BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

Sistematización de documentación para programas de la Secretaría de Desarrollo Rural Sustentabilidad y Ordenamiento Territorial aplicando MVC.

TESIS PARA OBTENER EL TITULO DE:

LICENCIADA EN INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

FRANCISCA LÓPEZ VELÁZQUEZ

ASESOR:

CARLOS ARMANDO RÍOS ACEVEDO

ABRIL 2018

Agradecimientos

En primer lugar quiero agradecer con mucho cariño, a mis padres Claudio López Vázquez y Pompeya Velázquez Jiménez por todo el apoyo brindado durante la trayectoria de la carrera, y en esta última etapa; agradezco su paciencia y confianza que me han dado.

Quiero expresar mi agradecimiento a mi tía Guadalupe García Maldonado, por el apoyo en esta última etapa, por la confianza que me ha dado, por sus consejos y a la cual le he tomado mucho cariño.

También agradecer a mi primo Carlos Sánchez García a quien aprecio mucho, por todo el apoyo y confianza que me ha brindado en esta última etapa.

Un profundo agradecimiento al Mtro. Carlos Armando Ríos Acevedo por todo el apoyo y motivación que me ha brindado desde que recurrí a él, por creer en mí, por la paciencia y darme la oportunidad de concluir esta etapa.

A la Mtra. Judith Pérez Marcial, por el apoyo en asesoría durante el trayecto de desarrollo del proyecto de tesis; a la cual admiro mucho como profesionista y ser humano.

Al profesor Gilberto López Poblano por su apoyo en asesoría, durante el trayecto de desarrollo de proyecto de tesis.

Y un sincero agradecimiento a mis amigos, Juan Carlos Pérez Medina y César Zamora Campos; por todo su apoyo, comprensión y motivación, por ser grandes personas de los cuales he aprendido mucho como ser humanos.

Índice

Introducción

El siguiente proyecto es una actualización de software, fue realizado para satisfacer las necesidades de los usuarios que laboran al interior de la Secretaría de Desarrollo Rural, Sustentabilidad y Ordenamiento Territorial. Dicho sistema maneja diversas solicitudes (compras, cómputo, vehicular y general) de llenado de datos.

La sistematización es para llevar un control de procesos que los agilice [1], [11]. Se pretende obtener resultados satisfactorios como: Buen manejo de la documentación y de registros almacenados en una base de datos que servirán para la manipulación de información requerida en cada solicitud.

El software que se utilizará para desarrollar la aplicación web será Codeigniter [3] que utiliza el lenguaje de programación PHP [2], [17] diseñado originalmente para realizar páginas web dinámicas, utilizando HTML5 [5] y Hojas de estilo en cascada donde se desarrollará la interfaz gráfica [6].

Se utilizará Codeigniter [4] por ser una de las aplicaciones web más completas que hay actualmente, caracterizada por sus herramientas que posee; facilita la creación de sistemas haciéndolo flexible, dinámico y de fácil acceso, para cualquier usuario (Programador) que lo utilice.

Antecedentes del Proyecto.

El manejo de información derivaba en delimitada agilidad de procesos [1], [10] al realizar diversas solicitudes dentro de la Secretaría; como resultado de ello, la recepción de una respuesta tomaba tiempo considerable.

Estos casos se presentaban cuando se requería de algún material como herramienta de trabajo, reparación de equipo de cómputo, entre muchas otras necesidades. El colaborador recurría al director responsable de área, para autorizar dicha solicitud mediante un documento impreso, estas solicitudes

eran elaboradas manualmente en Excel, haciendo una serie de pasos tales como:

- Imprimir formato de hoja con el nombre de la solicitud requerida (cómputo, compras, vehicular y general).
- Búsqueda de registros en la base de datos de Excel.
- Llenar la solicitud correspondiente a mano.
- Operaciones presupuestales (manualmente).
- Tiempo de respuesta (en espera).

Objetivos Generales y Específicos del Proyecto.

Actualizar el software para el funcionamiento y mejora de la sistematización de documentos por medio de una aplicación web que siga contribuyendo a las necesidades de los colaboradores de la Secretaría, continuar realizando solicitudes en el momento que lo requieran.

Utilizar Codeigniter para seguir aportando beneficios al sistema teniendo como objetivos específicos, lo siguiente:

- Agilizar los procesos y tareas.
- Brindar mayor confiabilidad de los datos.
- Brindar mayor soporte.
- Operar desde su área de trabajo.
- Precisión de los datos almacenados.

Metodología.

La metodología la cual estará basado este proyecto, es la metodología de Scrum [7], una de las metodologías actuales y utilizadas en muchas de las empresas, por su forma de trabajar en equipo, y de finalizar los proyectos en tiempo y forma.

El primer paso en esta metodología, es identificar las historias Backlog, propuestas por el usuario o cliente, historias que describen las necesidades del software [13].

De estos pasos generamos una pila de producto (pila de Sprint), será una pila priorizada, ya que sus elementos pueden ser elegidos dependiendo de la importancia que se le asigne a cada elemento de la pila y así poder trabajar en ella.

La planificación es una parte importante a la hora de empezar a trabajar con el proyecto, ya que Scrum maneja una serie de formas practicas de planificación como: Utilizar un tablero con tarjetas adheribles, la cual contiene las tareas a realizar del Sprint, aplicar alguna de sus técnicas de tiempo estimado (ojo de buen cubero, calculo de velocidad y técnica de estimación).

Se requiere una serie de pruebas en cada elemento de la pila, para verificar que la demo (pruebas del sistema) esté funcionando bien.

Infraestructura.

1. Utilizaremos como herramienta de trabajo un computador con procesador de gama alta, que soporte la instalación de ciertos programas.

2. Utilizaremos un software llamado Codeigniter [4], el cual posee las siguientes características:

    - Codeigniter es un framework; un programa para desarrollar otros programas
    - Es fácil de instalar en cualquier servidor.
    - Es un producto de código abierto (libre de uso para cualquier aplicación)
    - Esta desarrollado bajo el lenguaje de programación PHP.
    - Contiene una serie de librerías bien definidas que sirve para el desarrollo de aplicaciones web.

- Tiene una manera propia de clasificar sus diferentes scripts, que sirve para que el código este bien organizado.
- Utiliza una metodología MVC (Modelo Vista Controlador), es un estándar de programación de aplicaciones.
- Define una arquitectura ordenada para el desarrollo de páginas web.
- Tiene herramientas que ayudan a hacer aplicaciones más versátiles y seguras.
- Contiene funciones que son opcionales, para que el programador tenga libertad en el desarrollo de la página web.

3. Utilizaremos HTML5 [5] para la construcción del sitio web.

    - Tiene una estructura que posee los elementos necesarios para ubicar contenido estático o dinámico.
    - Su estructura debe proveer forma, organización y flexibilidad, para que su base sea fuerte.
    - Es una plataforma básica para aplicaciones
    - Posee variedad de dispositivos para acceder a internet
    - Diversidad de interfaces disponibles para interactuar con la web.
    - HTML5 posee como complemento a CSS (estilos) y JavaScript.

4. Utilizaremos un servidor XAMPP [8].
    - Funciona como servidor local, para uso de pruebas en desarrollo de páginas web.
    - Es un software libre.
    - Es un servidor local seguro.
    - Fácil de manejar.
    - Se adapta a cualquier S.O (sistema operativo), y adaptable en Apache, MySQL, PHP, Perl.

Resultados Esperados.

El sistema debe cumplir con lo siguiente:

- Flexibilidad; es decir, si en algún momento se requiere hacer cambios en el programa para mejorarlo, o agregar información nueva, se podrá hacer sin ningún problema.
- Podrá ser ejecutado en el momento que lo requieran, dependiendo del área donde se encuentre el colaborador dentro de la misma secretaría, ya que el sistema será vía Intranet.
- Será dinámico. Se podrán hacer diversas actividades en el mismo.
- Será versátil. Fácil de manejar y fácil acceso.
- El sistema debe de ser seguro. Cumplirá con todos los niveles de seguridad, para resguardo de los datos.

Capítulo 1 Se hará un estudio acerca del marco teórico, con la finalidad de reconocer la metodología y las herramientas que serán de utilidad en el proceso de desarrollo del proyecto. Estudiaremos con profundidad la metodología de Scrum para su implementación. Se hará un estudio sobre los diagramas que ofrece UML (Lenguaje Unificado de Modelado) [21] para el desarrollo del diseño del software; así como las herramientas de Codeigniter, HTML5, PHP, XAMPP en su funcionalidad y hora de su ejecución.

Capítulo 2 Se llevará a cabo el planteamiento del problema, aplicando la metodología Scrum en todo el proceso de desarrollo; se hará el modelo del diseño del software mediante el diagrama de casos de uso y el diagrama de clases que ofrece UML (Lenguaje Unificado de Modelado) y las herramientas de Codeigniter, HTML5, PHP, XAMPP con base a los estudios en el capítulo 1, haciendo pruebas de código, conexión a base de datos y diseño; para verificar la funcionalidad del software.

Capítulo 3 En este capítulo se realizará el prototipo del software; en la forma real del sistema, que ayudara a verificar su funcionalidad por medio de pruebas, que permitirá al usuario hacer uso del sistema (manual de usuario).

Capítulo 1 Agilización de procesos bajo la metodología de Scrum mediante el uso de MVC en Codeigniter.

La siguiente investigación es a cerca del estudio de las diferentes técnicas propuestas para el desarrollo de la sistematización; estudiar la definición de los diferentes conceptos y la importancia de la utilización de cada uno de ellos.

Por tal motivo se propone implementar un sistema para automatizar y agilizar los procesos. Para ello se estudiara el framework Codeigniter 3.1.7 [3], se implementará este framework por su arquitectura MVC (Modelo Vista Controlador) haciendo el desarrollo de la programación ordenada, rápida, fácil de crear, bajo el lenguaje de programación PHP [17]; trae diversas herramientas que ayudan a hacerlo, seguro, flexible y versátil.
Se estudiaran los diagramas de UML (Lenguaje Unificado de Modelado) [21], para el modelado de diseño del sistema.

Además del framework que se tiene pensado, se estudiara la metodología Ágil de Scrum [7] bajo el criterio de mejores formas de implementación, adaptándolo en todo proceso del proyecto.

1.1 Antecedentes de la metodología de Scrum.

Scrum [7] exhorta a conocer sus principios y ejecutarlo correctamente, siguiendo los pasos como lo indica sabremos identificar sus objetivos principales, crear una pila de producto o pila de historias, crear una pila de Sprint y su estimación, representándolas por medio de una gráfica de Burn-Down y hacer pruebas de producto.
El principal objetivo de Scrum es maximizar la velocidad de un equipo trabajando a la par, con estimaciones asignadas por el dueño de producto en tiempos cortos de entrega el cual derive en avanzar bajo las estrategias de esta metodología [7]. El alcance del tiempo estimado en cada iteración de la pila, será el objetivo principal para hacer un producto entregable. Vea el ejemplo de la figura 1.1.

