de inicios de sesión y protección contra ataques de sitios cruzados (CSRF).
- Fácil escalabilidad.
- Facilidad en el manejo de distintos ambientes de desarrollo.
- Manejo de su propio lenguaje de plantillas (blade), el cual minimiza el tiempo de desarrollo y facilita la lectura del código.
- Creación de migraciones de la base de datos.
- Soporte nativo para Eloquent ORM. Esto nos garantiza mayor rendimiento y manejo más fácil de instancias de modelos de la base de datos.

2.5 Resumen Técnico

Lenguaje de Programación	PHP5
Arquitectura	MVC (modelo-vista-controlador)
Motor base de datos	MySQL 5.5
Framework a usar	Laravel
Cloud Computing	SaaS

Tabla 3 - Resumen Técnico

Creado Por: Franklin Paula / Xavier Rivas (2014-09-20)

Capítulo 3Metodologías de Desarrollo

3.1 Definición de Metodología

La metodología de desarrollo de software para este proyecto será la metodología AUP (Agile UnifiedProcess) . Esta metodología fue escogida debida a las siguientes características:

- Desarrollo Dirigido por Pruebas (test drivendevelopment - TDD).
- Modelado Ágil.
- Gestión de Cambios Ágil.
- Refactorización de Base de Datos para mejorar la productividad.

3.2 Metodologías de Desarrollo

Las metodologías de desarrollo de software son una forma o base para poder trabajar de manera estructurada, controlada y poder planificar todo el proceso de desarrollo del sistema o software.

A partir de 1960 el ciclo de vida de un software fue tomado para desarrollar sistemas grandes.

La idea original era continuar el desarrollo de dicho software para que no quede obsoleto o pierda interés después de cierto tiempo.

Con el tiempo se adaptaron técnicas de programación, capaces de desarrollar software de calidad, y criterios para poder comparar modelos de sistemas.

Las metodologías de desarrollo de software que han sido creadas e implementadas hasta el momento son las siguientes:

año	metodología
1970	programación estructurada sol
	programación estructurada Jackson
1980	Structured Systems Analysis and Design Methodology
	Structured Analysis and Design Technique
	Ingeniería de la información

1990	Rapid applicationdeveopment
	Programación orientada a objetos
	Virtual finitestate machine
	DynamicSystemsDevelopmentMethod
	Scrum
	RationalUnifiedProcess
	Extreme Programming
nuevo milenio	Enterprise UnifiedProcess
	Constructionistdesignmethodology
	Agile UnifiedProcess

Tabla 4 - Historia Metodologías de Desarrollo

Creado Por: Franklin Paula / Xavier Rivas (2014-09-20)

Cada metodología mencionada tiene su enfoque, cada enfoque se desarrolla en diferentes metodologías.

enfoque	framework	descripción
Modelo en cascada	lineal	Proceso secuencial con pasos hacia abajo empezando por análisis de requerimientos, diseño, implementación, pruebas, integración y mantenimiento.
Prototipado	iterativo	Permite ver la funcionalidad básica del sistema, sin necesidad de incluir todas las características y lógica del sistema. Con este enfoque el cliente puede evaluar de manera parcial el sistema
Incremental	lineal-iterativo	Enfocada a el control de riesgos, pensando en el desarrollo futuro. Una serie de mini-cascadas se llevan a cabo
Espiral	lineal-iterativo	Evaluación y reducción del riesgo del sistema, di viéndolo en segmentos para la facilidad de

		cambio. Los cuadrantes del espiral son los siguientes: determinar objetivos, evaluar alternativas, identificar riesgos, desarrollo y verificación resultados. Así por cada iteración. Por último se planea la siguiente iteración.
RAD	iterativo	Desarrollo iterativo y construcción de prototipos. El objetivo principal es el desarrollo rápido con calidad bajo costo. Reducción de riesgos y cumplimiento de la necesidad comercial. Se trabaja mediante prioridades del sistema, participación de los usuarios como testers y un documento para el futuro desarrollo y mantenimiento

Tabla 5 - Enfoques Metodologías de Desarrollo

Creado Por: Franklin Paula / Xavier Rivas (2014-09-20)

3.3 MVC

Es un patrón de arquitectura de software que se enfoca en organizar una aplicación de manera que la interfaz del usuario se separe de la lógica de la programación.

Este patrón es común en desarrollos de páginas web 2.0.

La interacción básica de este patrón es la siguiente:

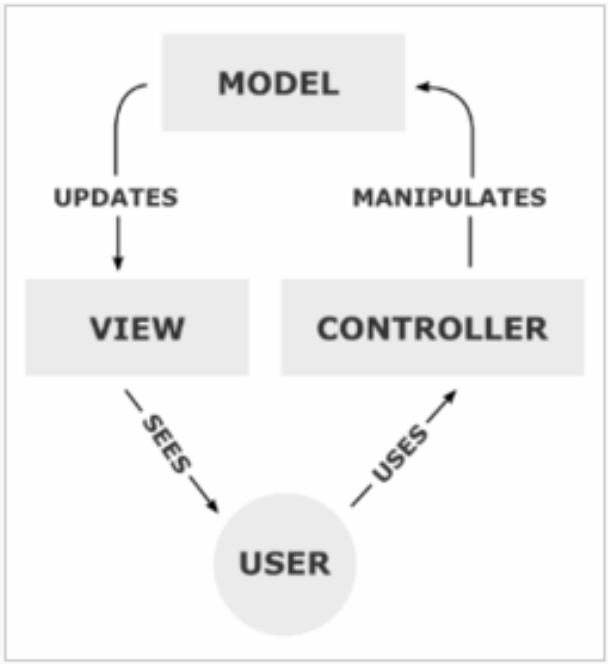

Ilustración 2 - MVC

Fuente:http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller#mediaviewer/File:MVC-Process.svg

Modelo: es toda la representación de la información, puede generar todas las opciones de interacción con la base de datos (insertar, actualizar, borrar, consultar), además de esto el modelo es capaz de dar toda las clases de controles que sean necesarios. Podemos decir que el modelo es la base para que el patrón MVC funcione de manera correcta, debido a que de aquí parten como se va a manejar los datos, la seguridad de los mismos y como va a interactuar con el usuario final.

Controlador: el controlador podemos decir que es la etapa intermedia entre el modelo y la vista. Todo lo que el usuario requiera desde la vista pasa a través del controlador y este es el encargado de llamar al modelo para poder ejecutar dicha acción.

Eso quiere decir que el controlador responde a eventos e invoca peticiones al modelo. Por otro lado el controlador es capaz de enviar datos a la vista para que puedan ser visualizados, esto quiere decir que el controlador invoca al modelo y envía dichos datos a la vista para que pueda ser visualizada por el usuario final.

Vista: presenta la información en el formato adecuado para que pueda interactuar con el usuario final.

Esto quiere decir que es el resultado final que verá el usuario, por donde el usuario podrá interactuar con el cliente y también el medio por el cual el usuario podrá generar acciones que llamen al controlador y al modelo.

En el desarrollo web es importante entender que generalmente se da la diferencia entre cliente y servidor, en la cual la mayor parte del contenido se la realiza en el servidor para que el cliente tenga una carga ligera.

Pero esto no siempre es así, a veces es necesario que en el cliente exista un desarrollo parcial para que la web sea desempeñada de una manera óptima. Estos desarrollos generalmente son realizados con frameworks de lado del cliente, que se adaptan a patrones MVC.

3.4 Análisis de Requerimientos NDT

3.4.1 Requisitos de almacenamiento de información

Los requerimientos en infraestructura van a depender del alcance que tenga el proyecto(cuantas empresas requieran usar el servicio).