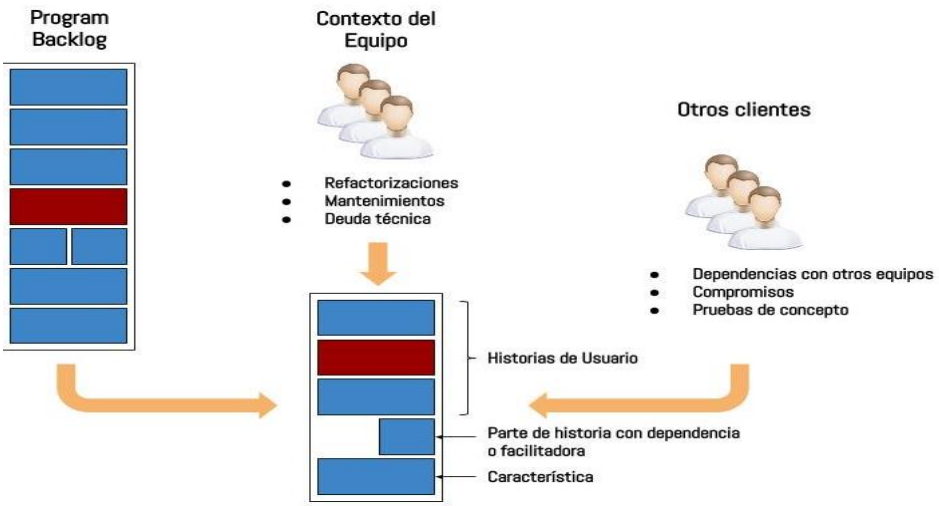

Figura 1.1 Metodología de Scrum.

1.1.1 Producto Backlog.

Este método enlista las principales características [13] del sistema a desarrollar convirtiéndolas en historias de usuario siendo el punto de inicio en toda la implementación, por consiguiente, la obtención de conjunto de historias bajo la terminología del cliente, haciendo una valoración que conlleve a priorizar cada historia, acorde a su importancia y al criterio de aceptación de un equipo Scrum, como lo muestra la figura 1.2.

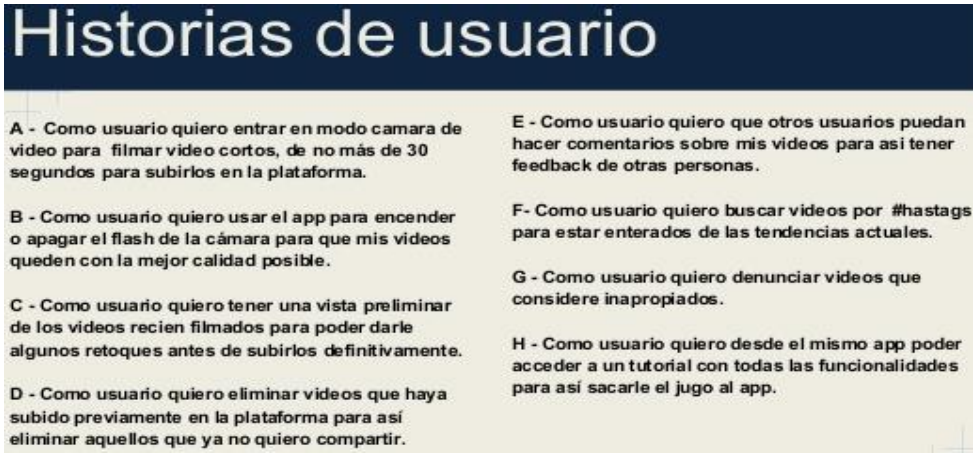

Figura 1.2 Historias de usuario.

1.1.2 Pila de producto y pila de Sprint

Acorde a las historias del producto Backlog y a la manera de priorizarlas formamos una pila de producto, cada historia en nuestra pila es un elemento.

Formamos una nueva pila, ahora será pila de Sprint [7], consiste en seleccionar historias de la pila de producto asignando un tiempo estimado y un puntaje que los definirá; ejemplo: historia A=15, historia B=20. El orden del Sprint depende de la importancia que se le asigne. Por tanto, conforme al conjunto de estimaciones formamos una pila de Sprint, como se ve en la figura 1.3.

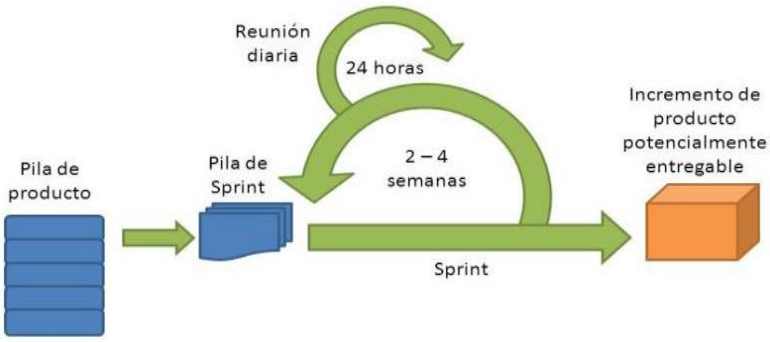

Figura 1.3 Pila de producto y pila de Sprint.

1.1.3 Planificación de Sprint

La planificación es el siguiente paso más importante para alcanzar el éxito en cuanto a tiempo [7], conforme a la velocidad del equipo, convirtiéndolo en producto entregable.

Tiempo estimado

- Nos percatarnos que la pila de producto esté perfectamente definida.
- Dueño de producto (Product Owner), es el responsable de hacer el producto entregable. Si no hay dueño, no hay producto.
- Scrum tiene como tarea que todo marche en orden siguiendo las reglas acordadas, impidiendo cualquier obstáculo que distraiga al equipo del objetivo.

- El equipo es quien define la estimación del Sprint, la importancia lo decide el dueño de producto.
- Es necesario que el dueño de producto asista a todas las reuniones y llegar a un acuerdo entre equipo, Scrum y dueño de producto, estas reuniones deben de ser constantes.
- Los ratos de importancia dedicado a cada elemento son cuestionados constantemente, siendo reordenados si es necesario debido a que las historias pueden ser divididas (A1, A2, A3…) en historias pequeñas.
- Una fecha de entrega de demo para cada Sprint.
- La pila de Sprint tiene que estar siempre actualizada, debido a los cambios.
- En la Programación Extrema (XP), la calidad en el código a entregar no se cuestiona, este asegura la funcionalidad del sistema, trabajando mediante el desarrollo guiado por pruebas, cuyo propósito es trabajar en el diseño.
- Una fecha de entrega de producto final de no más de 30 días.

1.1.4 Técnicas de estimación utilizadas en Scrum.

- Ojo de buen cubero: Es utilizado en equipos pequeños con Sprint cortos, es fácil llegar a un acuerdo entre dueño de producto y equipo sobre cuantas historias, podrían incluir y concluir en cierto tiempo. Ejemplo; agregar historias A, B, C, D, F… a la pila de Sprint con un tiempo estimado de 3 semanas para concluir.

- Cálculo de velocidad: Aquí se toma en cuenta la velocidad estimada del último Sprint y posteriormente tomarlo como modelo para estimar la velocidad siguiente; por tanto, se toman en cuenta 3 factores para calcularlo: días-hombres disponibles, factor de dedicación y velocidad real, la figura 1.4 muestra la fórmula para el cálculo de velocidad.

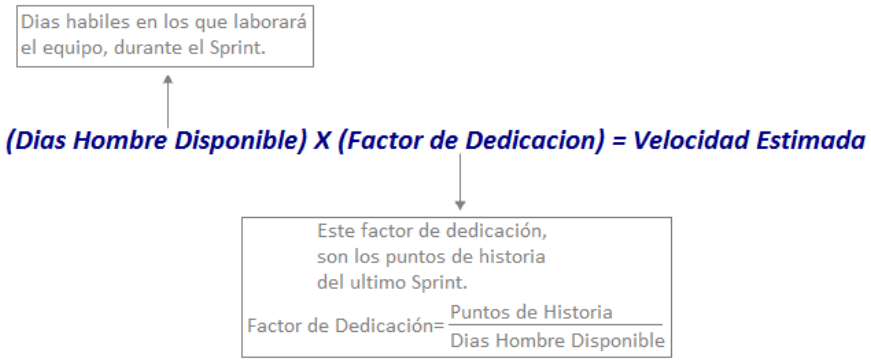

Figura 1.4 Fórmula para el cálculo de velocidad.

Por tanto, la planificación es la parte ardua en discutir que elementos acorde a su prioridad deben incluirse en la pila de Sprint, en donde el equipo debe tener toda la información necesaria, conocer perfectamente cada detalle sobre el proyecto y trabajar en ello sobre un corto tiempo, sin descuidar la calidad del producto, vea la representación de la figura 1.5.

Figura 1.5 Representación de la planificación de Sprint

1.1.5 Forma práctica de planificación de Scrum.

Hemos visto diversas formas en que Scrum hace su planificación de Sprint [7], por medio de reuniones entre Scrum Máster, equipo y dueño de producto; pero lo más lo factible es llevar a la práctica esta planificación, utilizando tarjetas que

especifican cada historia, sugiriendo adherirlas sobre un tablero o bien una pared como respaldo, se pueden manipular pegando y despegando variedad de tarjetas para reordénalas de acuerdo a la prioridad que asigne el equipo.
La forma de organizar estas tarjetas es utilizando un papel muy amplio al tamaño del tablero y crear campos para su orden, de esta manera visualizarlas mejor, queda abierta la opción de agregar campos adicionales solo si, lo crees necesario, recalcando que la simplicidad en esta implementación es nuestro propósito, tal como se observa la figura 1.6. La información cualitativa agregada en estos campos como:

- Pendiente: Son historias en espera.
- En curso: Historias en las que se está trabajando.
- Terminado: Historias que han concluido.
- Objetivo: Una gráfica para ver los avances por cada Sprint.

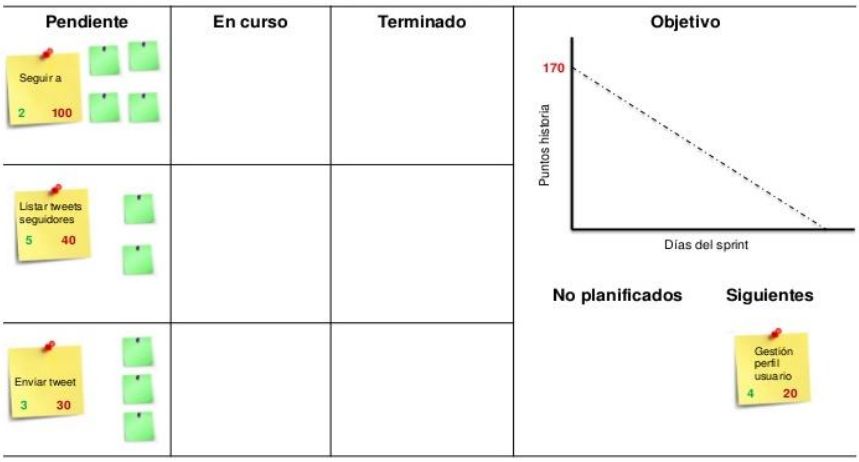

Figura 1.6 Tablón de tareas para la planificación de Sprint.

1.1.6 Diagramas de Burn-Down.

La planificación del tiempo estimado plasmada en forma gráfica es la mejor manera de visualizar que todo este marchando bien conforme al tiempo estimado de cada Sprint, tal como lo muestra la figura 1.7. El diagrama de Burn-Down [7], es un mediador de alerta en avisar si surgen problemas, la

planificación se verá afectado si se agregan más historias del que se tenía contemplado, esto podría estar sobrepasando el tiempo estimado, si esto ocurre optaremos por quitar algunos elementos de la pila; en caso contrario si hay tiempo de sobra, podríamos quizá agregar algunas historias al Sprint. Sin el diagrama no podríamos cerciorarnos de estos inconvenientes improvisados, que podría inducirnos a afectar completamente al sprint.

La gráfica debe ser calculada en días hábiles (días-hombre disponibles), para no descontrolar la planificación considerar la posibilidad de asistir algún fin de semana en casos muy especiales.

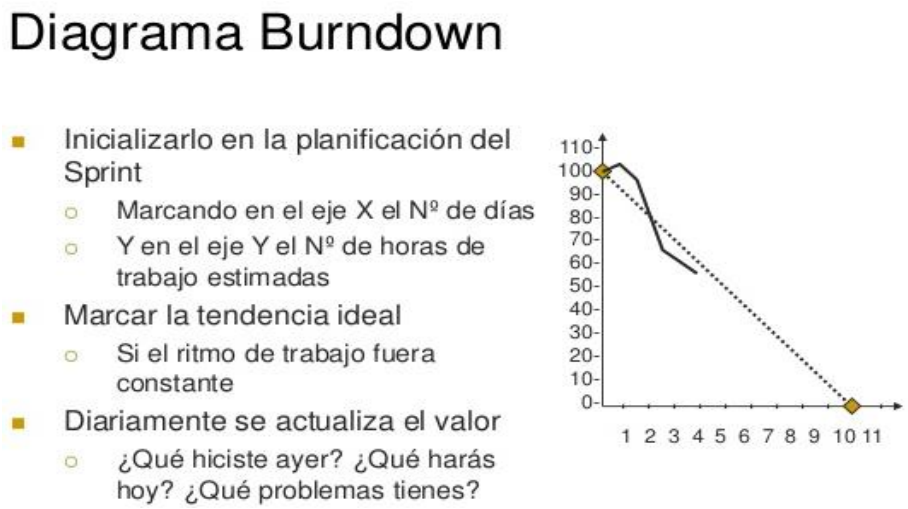

Figura 1.7. Grafica de Burn-Down

1.1.7 Demo de Sprint

Las demos son muy importantes [7], en ello visualizamos en forma real los avances de cada Sprint, la funcionalidad en cuanto a código; por medio estas pruebas, el equipo Scrum estará motivado por terminar el trabajo actual y empezar los siguientes Sprint. Hay que ser lo bastante ágiles y precisos en elaborar una demo sin demorarse tanto tiempo para no afectar la entrega.

1.1.8 Programación Extrema (XP)

La programación (código funcional) es una parte fundamental en el desarrollo de producto, debido a su importancia de XP es considerado en la planificación de Scrum [7], esto significa contemplar tiempo estimado. Si bien Scrum Xp son temas muy diferentes pero que ambas funcionan perfectamente y por ende se obtienen resultados satisfactorios que además interceden en el diseño y finalmente destinadas a una entrega de producto funcional de Sprint.

En XP utilizamos el método TDD (Test Driven Development), su significado en español desarrollo guiado por pruebas difiere en diversos aspectos, como hacer pruebas de programación, manejo de base de datos y servidores como el acceso a HTTP y enfocado principalmente a las pruebas de seguridad, como se ve en la figura 1.8. Es sugerible que dentro del mismo equipo Scrum haya una persona con la habilidad de verificar la funcionalidad en cada proceso mediante el servidor de pruebas y determinar si la prueba de código es totalmente funcional o aún carece de elementos.

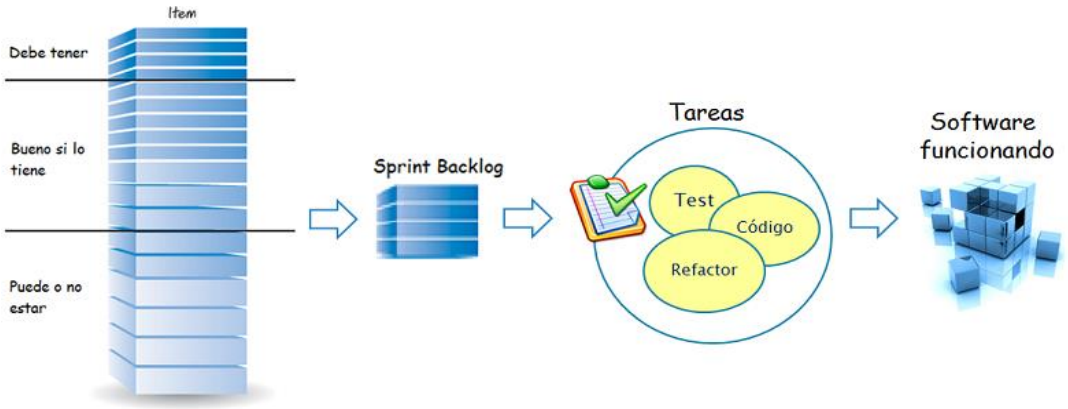

Figura 1.8 Desarrollo guiado por pruebas.

En efecto la TDD difiere principalmente en el diseño, verificando cada prueba que va desde las pruebas manuales hasta las pruebas de código para la mejora constante; seguido del escrutinio de la persona encargada de pruebas y del equipo de Scrum, sirviendo como filtro antes de cada entrega en los avances a usuarios finales.

Durante la XP es importante la participación del equipo resaltando sus habilidades en cuanto a organización para trabajar sobre el proyecto, pero sin antes olvidar la intervención del Scrum , ya que de él depende, la organización del equipo y de acatar las reglas durante el desarrollo del proyecto, la meta alcanzada por cada Sprint dependen de la responsabilidad del dueño del producto y entrega final al cliente [7].

1.2 Conocimientos básicos de Codeigniter

Codeigniter se caracteriza por su arquitectura (MVC) y versatilidad [4]; es una aplicación web adaptable para trabajar en cualquier plataforma de desarrollo y compatible con diferentes servidores, además de su fácil instalación, hace que sus elementos sean propios e interactúen entre sí para facilitar el proceso de desarrollo, por ello la importancia de tener presente sus características y conceptos básicos, para el buen manejo de sus componentes.

Codeigniter es un programa para desarrollar otros programas (Framework) de código abierto (Open Source), bajo el lenguaje de programación PHP, es muy específico en utilizar sus herramientas como sus librerías, Scripts, código, servidores, etc. Esta aplicación es muy ligera, no se sobrecarga el núcleo por la cantidad de código generado, solo carga los componentes que son seleccionados por el programador, haciendo este framework aún más complejo.

1.2.1 Arquitectura MVC (Modelo Vista Controlador)

Codeigniter utiliza la arquitectura MVC (Modelo Vista Controlador) [4], que facilita el manejo de sus herramientas. El proceso de desarrollo va desde la perspectiva del cliente hasta los procesos internos, utilizando las herramientas que ofrece este framework. Vea la representación de flujo de aplicación en la forma de interactuar con sus diferentes componentes que lo caracterizan, tal como se ve en la figura 1.9.

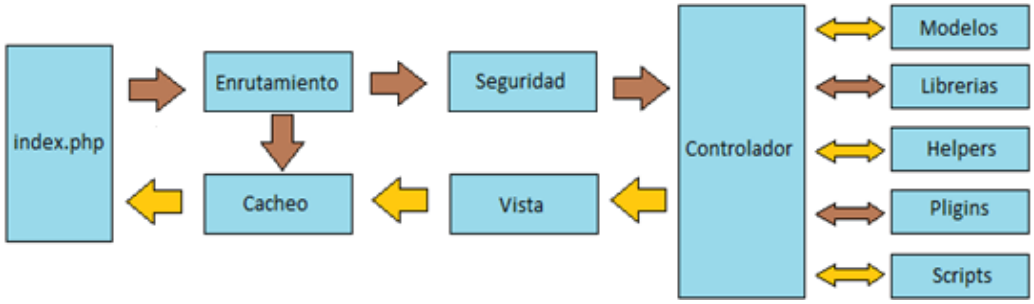

Figura 1.9 Flujo de aplicación de Codeigniter.

Todo usuario que consulte una página, inicia desde **index.php** siendo este la raíz de del sitio; el **enrutamiento** es el proceso de filtrado verificando la URL para saber que elemento debe de ser procesado y si el archivo ha sido generado anteriormente caerá en la **cache**, en caso contrario sigue con el proceso de seguridad, esto evitará redundancia de datos; siguiendo con el proceso de consulta, se hace un tratamiento de **seguridad** de los datos que posee la URL así como los POST que están ingresando a la página; el **controlador** se comunica con una serie de **módulos** (modelos, librerías, plugins, etc.), este procesamiento es atreves de las **vistas**, para finalmente producir un sitio web y que será enviada al navegador [4].

1.2.2 Proceso del Modelo Vista Controlador (MVC).

El MVC siendo un estándar de programación de aplicaciones es utilizado en otros frameworks (Laravel, Zend Framework, Ruby On Rails, etc.) por su organización en el código fuente, agrupándolo en tres partes. Los controladores, modelos y vistas, son desarrollados por los programadores en donde se puede modificar dicho código interactuando con los diferentes módulos (Librerías, Helpers, Plugins, Scripts, etc.) sin olvidar que estos son independientes.

MODELO: Recoge y organiza la información dentro de una base de datos, es la parte donde podemos insertar y eliminar información, hacer llamados a las funciones (consultas SQL) que necesitemos dentro del programa, debido a su encapsulamiento de complejidad de datos [4].