En un inicio los requerimientos de almacenamiento e infraestructura son los siguientes:

- Servidor Linuxcompatible con el sistema.
- 2 núcleos procesamiento.
- Memoria RAM 2Gb.
- Capacidad de transferencia del ancho de banda 300Gbps.
- Una IP dedicada.
- 50Gb disco duro.

3.4.2 Requisitos de actores

Los actores están divididos en usuarios administradores y usuarios clientes.

Administrador:

- Aprobar usuarios creados por Facebook.
- Crear usuarios nativos del sistema.

- Administrar usuarios receptores de alertas.
- Administrar Productos.
- Administrar Distribuidores.
- Generar pedidos a distribuidores.
- Generar recepciones de pedidos a distribuidores.
- Realizar ajustes de inventarios.
- Revisar alertas.

Usuario General:

- Administrar Productos.
- Administrar Distribuidores.
- Generar pedidos a distribuidores.
- Generar recepciones de pedidos a distribuidores.
- Revisar alertas.

Ilustración 3 - Diagrama de Casos de Uso

Creado Por: Franklin Paula / Xavier Rivas (2014-09-20)

3.4.3 Requisitos funcionales

La obtención de requerimientos se baso en la metodología JAD (JointApplicationDesign). Para poder realizar dicha metodología se realizaron las siguientes etapas:

- Adaptación
- Taller: en este taller se definen los objetivos del sistema, su alcance y sus requisitos

- Conclusión: se obtiene el documento final con los requerimientos aprobados.

Realizando el taller con todos los involucrados en el sistema, en la cual se definió, se revisó y se generó los requerimientos.

Del siguiente taller se obtuvo el siguiente producto (conclusión).

	Title ID	Full Description	Code	Priority	Workload	Risk	Status
→	1.	Inicio de sesión.	REQ_0001	5		High	Approved
2	2.	Cambio de contraseña.	REQ_0002	5		High	Approved
3	3.	CRUD general para Usuarios, productos y	REQ_0003	5		High	Approved
4	4.	Clave provisional en la creación de un nuevo usuario.	REQ_0004	5		High	Approved
5	5.	Manejar procesos generación y recepción de pedidos	REQ_0005	5		High	Approved
6	6.	Generar alertas a Facebook, SMS y correo	REQ_0006	5		High	Approved
7	7.	Ajustes de inventario.	REQ_0007	5		High	Approved
8	8.	Administración de alertas y los medios por los que recibe dichas alertas.	REQ_0008	5		High	Approved

Ilustración 4-Requerimientos Aprobados, Nivel de Riesgo y Prioridad

En lo cual se concluyo que todos los requerimientos tienen un nivel de riesgo alto, debido a que son parte fundamental del sistema, además se concluyó que todos deben tener la mas alta prioridad debido a que son parte fundamental del sistema.

Por lo tanto podemos listar los siguientes requerimientos ya detallados y aprobados:

1. El sistema tendrá la capacidad de permitir al usuario iniciar sesión y reconocer si este usuario es administrador o cliente general. El inicio de sesión podrá ser realizado en el mismo sistema o con Facebook.
2. Los usuarios tendrán la capacidad de cambiar su contraseña cuando ellos lo consideren necesario, teniendo que hacerlo la primera vez que inicien sesión, ya que su primera contraseña provisional será recibida en su correo electrónico.
3. El sistema brindará la capacidad de crear, actualizar y eliminar datos de cada uno de registros para los distintos catálogos (Usuarios, Productos y Distribuidores). Esta funcionalidad estará atada a los permisos que tenga el usuario de acuerdo a su perfil.
4. Al crear un nuevo usuario, se generará automáticamente una clave provisional de ingreso al sistema.
5. El sistema permitirá manejar los procesos de generación y recepción de pedidos de productos a proveedores.

6. El sistema será el encargado de generar alertas a Facebook, SMS y correo electrónico en las siguientes situaciones:
    a. Inventario de un producto llega al mínimo parametrizado.
    b. Inventario de un producto llega a 0 (se termina el producto).
    c. Un usuario ha creado su cuenta con Facebook y necesita aprobación.
    d. Se ha creado un producto nuevo.
    e. Se ha realizado un pedido.
    f. Se ha realizado una recepción de un pedido.
7. El sistema permitirá realizar ajustes de inventario, permitiendo seleccionar un producto y escribiendo la existencia actual de dicho producto. Esta funcionalidad estará atada a los permisos que tenga el usuario de acuerdo a su perfil.
8. El sistema tendrá la capacidad de administrar los usuarios que recibirán las alertas y los medios a los cuales les llegarán (Facebook, SMS y/o correo electrónico). Esta funcionalidad estará atada a los permisos que tenga el usuario de acuerdo a su perfil.

3.4.4 Requisitos de interacción

Trabajamos con diagramas de secuencia en los cuales vamos a entender como el usuario actúa ante ciertas funcionalidades del sistema.

Veremos los diagramas de secuencia de las principales funcionalidades y del corazón del sistema que en este caso son las alertas.

3.4.4.1 Activar Usuarios (perfil administrador)

Podemos observar en este gráfico la interacción del usuario con el sistema para poder activar un usuario.

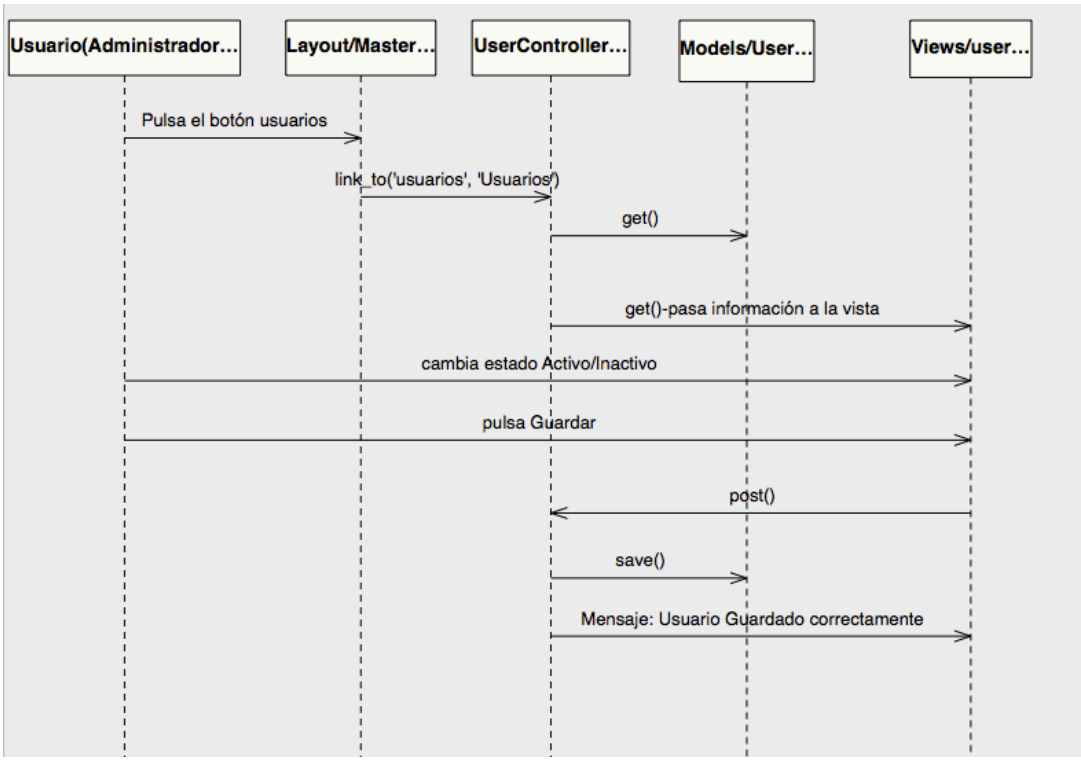

Ilustración 5 - Diagrama de secuencia para activar usuarios

Creado Por: Franklin Paula / Xavier Rivas (2014-09-20)

3.4.4.2Administrar Alertas

Podemos observar en el gráfico la interacción de un usuario para administrar las alertas que desea recibir.