El manejo de una base de datos depende del programador, para decidir si queremos utilizar las herramientas que ofrece Codeigniter o programar de la forma tradicional, desde la creación de tablas y manipulación de la información contenida en ellas, ya que por la flexibilidad que tiene este framework puede procesar el código en cualquier forma.

Declarar un modelo en Codeigniter.

Codeigniter recomienda utilizar los modelos para gestionar la conexión a la base de datos, ya que la gestión es más sencilla y puede ahorrarnos líneas de código. Lo que se maneja en estos modelos son las clases (class) y funciones (function), las cuales serán invocadas por los controladores [3].

- La forma de declarar un modelo es extendiendo la clase model y con la primera letra en mayúscula, la sintaxis es: *class Nombre_model extends Model {*
- Por consiguiente declaramos un constructor muy necesario en los modelos, la siguiente sintaxis: *function __construct (){*
- Una vez construido el modelo guardamos el archivo con un nombre cualquiera, con todas las letras minúsculas y con la extensión. php, dentro de esta carpeta system/application/models/
- La forma de cargar un modelo es *$this->load->model('Nombre_modelo').*
- Para hacer la conexión del servidor o base de datos a Codeigniter es hacer una configuración en el archivo autoload.php desde system/application/config/autoload.php, el cual debe quedar de esta forma: $autoload['libraries'] = array ('**database**', '**otras_librerias...**").
- Y otra modificación en el archivo database.php para colocar los datos desde de la base de datossystem/application/config/database.php, quedando de la siguiente forma:

```
$db['default']['hostname'] = "localhost";
$db['default']['username'] = "root";
$db['default']['password'] = "";
$db['default']['database'] = "nombre_base_de_datos";
$db['default']['dbdriver'] = "mysql";
```

```
$db['default']['dbprefix'] = "";
$db['default']['pconnect'] = TRUE;
$db['default']['db_debug'] = TRUE;
$db['default']['cache_on'] = FALSE;
$db['default']['cachedir'] = "";
$db['default']['char_set'] = "utf8";
$db['default']['dbcollat'] = "utf8_general_ci";
```

Un dato muy importante que se debe mencionar es que dentro de un modelo puede haber diversas funciones, que más tarde serán gestionadas por los controladores.

VISTA: Es la interfaz o vistas de usuarios, codifica y preserva la presentación final [3], es decir la consulta que hace el usuario y que visualiza como una respuesta. En esta parte la codificación tiene que ver con HTML el cual dará presentación a la página escrito en Javascript y estructurado por CSS para el manejo de párrafos, imágenes, enlaces etc. Las vistas son módulos invocadas directamente por los controladores y deciden el formato de cómo desea mostrar la página al usuario. Las siguientes líneas son una sintaxis sencilla y muy general para saber cómo se declara una vista en HTML.

```
<html lang="es">
<head>
<title>Bienvenido</title>
</head>
<body>
<h!>Bienvenido a mi web</h1>
</body>
</html>
```

Esta presentación de página lo guardaremos con un cualquier nombre, en esta carpeta: *system/application/views* y con la extensión PHP, la forma de invocar a una vista desde el controlador es: $this->load->view('Nombre_de _la_Vista'.

Debido a que las vistas forman la estructura de una página, dentro de este se puede colocar código PHP y al mismo tiempo trabajar con HTML, con el fin de conservar el código y mostrar al cliente solo la interfaz [3].

Declarar código PHP

El lenguaje de programación PHP [2], [19] y a su vez un lenguaje de scripts es un lenguaje muy poderoso y popular; que son ejecutados en el servidor, y es ejecutado en diversas plataformas (Windows, Linux, Unix, Mac OS X). Cuando se crea un archivo de código se guardan con extensión ".php", al ser ejecutados devuelven el resultado como HTML simple.

PHP puede recopilar información de un formulario, tiene acceso a la base de datos para agregar, eliminar y modificar los datos; además pude abrir, cerrar, crear, borrar, escribir y leer archivos en el servidor; pero sobre todo puede cifrar datos.

Sintaxis:

```
<!DOCTYPE html>
<html>
<body>
<h1>My first PHP page</h1>
<?php
echo "Hello World!";
?>
</body>
</html>
```

CONTROLADOR: Es la parte lógica del programa donde se guardan las páginas que han sido realizadas y encargado de enlazar los modelos y vistas [3]. Del controlador depende que se genere la página web debido a los diferentes recursos que gestiona en el servidor y apoyándose de la gestión de los diferentes modelos y vistas. Los controladores son guardados en esta carpeta: *system/application/controllers/* y para acceder a ellos es por medio de http://localhost/index.php/nombre_del_controlador; edemas cuando creamos el controlador, la primera letra debe de ser en mayúsculas y las demás en minúsculas, ejemplo: class *Prueba extends Controller{*

Los modelos y vistas podrían ejecutarse directamente en el controlador, pero es muy recomendable seguir la arquitectura (mvc) para aprovechar los recursos que ofrece Codeigniter.

1.2.3 URLS en Codeigniter.

Las URLS son generadas al crear una aplicación que provienen de la raíz index.php, encargadas del enrutamiento. Deben identificar al controlador que trabajará en la gestión de datos de la página, para procesar la solicitud, por medio de una función (código fuente) y un parámetro (identificador de la función) [3], como se ve en la siguiente imagen 1.10.

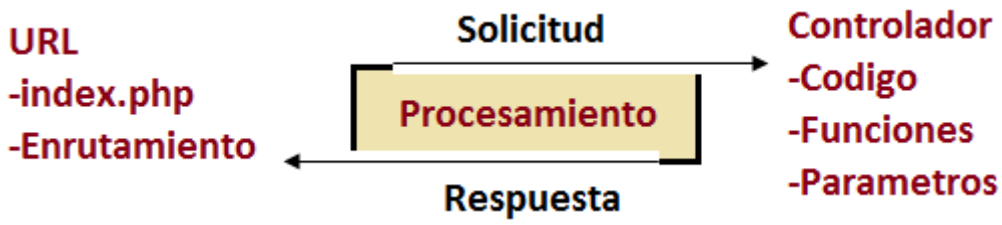

Figura 1.10 Gestión de una URL.

Estos controladores y funciones se les asignan un nombre cualquiera, para ser reconocidas fácilmente por la URL, ya que en un controlador se pueden asociar varias URLS. Vea la siguiente sintaxis en una forma sencilla de declararlos.

class Prueba extends Controller{

Prueba es el nombre del controlador que estamos creando.

function ejemplo($ejemplo_parametro){

Función del controlador, y declaración del parámetro

http://localhost/index.php/prueba

Esta URL es la dirección donde se encuentra el controlador.

Una forma de entender cómo actúa el MVC es que los controladores atenderán siempre la solicitud de una página mientras que las vistas se encargarán de mostrar la apariencia de la página.

Por la flexibilidad que tiene Codeigniter podemos hacer una serie de cambios en el código y personalizarlo según sea la necesidad y facilite la programación por mencionar un caso como borrar el index.php para hacerlo más directo, entre muchos otros beneficios, estos cambios se hacen desde las carpetas **system** y **application**, contenidas en el framework.

1.2.4 Servidor local XAMPP 3.2.2

Para la utilización de codeigniter, debemos tener instalado un servidor como lo es XAMPP [8], [20] que es un servidor local, fácil de utilizar y es un software libre, lo podemos obtener desde la página oficial *https://www.apachefriends.org/es/index.html* descargando la versión más actualizada, en este caso descargamos la versión 3.2.2, como lo muestra la figura 1.11.

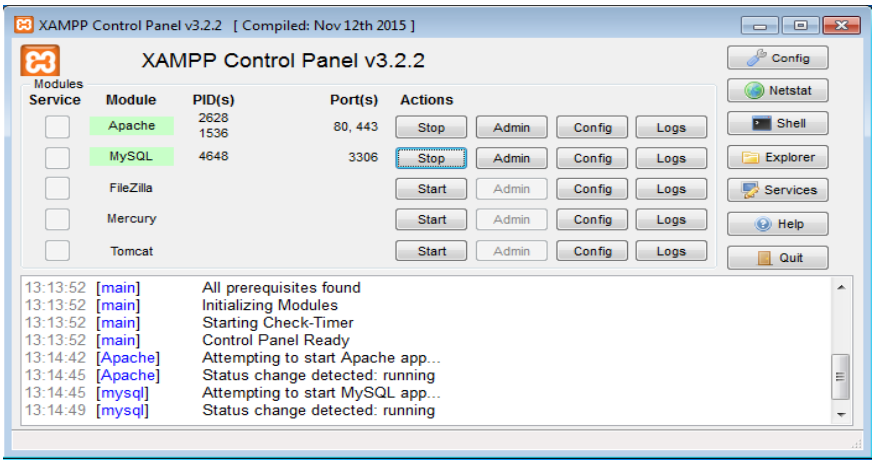

Figura 1.11 Instalación del servidor local XAMPP 3.2.2.

El servidor es primordial para cualquier framework, la elección del servidor dependerá del S.O que maneje (Linux, Windows, Apple, etc.) y de las características que tenga su equipo, es decir, si es para 32 o 64 bits.

1.2.5 Instalación de Codeigniter 3.1.7

Codeigniter [4] además de ser un software libre, de fácil acceso también es fácil de instalar. Para su utilización lo podemos descargar desde el sitio oficial *https://codeigniter.com/download,* descargando su versión más actualizada, en este caso utilizamos la versión 3.1.7.
Una vez descargada la carpeta de Codeigniter esta debe ser depositada en la carpeta ***htdocs*** contenida en el servidor, como lo muestra la figura 1.12.

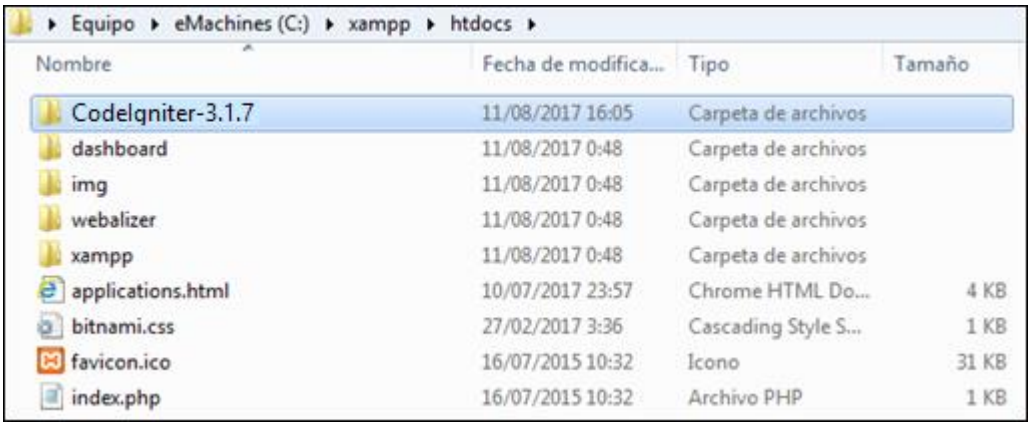

Figura 1.12 Ubicación de la carpeta Codeigniter en el servidor XAMPP.

Una vez colocada la carpeta dentro del servidor estará lista para ser utilizada; puede cambiar el nombre de la carpeta si lo desea, ya Codeigniter no es más que una carpeta con una estructura bien definida y con una configuración interna para su funcionamiento.

1.3 Utilización de HTML5.

HTML5 [5], [18] es una nueva plataforma para desarrollar páginas web, siendo popular actualmente por sus elementos que lo hacen más avanzado, es importante aclarar que HTML5 no es una nueva versión de HTML, cada uno es un tema distinto, en lo que ambos coinciden es en el propósito de crear la estructura de un sitio web.
La manera de trabajar con HTLM es una forma básica en organizar y compartir información con la web por medio de su lenguaje de texto; en su momento Java

y Flash fueron un éxito por trabajar en conjunto con HTLM, pero el mayor inconveniente fue la falta de integración, por lo que pronto dejo de ser bueno, y por ende se buscaron nuevas metodologías y lenguajes de programación más avanzadas.

Lo que abrió camino a la evolución de los navegadores fue la Web 2.0, impulsado por Javascript, debido a este giro radical los desarrolladores y diseñadores web interesados en seguir innovando lograron la perfecta combinación entre HTML, CSS y Javascript creando una implementación nueva y más completa conocida como HTLM5 [5], [18].

1.3.1 Adaptación de HTML5

HTML5 es mucho más que una simple estructura; lo que se puede lograr con esta implementación es la combinación de aplicaciones con dispositivos móviles, computación en la nube y trabajos en red. Es importante mencionar que no todos los navegadores soportan esta plataforma, entre los navegadores que más lo soportan es Internet Explorer, Mozilla Firefox y Google Chrome, lo más conveniente es utilizar las últimas versiones para que la aplicación que vamos a desarrollar no tenga ningún inconveniente [5].

1.3.2 Características de HTML5

Es esencial conocer los conceptos generales sobre las características que definen a HTML5 como son: Estructura, estilo y funcionalidad proporcionadas por sus componentes (HTML, CSS, JavaScript) [5], así como los elementos que posee cada uno, que servirán de ayuda a programadores para la creación de diversas interfaces y plataformas para distintas aplicaciones de dispositivos e interactuar con la web.

1.3.3 Conceptos básicos de HTML (estructura).

HTLM desde el concepto de HTML5 utiliza un lenguaje de **etiquetas** que definen cada **elemento** de la estructura, dichas etiquetas se encargan de la organización estructural, con la finalidad de proveer forma y flexibilidad, para

crear fundamentos muy fuertes como los cimientos de un edificio [5]. En cualquier estructura básica de una página en HTML, debe constar de cabecera, cuerpo y pie de página; el ejemplo de la figura 1 muestra cómo se declaran las etiquetas dentro del código HTML en un concepto básico.

Para la construcción de una página web por HTML debemos tener presente el concepto de algunas etiquetas básicas derivado del lenguaje de programación a utilizar y de esta manera ir definiendo el documento (estructura global) [5].

- Declaramos **<!DOCTYPE html>** para especificar qué tipo de documento vamos a crear e indica que es la primera línea del archivo. También avisa al navegador que deberá utilizar código HTML5.
- Los archivos creados en HTML serán finalizados por la siguiente etiqueta: **</ html>.**

Diversas etiquetas utilizadas en HTML su apertura será declarada con **<>** y con **</>** para cerrar dicha etiqueta.

- Para empezar a construir la estructura declaramos la etiqueta **<html>** para abrir y para cerrar declaramos **</html>** el cual se encargará de devolver todo el código escrito contenido en él.
- Dentro de la apertura del código HTML declaramos <html **lang=**”es”>, en donde el atributo **lang** indica el inicio **y** el atributo **“es”** define el idioma, en este caso indica que es el idioma español.

Dentro de las etiquetas <html> declaramos dos secciones que son cabecera, cuerpo y cierre de página; declarándolos con **<head>** y **<body>.**

- La etiqueta **<head>** es utilizado para declarar el título del documento (cabecera); en esta parte incorporamos archivos externos con estilo declaradas por otras etiquetas. La información que va dentro de estas etiquetas será ocultas para el usuario.
- La etiqueta **<body>** es la parte principal, utilizado para declarar el cuerpo del documento lo declarado dentro de esta etiqueta será visible para el usuario.

```
<!DOCTYPE html>
<html lang="es">
<head>
// Etiquetas que definen el titulo del documento.
</head>
<body>
// Etiquetas que definen el cuerpo del documento.
 <footer>
// Etiqueta para finalizar el cuerpo del documento.
</footer>
</body>
</html>
```

Imagen 1.13 Estructura global de HTML declarado por etiquetas.

La forma de utilizar las diversas etiquetas que tiene HTML5 será colocada dentro de las etiquetas principales (cabecera, cuerpo y pie de página), dependiendo de la función que tenga cada una. Viéndolo desde un panorama de una estructura de árbol que empieza desde **<html>** como la raíz; vea la representación de la figura 1.13

Definiendo la cabecera del documento.

- **<meta>** es colocada dentro de la cabecera, define el juego de caracteres especifica cómo el texto será presentado en pantalla, por ejemplo **<meta charset=”iso-8859-1”>**
- **<title>** es colocado dentro de la cabecera y especifica el título del documento que estamos creando, por ejemplo **<title>Este texto es el título del documento</title>**
- <link> es colocado dentro de la cabecera, es utilizado para incorporar estilos (estilos CSS), códigos Javascript, imágenes o iconos desde archivos externos.

Definiendo el cuerpo del documento.

- **<header>** es usado dentro del cuerpo o secciones específicas dentro del cuerpo, para proveer información introductoria como títulos, subtítulos, logos etc.

- **<nav>**utilizada para generar la barra de navegación, es ubicado después de la etiqueta de cierre de la cabecera, pero que pertenece a el cuerpo (<body>).
- **<section>** define y especifica secciones, las cuales pueden ser divididas en varios bloques o columnas, utilizado para mostrar información más relevante del documento (información principal).
- **<aside>**define una columna o sección (barra lateral), utilizado para mostrar datos relacionados con la información principal.

Cerrando el cuerpo del documento.

- Etiqueta **<footer>** utilizado para finalizar el cuerpo del documento (pie de página).

Es importante destacar que las etiquetas mencionadas solo son algunos datos básicos, ya que HTML5 utiliza diversas etiquetas para ir construyendo la estructura (cabecera, barra de navegación, cuerpo, barra lateral, barra institucional etc.), como se ve en la representación gráfica de la figura 1.14. Tal es el caso del cuerpo del documento en donde se tiene que ir definiendo el contenido y así con cada parte del documento que definirá el diseño que se quiere lograr [5].

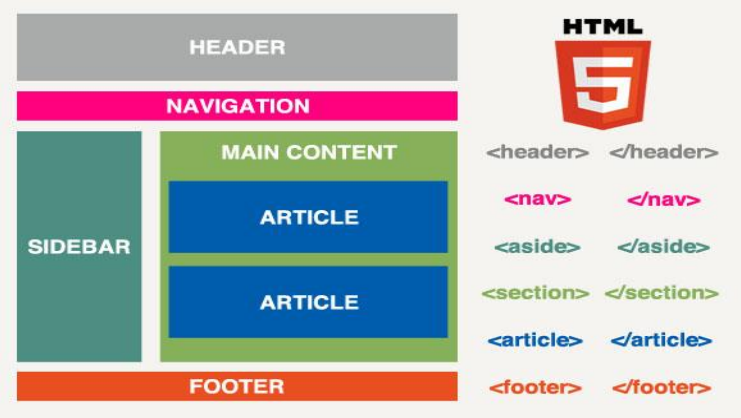

Imagen 1.14 Estructura básica de HTML5.

Algunas de las etiquetas de HTML han sido sustituidas, ya que se han integrado nuevos atributos y etiquetas (elementos) mucho más complejas definidas por CSS y Javascript, que han sido desarrollados por HTML5 con la finalidad de simplificar, especificar y organizar el código [5].

1.3.4 Conceptos básicos de CSS (hojas de estilo en cascada).

El deber de CSS es trabajar en el diseño para crear estilos siguiendo un grupo de reglas simples llamadas **modelos o sistema de disposición**, son reglas que debemos seguir, apropiadas para cambiar la apariencia de una página web como tamaño, texto, color, fondo, etc. Para después presentar esa estructura y diseño HTML en pantalla. [13]. El propósito de CSS es proveer diseño y flexibilidad.

Algunos detalles sobre los navegadores.

Los navegadores crean estilos por defecto, quedando lejos de las expectativas del diseñador, pero ahora podemos sobrescribir los estilos previstos por los navegadores por medio del set predeterminado de propiedades de CSS, que nos ayudan a obtener el estilo deseado; para ello debemos seguir un grupo de reglas que conforman un modelo de caja [13].

Los navegadores consideran cada elemento HTML como una caja (cada elemento presentado en pantalla es considerado como una caja), así que una estructura completa, es un grupo de cajas.

La forma de mostrar los elementos en pantalla es: por block (bloque) o por inline (línea); en donde los **elementos block** son posicionados uno sobre otro hacia abajo en la página (por lo regular estos elementos son los primeros en ser tratados por el navegador) y los **elementos inline** son posicionados uno al lado del otro, sin ningún salto de línea debido a que no hay más espacio horizontal para ser ubicados [13].

Puntos básicos para utilizar estilos CSS.

- Insertar estilos CSS en la cabecera del documento, con la intención de afectar los elementos HTML, por medio de referencias.

- Utilizar archivos externos para almacenar todos los estilos CSS y después llamar el archivo por medio de **<link>**e insertarlo dentro del documento requerido, de lo contrario los estilos podrían repetirse en todos los documentos.
  En este ejemplo; con la línea *<link rel="stylesheet" href="misestilos.css">* el navegador carga el archivo *misestilos.css* (son guardados en la misma carpeta donde se encuentra el archivo HTML) ya que contiene todos los estilos solicitados, para presentar el documento en pantalla.

- Algunos de los métodos para seleccionar que elementos de HTML serán afectados por CSS, es siguiendo algunas de las siguientes reglas:

  - Referencia por la palabra **clave** del elemento. Su función es hacer referencia a todos los elementos con dicha clave. Por ejemplo, con la línea *p {font-size: 20px}* cambiar a todos los estilos de todos los elemento **p** encontrados en el documento.