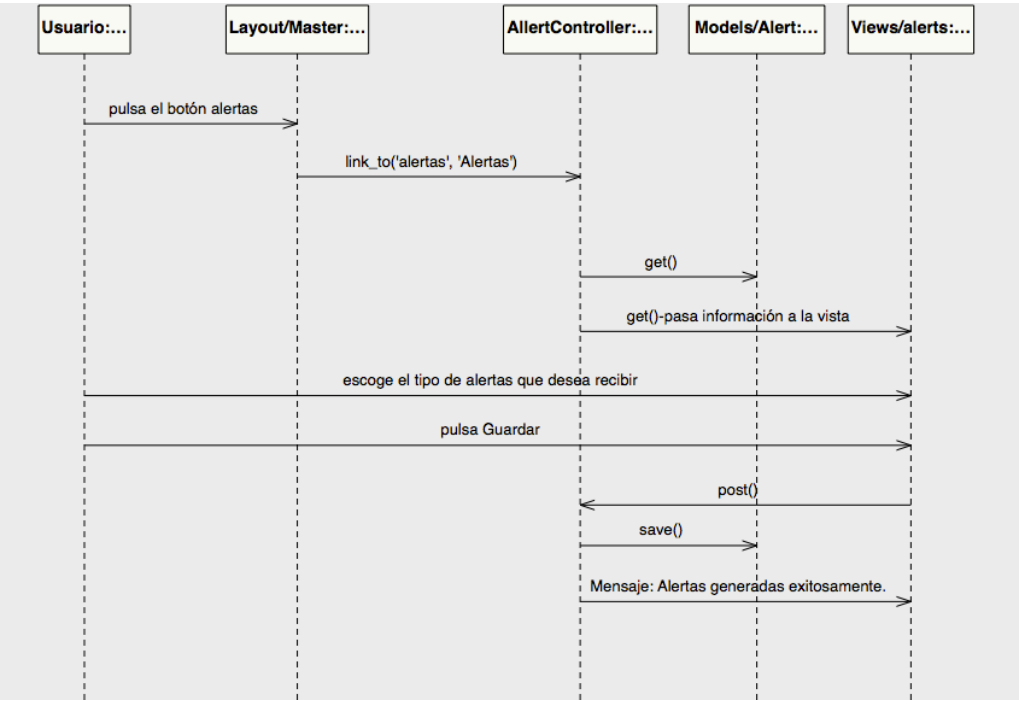

Ilustración 6 - Diagrama de Secuencia Administrar Alertas

Creado Por: Franklin Paula / Xavier Rivas (2014-09-20)

3.4.5 Requisitos no funcionales

Disponibilidad: El sistema estará disponible las 24 horas del día los 365 días del año, salvo por días previamente establecidos en los cuáles se dará mantenimiento al servidor; el sistema saldrá del aire máximo una hora.

Debido a que al ser un servicio en la nube se requiere un costo mensual para mantener el servidor al aire.

Seguridad: El sistema será seguro primero que nada por las bondades que nos brinda el framework antes mencionado, por otro lado la arquitectura de programación MVC, nos brinda seguridad debido a que los datos nunca pasan directamente por la interfaz del usuario.

El sistema estará protegido en contra de los siguientes ataques:

- ***Inyección Sql:***

Se denomina inyección sql o "sqlinjection" a toda actividad que incrusta código sql infiltrado en un sistema además de una porción de código.

Dicha actividad se presta debido a que los sistemas tienen vulnerabilidades informáticas al momento de validar entradas para realizar cualquier tipo de querysql (SELECT,INSERT,UPDATE,DELETE).

Entonces podemos decir que se produce un ataque sql cuando existe una inyección de código sql infiltrado en código sql nativo del sistema.

Dichas inyecciones generalmente son creadas para poder trabajar de manera maliciosa, por lo cual la seguridad de los datos del sistema puede quedar expuestos a terceros.

Todas estas inyecciones son causadas generalmente porque el programador no usa clases para ejecutar queriessql, por lo tanto las consultas se las hacen de forma tradicional.

Ejemplo:

Select * from tabla where name = “campo_id”;

Si se realiza la consulta de esta manera donde observamos que campo_id es un campo, ahí podría ir cualquier clase de sql malintencionado.

Por lo tanto si en el sistema en vez de escribir un id escribimos cualquier otra sentencia sql, dicha sentencia se ejecutaría sin ningún problema.

Select * from tabla where name = “Franklin”;

En este caso el usuario realizó una consulta normal, pero si el usuario es mal intencionado y escribe en vez de Franklin, el siguiente query: Franklin”;Deletefrom tablawherename LIKE “%

Se generaría el siguiente query:

Select * from tabla where name = “Franklin”;Delete from tabla where name LIKE “%”;

Si observamos el texto en verde es parte del código nativo de la aplicación. En cambio el texto en rojo es el código inyectado.

- ***CSRF:***

Podemos decir que es todo ataque que se realiza a través de una URL que genera una acción determinada. Dicha URL es ejecutada por un usuario

que el sistema confía, el usuario puede no saber que ha ejecutado la acción y puede ser desde otro sitio.

Ejemplo 1:

La petición www.cloudinventory.com/admin/users/delete/id/45, la URL es clara y lo que hace es borrar un usuario con id 45. Si dicha acción es plasmada en otra página y es ejecutada por el usuario se realizará sin ningún problema.

Ejemplo 2:

Supongamos que el sitio www.juegosgratis.com es ingresado por el usuario que administra www.cloudinventory.com

El usuario ha hecho log in en www.cloudinventory.com y no ha cerrado sesión.

Supongamos que el sitio www.juegosgratis.com tiene una imagenHTML con un atributo "src" incrustado una URL que genera una acción en el sitio www.cloudinventory.com de la siguiente manera:

<imgsrc =”www.cloudinventory.com/admin/users/delete/id/45” />

Al momento que la página www.juegosgratis.com cargue dicha imagen se ejecutará la acción delete del usuario 45 en la web www.cloudinventory.com.

El usuario no sabrá que ejecutó dicha acción. Así se produciría un ataque exitoso sin conciencia del usuario que administra la página www.cloudinventory.com.

Accesibilidad: El sistema será accesible, fácil de usar.

Podrá ser accedido desde cualquier parte del mundo, con acceso a Internet.

Usabilidad: El sistema será intuitivo.

Interfaz: el sistema tendrá una interfaz gráfica agradable al uso del usuario.

Costo: El costo del proyecto dependerá del número de usuarios. Pero para iniciar se centrará en el arrendamiento de un cloud server Centos que cuesta $139 dólares mensuales, el costo del dominio que varía de $10 a $15 dólares anuales.

Por ende el costo fijo mensual será $140 dólares.

El precio del servicio por implementar el software dependerá de lo siguiente:

- De 2 a 5 Usuarios $200 mensuales
- De 6 a 10 Usuarios $400 mensuales
- De 11 a 15 Usuarios $600 mensuales

Operatividad: el sistema será capaz de ser ejecutado en cualquier explorador que soporte HTML5.

Interoperatividad: el sistema será capaz de ser accedidodesde cualquier sistema operativo que tenga la capacidad de soportar un explorador de internet con soporte HTML5.

Mantenibilidad: el sistema será de fácil mantenimiento, debido al framework la organización de carpetas y cache es de fácil acceso.

Rendimiento: el sistema garantiza al menos el 90% de su rendimiento de mano con la infraestructura y el 10% debido a casos fortuitos.