  - Referencia por el atributo **id**. su función es hacer referencia a un elemento HTML en específico, esto evitará redundancia. Por ejemplo, *#texto1 { font-size: 20px }* y hacemos referencia con *id="texto1".*
  - Referencia por el atributo **class**. Su función es hacer referencia a cualquier elemento HTML que queramos, es recomendable utilizar **class** en lugar de **id**, ya que es más flexible. Por ejemplo, *texto1 { font-size: 20px }* con referencia a *class="texto1".*

Para tener más claro concepto de CSS es que trabaja con un modelo de caja tradicional, con el propósito de aplicarlos a la estructura HTML el cual se verá afectado en forma adecuada (siguiendo reglas propuestas por CSS), lo cual se

habla de adaptarlos a los documentos HTML o bien al modelo de caja seleccionado. Cabe mencionar que, al trabajar con el modelo de caja tradicional, no estamos trabajando con los elementos esenciales de HTML5.

CSS es el camino hacia la definición de CSS3, ya que es un componente más avanzado con herramientas mucho más complejas, el trabajo en conjunto de CSS3 con HTML hace referencia a utilizar nuevos elementos definidos por HTML5 [13].

1.3.5 Conceptos básicos de CSS3 (estilos).

CSS3 no solo se refiere al manejo de diseño y estilo, debido a su desarrollo al paso de los años ahora CSS3 es capaz de dar forma y movimiento (Esquinas redondeadas, sombras, hasta transformaciones de los elementos) a una página web [13], dando un cambio radical y dejando muy por debajo de las expectativas que ofrece CSS.
El propósito de CSS3, es aplicar sus propiedades sobre una misma plantilla, y de esta forma lograr efectos visuales, para ello necesitamos estudiar cada una de sus propiedades y tener claro sus conceptos básicos.

- **Border-radius.** Esta propiedades el encargado de dar el efecto de esquinas redondeadas a una caja, esta herramienta trabaja con valores para definir las cuatro esquinas de la caja, por ejemplo.

  -moz-border-radius: 20px 10px 30px 50px;
  -webkit-border-radius: 20px 10px 30px 50px;
  border-radius: 20px 10px 30px 50px;

- **Box-shadow.** El efecto de esta propiedad es dar sombra a la caja, utiliza el prefijo **rgb()** para dar color y una palabra clave **inset** para convertir una palabra externa en interna, y así como las demás propiedades asignar valores, esta función puede tomar hasta cinco parámetros.

-moz-box-shadow: rgb(150,150,150) 5px 5px;
-webkit-box-shadow: rgb(150,150,150) 5px 5px;
box-shadow: rgb(150,150,150) 5px 5px;

- **Text-Shadow.** El efecto de esta propiedad es dar sombras a los textos, utiliza cinco valores que son: color, desplazamiento horizontal, vertical y difuminación. Por ejemplo, *text-shadow: #000000 5px 5px 10px.*
- **@font-face.** Esta propiedad permite a los diseñadores utilizar sus propios textos, por medio de una carpeta que contendrá la fuente específica y adecuada, elegida por el diseñador, por ejemplo*: @font-face{ font-family: Mifuente; src: url('font.ttf') }.*Esta propiedad requiere de dos estilos **Font-family** el cual especifica el nombre que asignaremos a la fuente elegida, y **src** que indica la dirección UR del archivo donde se encuentra la fuente que vamos a utilizar.

- **Gradiente lineal.** Esta función es utilizada para ser aplicada a las propiedades como **background** o **background-image** y de esta manera obtener un gradiente lineal; la sintaxis para declararlo es el siguiente *linear-gradient(posición inicio, color inicial, color final),* junto con la función *linear-gradient().*Los valores que utiliza son en pixeles.

- **Gradiente radial.** Esta función es utilizada para ser aplicada a las propiedades como **background** o **background-image** y de esta manera obtener un gradiente radial;la sintaxis para declararlo es la siguiente *radial-gradient(posición inicio, forma, color inicial, color final).*Los valores que utiliza son en pixeles

- **Rgba.** Esta función **rgba()** toma cuatro valores para definir el color, como rojo, verde, azul y la opacidad. Por ejemplo, rojo (0-255).

- **Hsla.** Esta función **hsla()** toma cuatro valores para definir el tono, saturación, luminosidad y la opacidad.

- **Outline.** Trabaja junto con la propiedad llamada *outline-offset*, generando un segundo borde. Por ejemplo, *outline: 1px solid#000000; outline-offset: 10px.*

- **Border-image.** Esta propiedad crea un borde con una imagen personalizada y se declara con las propiedades **border** o **border-with,** el cual requiere de tres valores; URL de la imagen, tamaño de las imágenes para construir el borde y para ubicar las piezas utiliza una palabra calve cualquiera, por ejemplo, *border-image: url("file.png") 15 stretch.*

- **Transform.** Esta propiedad modifica la forma de un elemento utilizando cuatro funciones **scale ()**, **rotate ()**, **skew ()**, y **translate ().**

- **Transition.** Estas propiedades utilizadas para crear una transición entre dos estados de un elemento, utilizando cuatro parámetros: la propiedad afectada, tiempo de transición utilizando una clave y un valor de retardo. Ejemplo, *transition: color 2s linear 1s.*

Una vez declarando las propiedades y funciones, podemos experimentar con ellas, asignando diferentes valores y lograr el efecto que mejor nos convenga, que va desde la creatividad del diseñador y sin olvidar que depende del navegador utilizando los prefijos **-moz** y **–webkit** para que funcione en navegadores basados con motores Gecko y WebKit (Firefox, Safari y Google Chrome).

1.3.6 Conceptos básicos de JavaScript (funcionalidad).

JavaScript [14] es un lenguaje que ha sido mejorado en muchos aspectos uno de ellos es acelerar su procesamiento de código, gracias a los nuevos motores para su interpretación de código máquina. Debido a su portabilidad e integración JavaScript se ha trasladado hacia los navegadores, sirviendo como un componente de HTML5. JavaScript se define por el manejo de archivos y funciones externos que son invocados más tarde por HTML [5] [14].

Para ir incorporando JavaScript a HTML es necesario definir algunos conceptos básicos y técnicas (en línea, manejadores de eventos, archivos externos) [14] que utiliza JavaScript; una de las técnicas más recomendable de usar es la inclusión de archivos externos, para el funcionamiento de HTML5.

En línea. Se ejecuta cuando un usuario realiza alguna acción, los cuales son procesados por manejadores de eventos (ratón, teclado, etc.) que ejecutan código JavaScript o funciones, utilizando los atributos de los elementos HTML que están disponibles en ese momento.

Embebido. Agrupa código (funciones) entre etiquetas (**<script>)**, de esta manera quedarán organizados con el propósito de afectar los elementos HTML por medio de referencias. Se recomienda colocar el código en la cabecera del documento, y después referenciarlo cuando se invoca una función para posiciónalo en cualquier lugar del documento.

Métodos de Javascript para referenciar los elementos HTML.

- **getElementsByTagName.** Por medio de palabra clave o por su nombre.
- **getElementById.** por identificador **id**, es decir por el valor de su atributo.
- **getElementsByClassName.** Por el valor de su atributo **class.**

Archivos externos. Cuando se crean nuevas funciones (códigos embebidos) se agregan nuevos métodos etc. hacen que el código JavaScript crezca y en consecuente el tamaño del documento aumente y a su vez el código se haga repetitivo, para ello es recomendable organizar el código en uno o más archivos externos e invocarlos cuando sea necesario por medio del atributo **src**, por ejemplo *<script src="micodigo.js"></script>,* donde el archivo externo es "micodigo.js".

Utilizando nuevos métodos para referenciar HTML.

- **querySelector().** Retorna el primer elemento que concuerda con el grupo de selectores específicos.

- **querySelectorAll().** Retorna todos los elementos que concuerdan con el grupo de selectores específicos.

Manejadores de eventos.

Un manejador de evento es aquella acción que realiza el usuario y son procesados por manejadores de eventos (presionar una tecla, clic en el ratón etc.) y funciones JavaScript asociados a ellos [5]. Veamos las tres formas de registrar un evento para elemento HTML.

Manejadores de eventos en línea. Se refiere a registrar eventos para un elemento en particular, aprovechando los atributos disponibles por HTML, aunque esta forma no es muy utilizada, no deja de ser buena, en su particularidad es utilizada en circunstancias especiales.

Manejadores de eventos como propiedades. Elegimos algún elemento HTML y se le asigna un manejador de evento, para referenciarlo se usa un selector de evento, como si fuese una propiedad. Estos eventos deben de ser registrados desde el código JavaScript, para evitar complicaciones.

El método addEventListener(). Este método tiene tres argumentos para ser utilizado: el nombre del evento, la función a ser ejecutada y un valor (falso, verdadero).

APIs (Application Programming Interface - Interfaz de Programación de Aplicaciones).

Debido a que JavaScript es un lenguaje de programación profesional y gracias a su portabilidad e integración y a la estrecha relación que tiene para interactuar con la web, es posible incorporar algunas APIs (interfaces de programación de aplicaciones) [5] para lograr nuevas funciones a través de técnicas de programación y facilitar a los programadores a utilizar herramientas muy complejas, como librerías para crear motores 3D para videojuegos, elementos grafios, etc.

Librerías que se utilizan a través de las APIs.

- **Canvas**. Utilizados para la creación de gráficos.
- **Drag and Drop.** Tiene el control de arrastrar el ratón y soltarlo.
- **Geolocation:** Tiene la intención de proveer información correspondiente con la ubicación física con la ubicación física del dispositivo que está accediendo a la aplicación.
- **Web Storage:** Almacena datos en el ordenador del usuario introduciendo dos atributos **sessionStorage** y **localStorage**.
- **Indexed Database:** Provee una base de datos destinada a aplicaciones web donde el usuario puede trabajar utilizando esa base de datos.
- **File:** Tiene la capacidad de leer, escribir y procesar archivos de usuario.
- **XMLHttpRequest Level 2:** Utilizada para la construcción de aplicaciones ajax, con nuevos métodos, para controlar el progreso de la operación y realizar operaciones cruzadas.
- **Cross Document Messaging:** Permite a aplicaciones comunicarse entre sí de diferentes cuadros o ventanas.
- **WebSockets:** Utiliza un mecanismo de dos vías entre clientes y servidores para generar aplicaciones en tiempo real.
- **Web Workers:** Permite el procesamiento de código detrás d escena, con una capacidad de multitarea a este lenguaje.
- **History:** Tiene la capacidad de incorporar los pasos de una aplicación dentro del historial de navegación.
- **Offline:** Mantiene las aplicaciones funcionales, incluso cuando un dispositivo es desconectado de la red.

De esta manera HTML define la estructura, incorporando nuevos elementos con la integración de las nuevas propiedades de CSS y CSS3 para hacer su estructura visual y sobre todo hacerla funcional por medio de JavaScript incorporando técnicas que ayuden a hacerla animada. Los componentes HTML, CSS y JavaScript son las bases que conforman HTML5 [5], una poderosa herramienta para desarrollar páginas web extremadamente funcionales como agregar audio, video, imágenes animadas etc.

HTML5 es muy extenso en definir todas sus técnicas, métodos y uso de sus herramientas, pero con la ayuda de algunas fuentes (tutoriales, libros, artículos, etc.) podrás aprender el manejo de todas sus funciones que HTML5 pone a tu disposición como programador y diseñador.

1.4 Concepto de UML (Lenguaje Unificado de Modelado)

Uno de los conocimientos básicos y aplicados en el desarrollo del software es UML [21], [22], [9] siendo un estándar de desarrollo de software, que permite diseñar la estructura del sistema, en una forma muy compleja, para entender su comportamiento del mismo. UML tiene diversos diagramas para la construcción del diseño (diseño estructural), tal como se describe a continuación.

1.4.1 Diagramas de clases.

El diagrama de clases [21] tiene como concepto que una *clase* es cualquier cosa o grupo de cosas con atributos que tienen similitud. Su representación es un rectángulo con tres secciones, vea la figura 1.15. Puesto que el diagrama de clases, es la relación entre dichas clases que involucran al sistema, tal como se puede observar en la figura 1.15.

Figura 1.15 Representación de una clase.

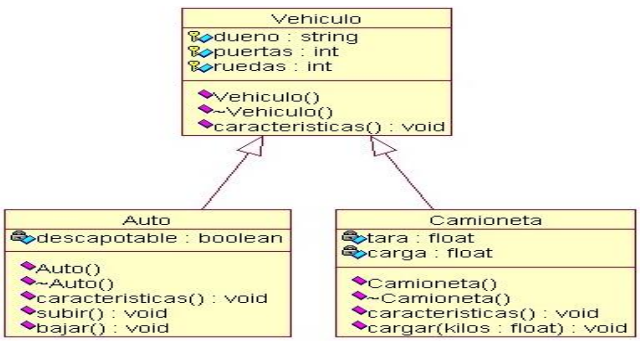

Figura 1.16 Representación del diagrama de clases.

1.4.2 Diagrama de objetos

Un diagrama de objeto [21] no es más que la instancia de una clase (representado por un rectángulo), es decir que tiene acciones y atributos específicos de una entidad; la cual se representa por medio de un texto subrayado ó con letras mayúsculas, dentro de un rectángulo, tal como lo muestra la figura 1.17.

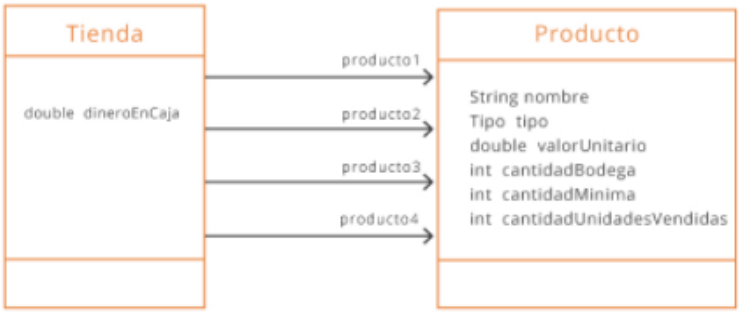

Figura 1.17 Representación del diagrama de objetos.

1.4.3 Diagramas de casos de uso.

Antes de la explicación de este diagrama; es importante tener en cuenta la simbología básica, tal como lo muestra la figura 1.18.

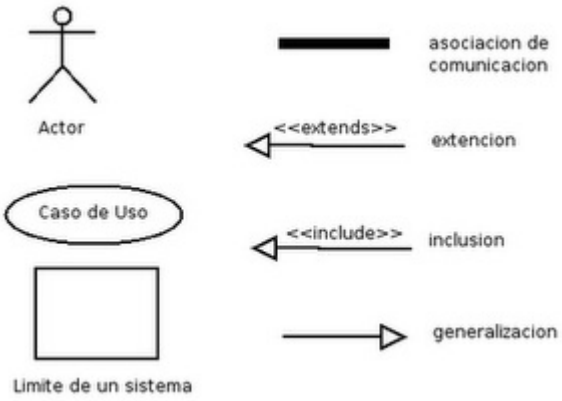

Figura 1.18 Simbología de un diagrama

Ahora bien, el diagrama de casos de uso [21][22]; se refiere a la descripción de las acciones o comportamiento que debe realizar el sistema, desde el punto de

vista del usuario; además se basa en identificar a los actores principales (usuario / humano o maquina) y la forma de interactuar con el sistema. Estas acciones van escritas dentro de una elipse, la figura 1.19 muestra la forma de interactuar entre los actores.

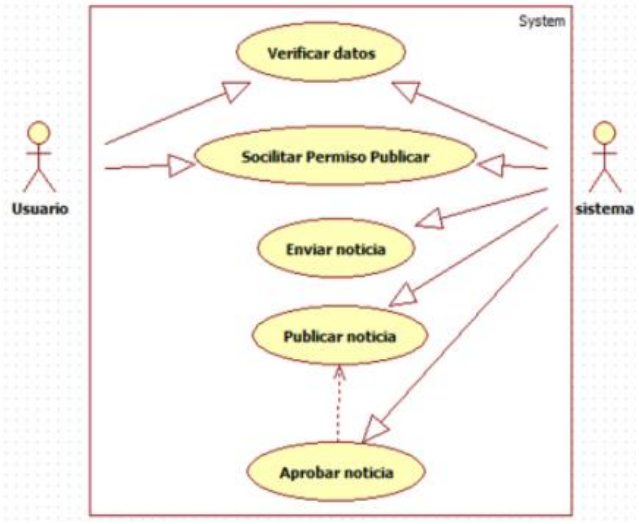

Figura 1.19 Representación del diagrama de casos de uso.

1.4.4 Diagramas de estados.

El diagrama de estados [21] se refiere al estado en que se encuentra cada acción; es decir, si esta en reposo, con dirección hacia abajo, con dirección hacia arriba, etc. Vea el ejemplo en la figura 1.20.

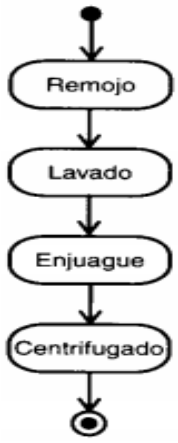

Figura 1.20 Representación del diagrama de estados.

1.4.5 Diagrama de secuencias.

El diagrama de secuencia [21] representa una línea de vida basado en las interacciones entre usuario y sistema, el cual está representado por un rectángulo que encierra el nombre de un objeto, y en la parte superior el actor, tal como se observa en la figura 1.21.

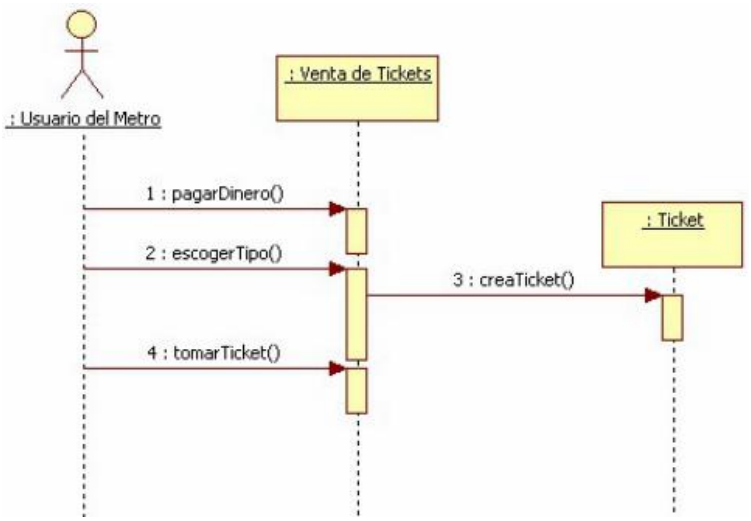

Figura 1.21 Representación del diagrama de secuencias.

1.4.6 Diagrama de actividades.

Este diagrama de actividades [21] se refiere a las acciones que se dan dentro de un caso de uso, es decir, la actividad que desempeña cada elemento dentro del sistema, tal como se puede observar en el ejemplo de la figura 1.22.

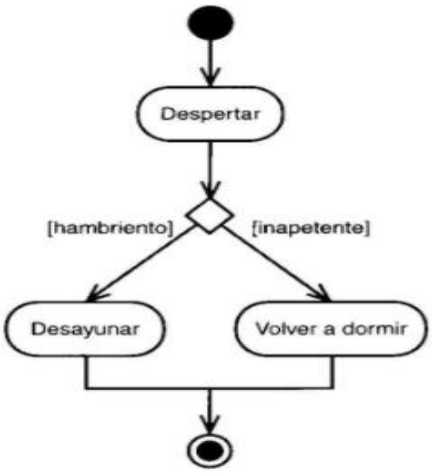

Figura 1.22 Representación del diagrama de actividades.

1.4.7 Diagrama de colaboraciones.

El diagrama de colaboraciones [21], se refiere a la forma de trabajar en conjunto entre todos los elementos que componen a un sistema, para que su funcionamiento sea el correcto, vea la figura 1.23.

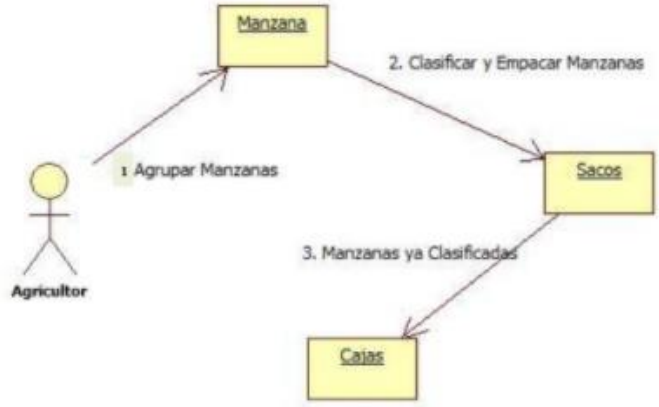

Figura 1.23 Representación del diagrama de colaboraciones.

1.4.8 Diagrama de componentes.

El diagrama de componentes [21] se refieres a un elemento en particular; es decir que acción debe cumplir cada elemento del sistema y que más tarde estarán interconectados. Vea el ejemplo de la representación 2.24.

Figura 1.24 Representación del diagrama de componentes.

1.4.9 Diagrama de distribución.

El diagrama de distribución [21] se refiere a la arquitectura de física de un sistema, en la cual se puede visualizar las interconexiones entre los equipos (computadora) que participan en el sistema. Cada equipo se representa por un

cubo, una línea que las une (interconexión), vea su representación en la figura 1.25.