3.5 Diseño, desarrollo, implementación y pruebas

3.5.1 Introducción a AUP

Agile UnifiedProcess es una metodología de desarrollo enfocada a evitar caminos tradicionales que compliquen el desarrollo de un sistema, de forma liviana, centrándose en la gente y su desarrollo.Es una versión simplificada del Proceso Unificado de Rational (RUP).

La forma de desarrollo de AUP es enfocada en técnicas ágiles y conceptos que todavía se mantienen activos en RUP.

Entre algunas de las técnicas usadas por AUP tenemos las siguientes:

- Desarrollo dirigido por pruebas
- Modelado Ágil
- Gestión de cambios ágil
- Refactorización de Base de Datos para mejorar la productividad

Es un marco de desarrollo iterativo e incremental, se considera iterativo e incremental debido a que cada desarrollo pasa por una etapa y especifica un conjunto de actividades, además que siempre se menciona los artefactos involucrados en el desarrollo de software.

Debido a que es un marco de desarrollo que se enfoca en etapas y procesos, puede ser fácil de adaptar a nuevos marcos de desarrollo.

AUP establece un modelo simple que en una sola disciplina reúne las disciplinas de modelado de negocio, requisitos, análisis y diseño.

3.5.2 Historia de AUP

La metodología de desarrollo AUP es relativamente nueva, debido a su esquema.

Su primera versión fue liberada en el año 2005, su creador es Scott Ambler.

Scott Ambler dedicó todo su trabajo en mejorar la metodología RUP(RationalUnifiedProcess).

En 1999 desarrolló la metodología EUP(Enterprise UnifiedProcess) que convierte a RUP en un ciclo de vida para el desarrollo de sistemas.

Scott Ambler nació en Canadá en el año 1966, es consultor y autor, actualmente trabaja en su consultoría Scott Ambler + Associates.

Es autor de varios libros y publicaciones tales como:

Publicación	Año
The object primer: the application developer's guide to object-orientation. SIGS Books	1995
Process patterns: building large-scale systems using object	1998

technology. Cambridge University Press	
Agile Modeling: Effective Practices for Extreme Programming and the Unified Process	2002
The Unified Process Transition and Production Phases	2002
The Practical Guide to Enterprise Architecture with James McGovern, Mike Stevens, James Linn, VikasSharan, and Elias Jo. Prentice Hall	2003
Agile database techniques: effective strategies for the agile software developer.	2003
The Object Primer: Agile Model Driven Development with UML 2.	2004
Enterprise Unified Process: Extending the Rational Unified Process	2005
Refactoring Databases: Evolutionary Database Design	2006
Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise	2012

Tabla 6 - Publicaciones de Scott Ambler

Creado Por: Franklin Paula / Xavier Rivas (2014-09-20)

3.5.3 Descripción de AUP

La descripción del flujo de trabajo de AUP se basa en 4 simples pasos:

Modelado:es el flujo de trabajo en el cual se basa el entendimiento del negocio, su problema actual, y buscar una solución para dicho problema.

Implementación:es el flujo de trabajo que se encarga de implementar la solución del problema (modelado) a través de un código ejecutable, después de implementarlas, se realiza una prueba básica del código (pruebas unitarias)

Prueba: es el flujo que sirve para garantizar la calidad del sistema mediante evaluaciones, verificar que se cumplan los requerimientos y validar si el sistema funciona tal y como está.

Despliegue: es el flujo que se encarga de describir el plan que se va a realizar para que el sistema sea implementado, ejecutado, etc. Para que quede a disposición de los usuarios finales.

3.5.4 Lineamientos para trabajar con AUP.

Son 5 los lineamientos requeridos para trabajar con AUP.

Simplicidad:la simplicidad es enfocada a que no debe existir mucha documentación del sistema, el sistema debe ser de un uso fácil e intuitivo, la documentación estará enfocada a describir de manera concisa el funcionamiento del sistema.

Agilidad:desarrollo rápido del sistema.

Actividades de alto valor:esto se refiere a que todo el desarrollo se basa en prioridades, esto quiere decir que algunas funciones, secciones, etc. Tendrán más prioridad que otras, esto quiere decir que la atención del desarrollo será dirigida a los componentes que en realidad lo requieran y no en todo el proyecto.

Herramienta de independencia:se refiere a que se puede usar cualquier tipo de herramienta con AUP, la que mejor se acople al sistema. Lo importante es que estas herramientas nos ayuden en el desarrollo y que no sean una complicación, un ejemplo es el desarrollo con herramientas libres o "open source" que nos pueden ayudar bastante y además no se debe pagar por utilizarlas.

Adaptar el producto a sus necesidades:esto va de la mano con el lineamiento anterior, todas las herramientas utilizadas, deben ser adaptadas a nuestras necesidades como desarrolladores. Una herramienta que nos cause problemas, que debamos pagar por utilizarla,

o que nos dé un lineamiento de trabajo, nos puede complicar más que ayudar, por lo tanto debemos ser cuidadosos al momento de escoger nuestras herramientas de trabajo.

3.5.5 Etapas de AUP

3.5.5.1 Iniciación

Identificación del alcance y dimensión del proyecto, propuesta de la arquitectura y presupuesto del cliente.

3.5.5.2 Elaboración

Confirmar que la arquitectura sea la correcta.

3.5.5.3 Construcción

Desarrollo del sistema siguiendo las prioridades del mismo.

3.5.5.4 Transición

Validar e implementar el sistema.

Capítulo 4Diseño

4.1 Diseño de la base de datos

El diseño está basado en el modelo conceptual y el modelo físico orientado a MySQL

Modelo Conceptual:

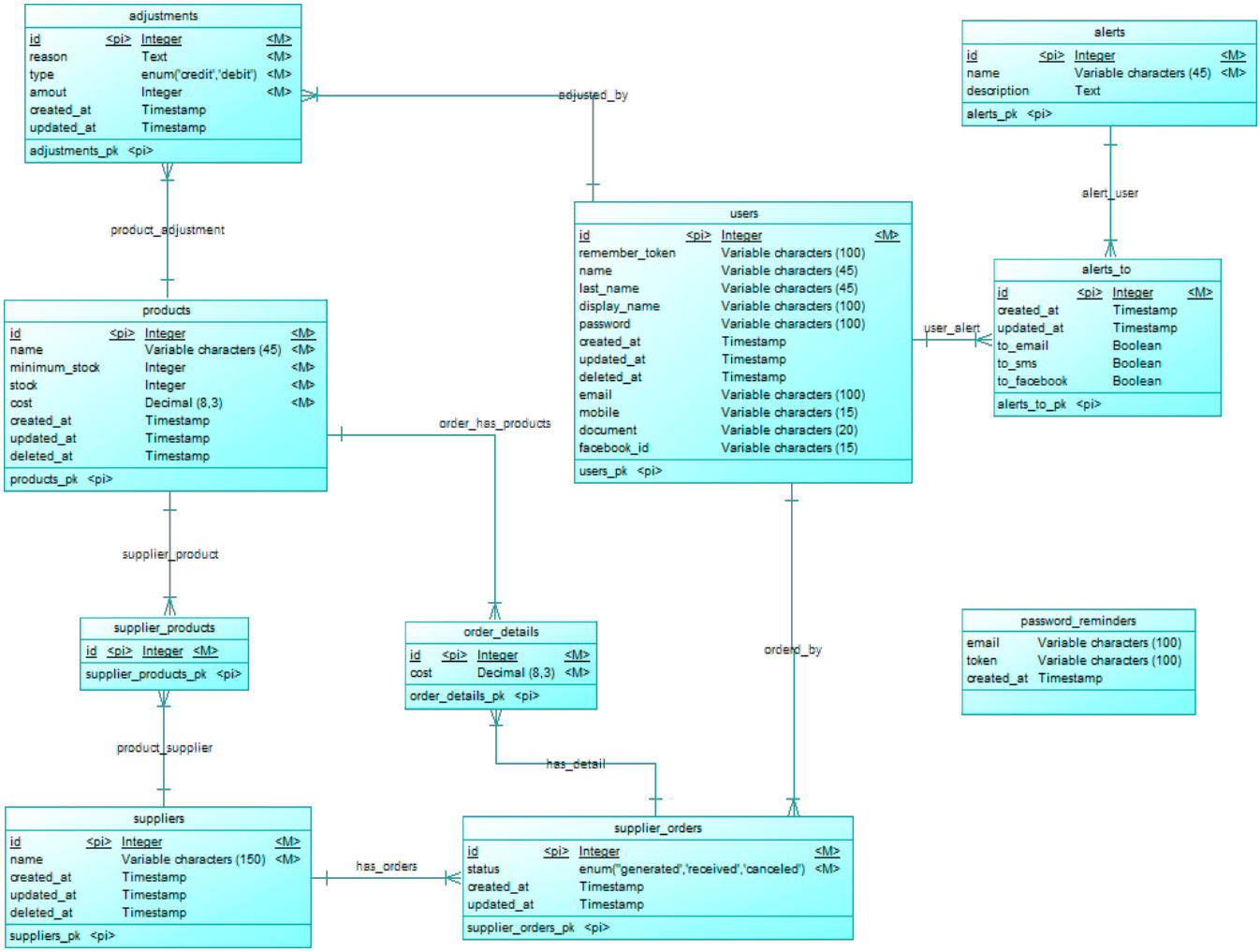

Ilustración 7 - Modelo Conceptual de la base de datos

Creado Por: Franklin Paula / Xavier Rivas (2014-09-20)

Modelo Físico:

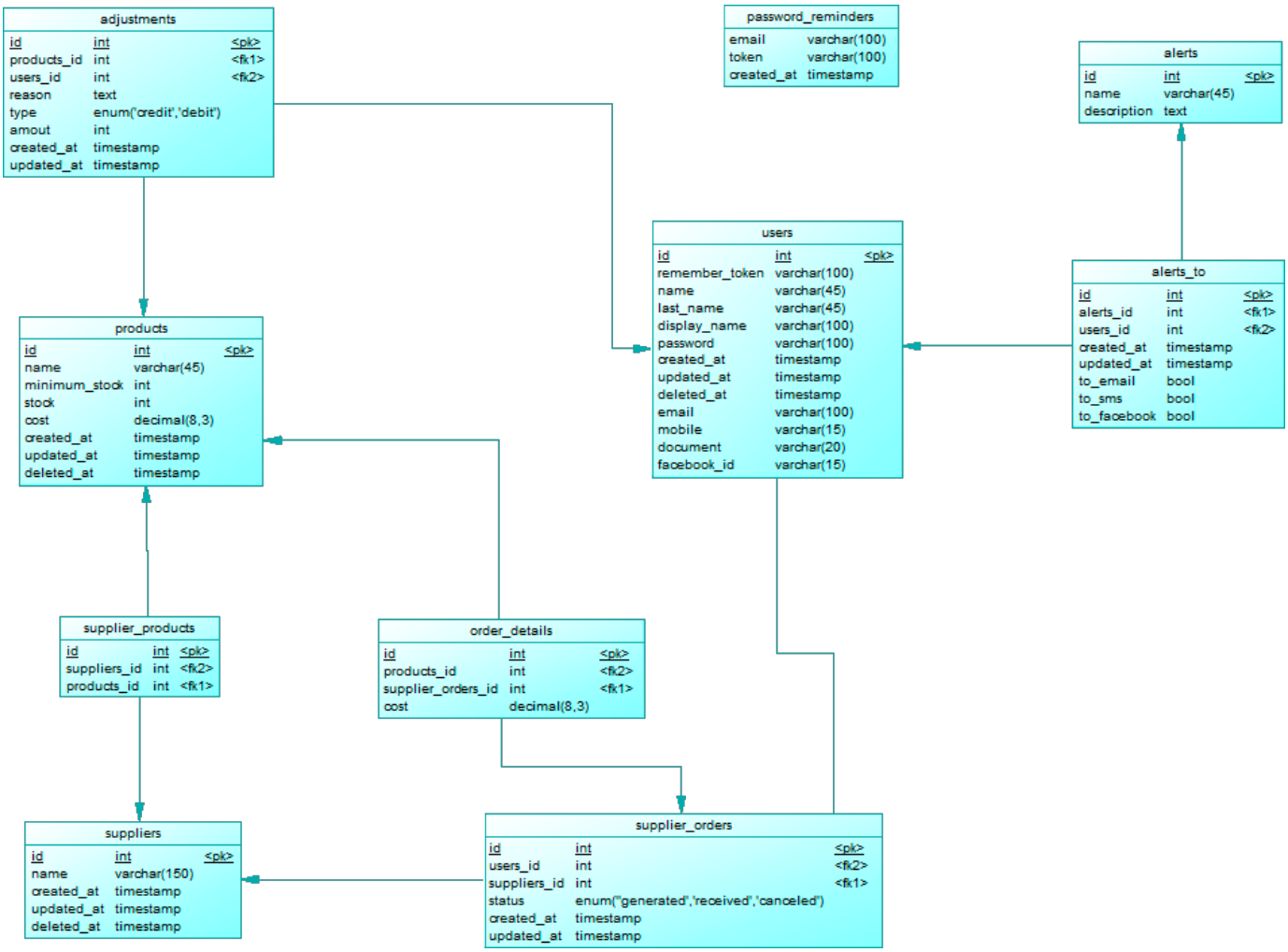

Ilustración 8 - Modelo Físico de la base de datos

Creado Por: Franklin Paula / Xavier Rivas (2014-09-20)

4.2 Estructura de interfaces

La estructura de las interfaces será la siguiente:

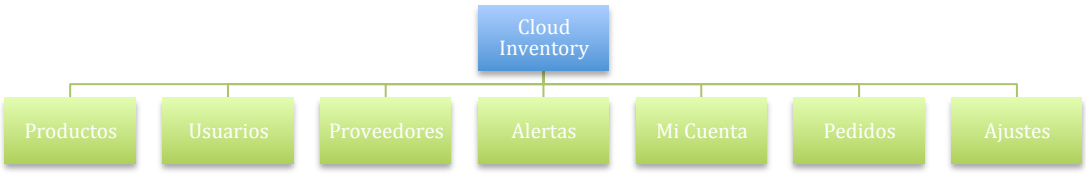

Ilustración 9 - Interfaces a desarrollar

Creado Por: Franklin Paula / Xavier Rivas (2014-09-20)

Se desarrollarán las páginas indicadas anteriormente basadas en la estructura de interfaz mostrada a continuación:

Ilustración 10 - Estructura de las Interfaces

Capítulo 5Desarrollo

5.1 Creación de base de datos

La creación de base de datos es realizado a partir del modelo físico.

Todos los procesos que afectan más de una tabla son realizados con transacciones en la base de datos, esto quiere decir que si falla un proceso se revierte toda la transacción y por ende no queda ninguna tabla afectada.

El script de base de datos fue generado con PowerDesigner, a partir del modelo físico.

Dicho script se encuentra adjunto en el CD-ROM.

5.2 Funcionalidad de las interfaces diseñadas

Las interfaces diseñadas fueron realizadas con definiciones de usabilidad de usuario, para que el usuario tenga una vista sencilla del sistema.

El diseño ayuda a minimizar los clics y el cambio de pantallas.

El cuerpo izquierdo presenta la lista de registros existentes, mientras el cuerpo principal contiene el detalle del nuevo registro o del registro seleccionado.

Todas la interfaces se basaron en los conceptos de UI y UX.

- ***UI (user interface/interfaz de usuario):***

La interfaz de usuario es el medio por el cual un usuario puede comunicarse con un sistema, dicha interfaz debe ser sencilla de usar para que el usuario pueda manejarla de manera intuitiva.

Debe tener un menú de navegación, con botones y palabras que indiquen exactamente la acción que se está realizando.

Por lo tanto toda interfaz de usuario debe ser fácil de entender y fácil de accionar.

Las interfaces de usuario varían dependiendo del sistema, todo sistema es diferente por ende toda interfaz de usuario es diferente.

- ***UX (Userexperience / experiencia de usuario)***

La experiencia del usuario es como los usuarios se sienten navegando o trabajando en un sistema, el conjunto de factores, elementos, botones,

sonidos, etc. Serán los que causen una experiencia positiva o negativa del usuario hacia el sistema.

Esta experiencia no solo depende de la interfaz, también depende del desenvolvimiento del sistema, tiempo de respuesta, etc.

Los principales elementos son los siguientes:

- Hardware
- Software
- Usabilidad
- Diseño gráfico y visual
- Calidad de los contenidos
- Buscabilidad o encontrabilidad
- Utilidad

Como podemos entender UX, va de la mano con los requerimientos funcionales, casos de uso, etc., que fueron realizados de manera exitosa.

El sistema desarrollado es intuitivo, fácil de usar y fácil de acceder, por lo tanto cumple con todo lo necesario para que el usuario tenga la mejor experiencia posible.

Todas las interfaces están basadas en la estructura antes mencionada.

Como podemos ver a continuación, todas las interfaces tienen un diseño muy fácil y comprensible.

Elementos	Cumple con características UX
Hardware	Si
Software	Si
Diseño gráfico y visual	Si
Calidad de los contenidos	Si
Buscabilidad	Si

Utilidad	Si

Tabla 7 - Características de las interfaces

Creado Por: Franklin Paula / Xavier Rivas (2014-09-20)

5.3 Interfaces

5.3.1 Administración de usuarios

En la Ilustración 10 podemos observar la administración de usuarios, sus campos al editar o crear un usuario, y los usuarios existentes.

Ilustración 11 - Interfaz Administración de Usuarios

5.3.2 Administración de alertas

En la Ilustración 11 podemos observar la administración de las alertas, escogiendo a donde se desea que se notifique dependiendo del tipo de alerta.

Ilustración 12 - Interfaz Administración de Alertas

5.3.3 Administración de productos

En la Ilustración 12 podemos observar la interfaz de productos, los productos existentes y además los campos de cada producto.

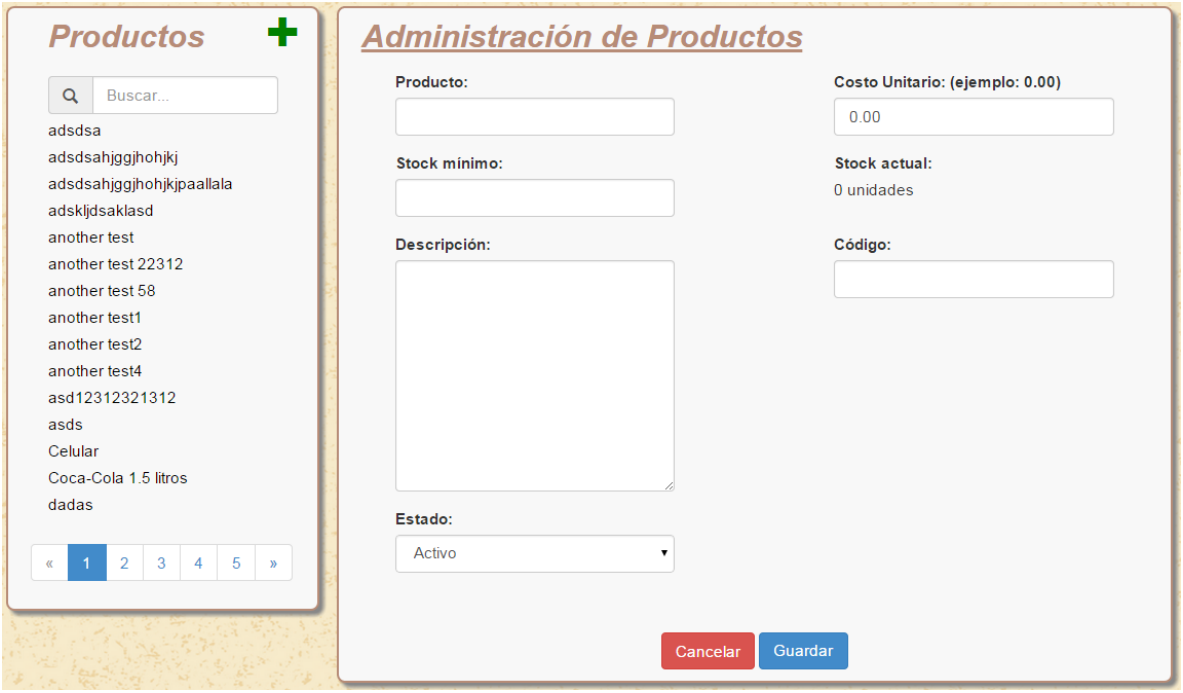

Ilustración 13 - Interfaz Administración de Productos

5.3.4 Administración de distribuidores (detalle)

En la ilustración 13 podemos observar los distribuidores existentes y los campos que existen del distribuidor al momento de actualizar o crear uno.

Ilustración 14 - Interfaz Administración de distribuidores(detalle)

5.3.5 Administración de distribuidores(producto)

En la ilustración 14 podemos observar los productos que son asignados a cada distribuidor, podemos agregar o eliminar productos.

Ilustración 15 - Administración de distribuidores(productos)

5.3.6Generación de pedidos

En la ilustración 15 podemos observar la interfaz de generación de un pedido, los pendientes y sus pedidos.

Ilustración 16 - Interfaz Generación de Pedidos

5.3.7Recepción de pedidos

En la ilustración 16 podemos observar la interfaz de recepción de pedidos, en la cual tenemos los pendientes y sus pedidos.

Ilustración 17 - Interfaz Recepción de Pedidos

5.3.8 Ajustes

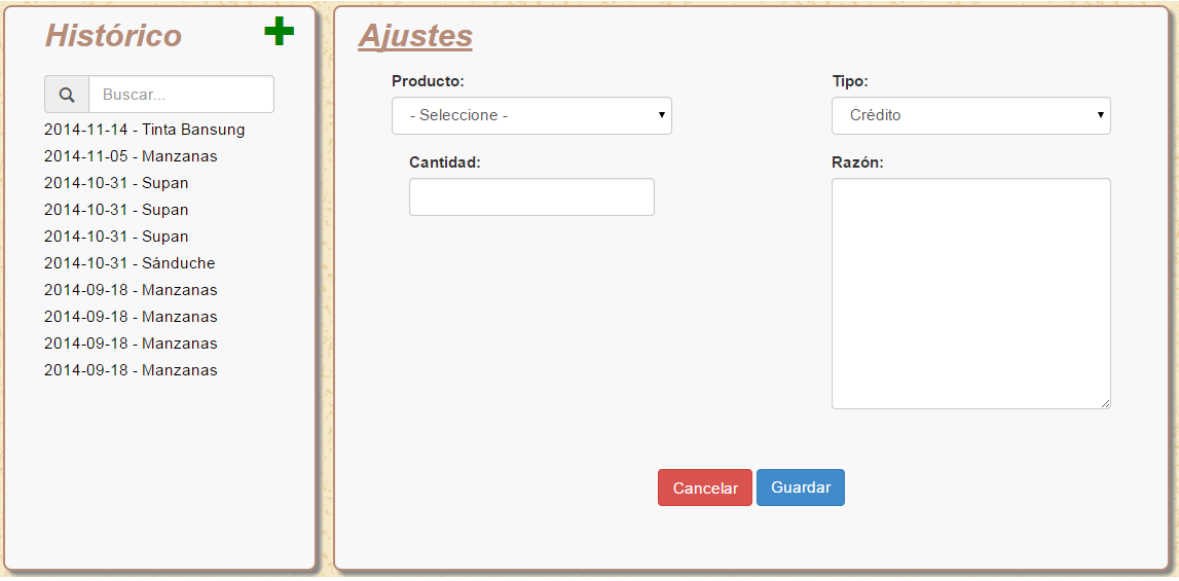

Ilustración 18 - Interfaz de Ajustes

Capítulo 6 Implementación y pruebas

6.1 Codificación

La codificación se realizó siguiendo un modelo MVC, con ayuda del framework de PHP llamado Laravel, en su versión 4.2. Se utilizó varias de sus utilidades, entre ellas podemos nombrar las siguientes:

- Autenticación de usuarios.
- Recordatorio de contraseña.
- Eloquent ORM (manejo e interacción de base de datos).
- Bladetemplating (simplicidad en generación de código HTML).
- Protección contra ataques SCRF.
- Protección contra ataques de inyección sql.

La interfaz gráfica fue diseñada utilizando Bootstrap, framework que simplifica el manejo correcto de HTML, CSSy ciertas funcionalidades Javascript.

Para el envío de alertas se siguió el siguiente algoritmo mediante un diagrama de flujo, para poder entender cómo funciona el corazón del sistema que es el envío de las alertas.

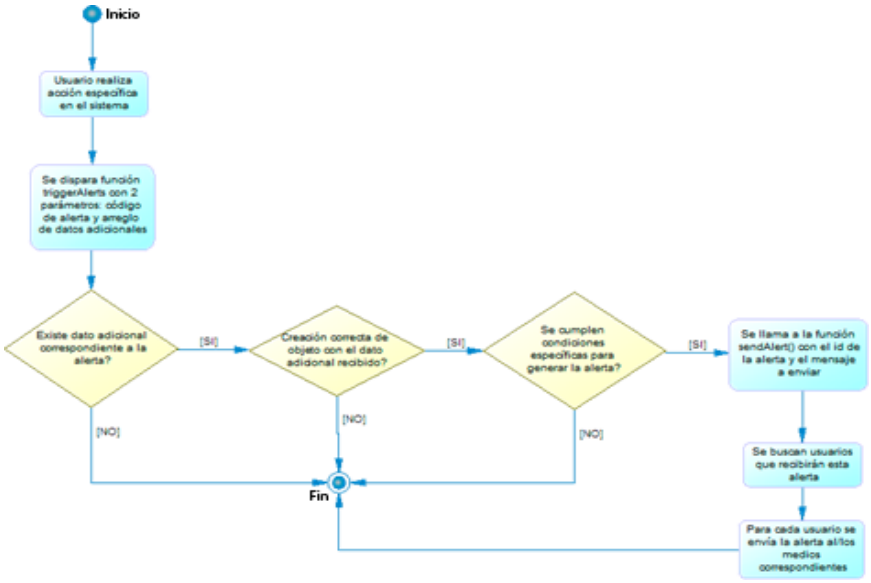

Ilustración 19 - Diagrama de Flujo de las Alertas

Creado Por: Franklin Paula / Xavier Rivas (2014-09-20)

Todo el código se encuentra adjunto en el CD-ROM.

6.2 Pruebas del Sistema

Módulo Usuarios

Nº	Descripción	Resultado esperado	Estado
1	Crear un usuario sin ingresar ningún dato en el formulario	Se muestran errores en el formulario	Correcto
2	Crear un usuario con datos correctos	Se muestra mensaje de creación satisfactoria y se crea un registro en BDD, además aparece el usuario creado en el listado izquierdo de la interfaz. Se envía un correo electrónico al email ingresado con los datos de inicio de sesión.	Correcto
3	Crear un usuario con email en formato incorrecto	Se muestra el error en el campo Email y no se crea ningún registro en la BDD	Correcto
4	Crear un usuario con caracteres alfabéticos en el campo teléfono	Se muestra el error en el campo Teléfono y no se crea ningún registro en la BDD	Correcto
5	Editar usuario sin cambiar ningún valor	Se muestra mensaje de éxito.	Correcto
6	Editar usuario cambiando su email a uno perteneciente a otro usuario	Se muestra el error en el campo correspondiente y no se edita el registro en BDD.	Correcto

Tabla 8 - Pruebas Módulo Usuarios

Módulo Productos

Nº	Descripción	Resultado esperado	Estado	Acción
1	Crear un producto sin ingresar ningún dato en el formulario	Se muestran errores en el formulario	Correcto	
2	Crear un producto con datos correctos	Se muestra mensaje de creación satisfactoria y se crea un registro en BDD, además aparece el producto creado en el listado izquierdo de la interfaz	Correcto	
3	Crear un producto con stock negativo.	Se muestra el error en el campo stock y no se crea ningún registro en la BDD	Corregido	Aumentar validación de campo positivo
4	Editar producto sin cambiar ningún valor	Se muestra mensaje de éxito.	Correcto	
5	Editar producto cambiando su estado	Se edita el registro en BDD y se muestra mensaje de éxito.	Correcto	

Tabla 9 - Pruebas Módulo Productos

Módulo Distribuidores

Nº	Descripción	Resultado esperado	Estado
1	Crear distribuidor sin ingresar ningún dato en el formulario	Se muestran errores en el formulario	Correcto
2	Crear distribuidor con datos correctos	Se muestra mensaje de creación satisfactoria y se crea un registro en	Correcto

		BDD, además aparece el distribuidor creado en el listado izquierdo de la interfaz	
3	Crear un producto con RUC existente en BDD	Se muestra el error en el campo RUC y no se crea ningún registro en la BDD	Correcto
4	Editar producto. Agregar productos en la pestaña "Productos"	Los campos funcionan correctamente, no se despliega mensaje de error alguno y se crean registros en BDD.	Correcto
5	Revisar que productos desactivados no aparezcan en el listado de la pestaña "Productos"	Productos desactivados no aparecen en la lista.	Correcto

Tabla 10 - Pruebas Módulo Distribuidores

Módulo Pedidos

Nº	Descripción	Resultado esperado	Estado
1	Crear un pedido sin ingresar ningún dato en el formulario	Se muestran errores en el formulario	Correcto
2	Agregar 2 veces el mismo producto al pedido	Se muestra mensaje de error indicando que el producto puede ser añadido una única vez al pedido.	Correcto
3	Modificar cantidad de un producto agregado al pedido.	La interfaz muestra un cambio en la cantidad del producto de la fila con la cual se está interactuando	Correcto
4	Crear pedido	Se muestra mensaje de éxito y se crea un registro en BDD en estado "pending"	Correcto

5	Seleccionar pedido pendiente	Aparece un detalle de los productos en el pedido seleccionado y con la cantidad correcta anteriormente ingresada.	Correcto
6	Recibir pedido	Se muestra mensaje de éxito, se modifica el registro a estado "received" en BDD y se afecta el stock de los productos relacionados al pedido en la cantidad correcta.	Correcto

Tabla 11 - Pruebas Módulo Pedidos

Módulo Ajustes

Nº	Descripción	Resultado esperado	Estado
1	Crear un juste sin ingresar ningún dato en el formulario	Se muestran errores en el formulario.	Correcto
2	Crear una alerta tipo débito, con cantidad mayor al stock del producto	Se muestra mensaje de error y no se afecta ningún registro en BDD.	Correcto
3	Crear una alerta tipo crédito para un producto con cualquier cantidad	Se muestra mensaje de éxito y se afecta el stock de dicho producto en BDD.	Correcto
4	Seleccionar un ajuste histórico	Se muestran los datos correctos del ajuste realizado	Correcto

Tabla 12 - Pruebas Módulos Ajustes

Módulo Alertas

Nº	Descripción	Resultado esperado	Estado
1	Seleccionar un usuario que no	Ninguna casilla de verificación de la interfaz está seleccionada.	Correcto

	tenga alertas relacionadas		
2	Seleccionar alertas al azar y guardar.	Se muestra mensaje de éxito.	Correcto
3	Seleccionar usuario editado en Nº2.	Las casillas de verificación que aparecen seleccionadas son las mismas que fueron previamente elegidas en el numeral anterior.	Correcto

Tabla 13 - Pruebas Módulo Alertas

Funcionalidad

Nº	Descripción	Resultado esperado	Estado
1	Inicio de sesión	Se verifican credenciales al momento de iniciar sesión	Correcto
2	Recordatorio de contraseña	Se envía email con nuevas credenciales con validez de 5 minutos.	Correcto
3	Carga de menú de acuerdo al perfil de usuario	Se solo presentan las opciones a las que el usuario tiene acceso dentro del menú.	Correcto
4	Seguridad por URL	Iniciar sesión con un administrador. Ingresar al menú "Receptores de alertas". Copiar URL y cerrar sesión. Iniciar Sesión con un usuario tipo cliente. Pegar URL. Se hace una redirección a la página de inicio.	Correcto
5	Generación de alerta de mínimo de stock	Se realiza un ajuste de tipo débito del producto "Manzana", con la misma cantidad que hay en existencia. Se deben generar alertas de mínimo de stock y de inventario cero a los usuarios y medios correspondientes.	Correcto

Tabla 14 - Pruebas Funcionalidad

6.3 Conclusiones y Recomendaciones

Conclusiones

1. El desarrollo del sistema se simplificó en gran parte debido a la experiencia en el campo del desarrollo de software que tienen los miembros del equipo.
2. Los requerimientos fueron analizados para que el usuario final tenga una experiencia sencilla y clara del sistema.
3. Acerca de Laravel:
    - Laravel es un framework poderoso que nos ayuda a crear consultas, registros, etc. De una manera sencilla y fácil y para este caso se acoplo de una manera eficaz al desarrollo.
    - Laravel como framework nos ayudó a realizar todas las seguridades para que el sistema sea protegido contra ataques comunes en la web.
    - Laravel posee librerías poderosas que nos ayudaron a gestionar de manera más sencilla la base de datos, la inserción de datos, la consulta de datos y la actualización de datos.
    - La organización de archivos en Laravel nos indica claramente cómo manejar una arquitectura MVC
4. La arquitectura MVC nos ayuda a manejar un ambiente organizado de desarrollo.
5. Todas las validaciones fueron realizadas del lado del servidor, por medidas de seguridad, debido a que realizar validaciones del lado del cliente pudo ser una vulnerabilidad del sistema, si desactivan el Javascript en el explorador.
6. Con un ambiente de desarrollo estable, organización de archivos y una arquitectura clara, el desarrollo se vuelve eficaz y eficiente.
7. La metodología escogida AUP nos ayudo debido a que queríamos que el sistema sea ágil, y rápido de realizar. AUP nos ayudó con un modelado ágil, refactorizaciones rápidas, etc. Los casos de prueba además fueron realizados de tal manera que se pueda corregir bugs de manera prácticamente inmediata.

8. El diseño de la base de datos y en si del sistema es adaptable y portable por lo tanto agregar nuevos módulos no tendría ningún inconveniente para futuras versiones.
9. De acuerdo a la experiencia en algunos casos, el cliente no tiene claro lo que quiere, por lo tanto esta aplicación ofrece el producto definido, ajustando a requerimientos específicos del cliente.

Recomendaciones

1. En tiempos actuales, a medida de lo posible, debe implementarse la mayor cantidad de seguridades posibles, sobre todo en sistemas que estarán en la nube, para garantizar la seguridad de la información de los clientes.
2. Tener claras las funcionalidades del sistema previo inicio del desarrollo, para así evitar pérdidas de tiempo tanto por nuevos desarrollos como por ediciones de funcionalidades previamente realizadas.
3. Utilizar frameworks para el desarrollo de aplicaciones, ya que estos ayudan a mejorar nuestro desempeño, mejoran las técnicas de desarrollo, e implementan varios tipos de seguridad internamente, evitando así hacerlo personalmente.
4. Es recomendable que el sistema sea utilizado en exploradores Firefox, Chrome o Safari.
5. En caso de que se modifique el sistema, es recomendable realizarlo mediante repositorios Git. Para que siempre este visible el cambio, en que parte fue y poder tener un respaldo.
6. Para una nueva versión se podría aumentar lo siguiente:

   Control de actividad de usuarios

   Notificaciones dinámicas (el sistema analiza que notificación podría ayudarle más al usuario que otra y la activa sin necesidad que el usuario lo solicite.).

   Módulo de ventas.
7. Si el sistema va a ser modificado es necesario contratar o conseguir personal con conocimientos claros en Laravel, MVC y base de datos.

8. Ningún sistema es 100% seguro, lo importante es entender el problema y si llega a ser vulnerado corregir el problema de manera rápida.
9. Durante el desarrollo del proyecto se ha enfocado mucho en lo que es Frameworks MVC y en especial Laravel, por lo tanto se recomienda a la Facultad de Ingeniería, realizar seminarios en los cuales se expliquen estas tecnologías para poder desarrollar sistemas, páginas web, etc. A veces la facultad nos enseña teoría y metodologías, pero salimos al mundo laboral y entendemos que las herramientas de desarrollo son muy diferentes a las aprendidas en el ámbito académico, por lo tanto es necesario un seminario para poder entender dichas tecnologías.

CAPÍTULO 7 GLOSARIO

A

abstracción

Aislar un elemento del resto de similares., 9

accesibilidad

Grado en el que las personas pueden usar el sistema., 21

alerta

Alarma disparada cuando ocurre una acción específica., 17

ANSI

American National Standards Institute. Organizaci{on que supervisa el desarrollo de estándares en EEUU. Esta organizaación es miembro también de la ISO., 8

AUP

Agile unified process. Metodología de desarrollo de software dirigido por pruebas y desarrollo rápido y ordenado, con facilidad para la gestión de cambios., 11

B

bajo acoplamiento

Tener las clases lo menos ligadas posible entre sí., 9

base de datos relacional

Modelo de base de datos que permite crear relaciones entre las tablas de la base de datos., 7

C

caché

Memoria de acceso rápido en un ordenador donde se pone la información más recientemente procesada, 3

Cloud Computing

Es un paradigma que permite dar servicios a través de Internet., 1

código abierto

Software distribuido y desarrollado libremente., 4

controlador

Puente de conexión entre los datos de la base de datos y el despliegue de ésta por pantalla. Aquí se debe poner toda la lógica del negocio., 14

Cookies

Pequeña cantidad de información almacenada en el navegador del usuario, la cual indica la actividad reciente del usuario., 6

CSRF

Ataque realizado por URL., 20

curva de aprendizaje

Grado de éxito obtenido durante el aprendizaje con relación al tiempo transcurrido., 4

D

despliegue

Descripción del plan a ser ejecutado para que el sistema sea implementado., 25

diagramas de secuencia

Diagrama utilizado para indicar la interacción entre objetos en un sistema según UML., 17

disponibilidad

Tiempo que el servicio está a disposición de los usuarios., 19

E

escalabilidad

Capacidad de agregar nuevas funcionalidades al sistema que ya está desarrollado., 10

F

framework

Herramienta que a base de un lenguaje de programación, implementa funcionalidades