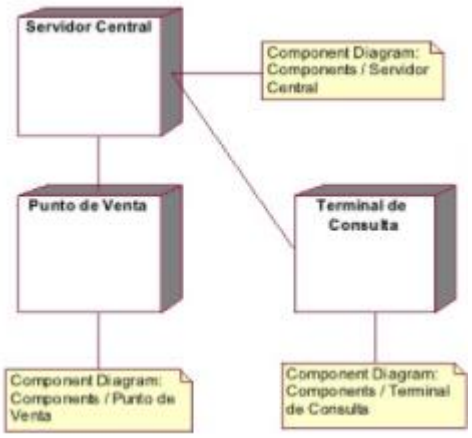

Figura 1.25 Representación del diagrama de componentes.

Es importante mencionar que dependiendo de las características del sistema, es como elegimos el tipo de diagrama (s) que engloban a UML [21], [22] y mejor convenga para su diseño (modelado del sistema).

Conclusión del capítulo

Se ha hecho un arduo estudio a cerca de la metodología de Scrum, así como el diseño basado en el UML (Lenguaje Unificado de Modelado) y de la utilización del framework Codeigniter, para desarrollar el proyecto de sistematización. En base a los estudios de este capítulo, que se ha investigado; podemos garantizar que al llevar a cabo esta metodología es la mejor opción para este proyecto; en cuanto al diseño podemos elegir alguno de los diagramas que mejor convenga para la implementación del sistema, así como la utilización de Codeigniter que brinda ventajas y beneficios, debido a que pude utilizar herramientas externas para que el sistema sea mucho más completo.

Capítulo 2 Análisis y diseño del sistema.

2.1 Planteamiento del problema de la SDRSOT (Secretaría de Desarrollo Rural Sustentabilidad y Ordenamiento Territorial).

En todo desarrollo web, se necesita de un análisis profundo del problema, tanto de la aplicación a utilizar, como del problema a resolver, en una forma detallada con la finalidad de ser precisos en el proceso de desarrollo del proyecto. Tal como continuación se presenta.

Solicitud de cómputo: para el llenado de esta solicitud, al igual que la vehicular, se requirió de un clasificador por objeto del gasto para la administración, que se aplica en la elaboración del presupuesto de egresos que constituye a cada solicitud, en este caso la de cómputo. De igual forma se requirió contar con la información de todos los equipos de cómputo que están a disposición de la secretaría, cada equipo contará con nombre de usuario, departamento, serie, modelo, marca e IME; por consiguiente los datos de todos los colaboradores.

Solicitud vehicular: Para el llenado de esta solicitud, se requirió de un clasificador por objeto del gasto para la administración, que se aplica en la elaboración del presupuesto de egresos que constituye a esta y cada una de las solicitudes, así como su respectiva denominación y partida. Se adquirió información de todos los vehículos proporcionados por la secretaría, cada vehículo cuenta con su respectiva placa, modelo y marca, de igual forma se adquirió los datos de todos los colaboradores, es decir nombre y cargo.
NOTA: Los datos de los colaboradores, serán almacenados en cada solicitud.

Solicitud general: Para el llenado de esta solicitud, se requirió de un clasificador por objeto del gasto para la administración, que se aplica en la elaboración del presupuesto de egresos que constituye a esta y cada una de las solicitudes, así como su respectiva denominación y partida.
NOTA: Los datos de los colaboradores, serán almacenados en cada solicitud.

Solicitud de compras: Para el llenado de esta solicitud, se requirió almacenar la siguiente información:

1. Nombre de la UR de la clave presupuestal; es decir, el área que lo está solicitando; el número de la UR al que corresponde el presupuesto con el que se pagará la compra. (UR es un identificador de los departamentos que hay en la secretaría. P/e UR 578 es el departamento de Dirección de calidad aire y cambio climático).
2. se llevará un consecutivo por la UR para el control interno de la misma, es decir el número de solicitud.
3. Un control de fecha para saber quien hizo una solicitud y que día; la fecha deberá estar conformada por día/mes/año; en la denominación se coloca el nombre completo de la partida presupuestal; la partida es el numero de codificación, que aparece en el clasificador por objeto de gasto; la adquisición para uso exclusivo de adquisiciones, para colocar el sello de recepción y control de esta área.

Se requiere el número de partida, llevará una numeración consecutiva, por producto solicitado; cantidad, se colocará el número de la cantidad de cada producto solicitado; unidad de medida, dependiendo del producto solicitado, se especificara la unidad de medida: pieza, caja, litros etc.

En la descripción se colocará, se pondrá el nombre del producto solicitado; en el campo de especificaciones, se pone de manera detallada lo que se requiere del producto descrito; en el importe aproximado (unitario, total, subtotal, IVA y el total final de todo), se coloca el precio aproximado antes del IVA, es muy importante que comprendan que el precio que coloquen es un estimado , sin embargo, si el precio está muy por debajo del promedio, la compra se cancelará y correrá por parte del área solicitante. Es importante desglosar el IVA, en caso de que la compra no aplique IVA se pondrá $0.00; en el motivo de la compra, se debe justificar debidamente el motivo de la compra, el cual debe coincidir con el programa presupuestario del que se trate; en la clave presupuestal viene en el oficio de autorización de recurso y es con ese número, con el que se van a pagar las compras, consta de 20 números y letras, y la parte principal está conformado por una letra y tres números que se encuentran en las ultimas 5 posiciones de dicha clave.

También se requiere del nombre del programa presupuestario, es muy importante que todo valla muy bien justificado, es decir, debe haber estricta

congruencia entre la UR y la el programa presupuestario, así como el fin que va a tener el material, que se está adquiriendo. El nombre del programa también viene en su oficio de autorización; en el nombre de quien recibe y lugar de entrega, se debe poner el nombre de la persona que va a recibir el material y el lugar de entrega deberá tener nombre del lugar y dirección; el teléfono y horario se colocan con la finalidad de que el proveedor haga la cita para la entrega y efectivamente se encuentre la persona que va a recibir el material, en el concepto de horario también se puede colocar la leyenda “favor de sacar cita para entrega”.

Se requieren de dos datos que son de gran importancia en la realización de la solicitud, como es: solicitante (nombre, firma y cargo), en esta parte firma el responsable de la UR, es decir, como se explicó, todo debe de ser congruente con la clave presupuestal.

Se tiene el caso de la UR 581, es la Coordinación General de Recursos Naturales, estructuralmente dirigen la UR 582 y 583, las subdirecciones de parques, pertenecen a la UR 581 orgánicamente, sin embargo, presupuestalmente, el programa presupuestario que pertenece a parques E173, pertenece a la UR 583 Dirección de Conservación y Vida Silvestre, de esta la única persona “presupuestalmente autorizada” para firmar estas solicitudes, sería el director de dicha UR y no el coordinador, motivo por el cual se realiza el formato de firmas autorizadas, con la finalidad de no romper con la unidad de mando y la delegación de autoridad de la coordinación y de las subsecretarias, así como el control del presupuesto que ustedes requieran, en ese formato podrán especificar la persona autorizada para solicitar el material, pertinente a sus aéreas, ya que esto, lo encontramos con varias URs.

Y por último en el campo de recibe (nombre, firma y cargo), solamente la jefa del Departamento de Recursos Materiales, estarán autorizados para firmar esta parte.

2.2 Metodología ágil de Scrum para la SDRSOT.

Aplicar la metodología Scrum [2], es el principio en el desarrollo de este proyecto y sobre todo aplicar las reglas que propone Scrum en su metodología

para hacer un producto entregable; pero sobre todo alcanzar los objetivos en cuanto a tiempo. Empezamos definiendo lo siguiente.

2.2.1 Historias backlog de la SDRSOT.

Obtenemos todas las historias de usuario [13] (especificaciones y requerimientos), con forme a las necesidades que el usuario ha solicitado.

1. PÁGINA PRINCIPAL: El sistema deberá llevar dos campos, para que la persona que desee realizar una solicitud acceda a la pagina con un id y contraseña.

2. MENÚ DE OPCIONES: Es la pagina principal con un menú de opciones que nos llevara a realizar la solicitud deseada.

3. SOLICITUD DE COMPRAS. Opción uno de la lista del menú; es la página donde se podrá realizar una solicitud de compras.

4. SOLICITUD DE CÓMPUTO. Opción dos de la lista del menú; es la página donde se podrá realizar una solicitud de cómputo.

5. SOLICITUD VEHICULAR. Opción tres de la lista del menú; es la página donde se podrá realizar una solicitud vehicular.

6. SOLICITUD GENERAL. Opción cuatro de la lista del menú; es la página donde se podrá realizar una solicitud general.

7. IMPRIMIR SOLICITUD. Opción de imprimir formato; Una vez que el usuario llene el formato de la solicitud correspondiente con los datos adecuados, pueda imprimir dicha solicitud.

2.2.2 Pila de producto (historias de usuario) de la SDRSOT.

Creamos una tabla con las especificaciones de cada historia (tabla 2.1), con la idea de tener en claro lo que se requiere que haga el sistema, y lo que se podría necesitar para poder realizarla [7].

PILA DE PRODUCTO BACKLOG				
N°	**NOMBRE**	**ESTIMACION**	**COMO PROBLARLO**	**NOTAS**
1	PAGINA PRINCIPAL: ID Y CONTRASEÑA	16	Abrir página, el usuario ingresará su id y contraseña, para tener acceso al sistema.	Se requiere crear una base de datos e interfaz de página.
2	MENU DE OPCIONES	3	Seleccionar la solicitud que desea realizar.	Crear interfaz de página.
3	SOLICITUD DE COMPRAS	10	Llenar cada uno de los campos de texto que muestra la solicitud de compra.	Crear interfaz, y consultas a la base de datos.
4	SOLICITUD DE COMPUTO	10	Llenar cada uno de los campos de texto que muestra la solicitud de cómputo.	Crear interfaz, y consultas a la base de datos.
5	SOLICITUD VEHICULAR	10	Llenar cada uno de los campos de texto que muestra la solicitud vehicular.	Crear interfaz, y consultas a la base de datos.
6	SOLICITUD GENERAL	10	Llenar cada uno de los campos de texto que muestra la solicitud de general.	Crear interfaz, y consultas a la base de datos.
7	GENERAR SOLICITUD DE IMPRESION	8	Si la solicitud se ha llenado correctamente, el usuario dará clic en el botón enviar y automáticamente se genera el documento listo para su impresión.	Crear interfaz, y consultas a la base de datos.

Tabla 2.1 Historias usuario.

Consideramos todas las historias de usuario, como un bloque o elemento que conforma la pila de producto tal como se ve en la figura 2.1, y de esta manera sea más fácil trabajar en cada una de las historias.

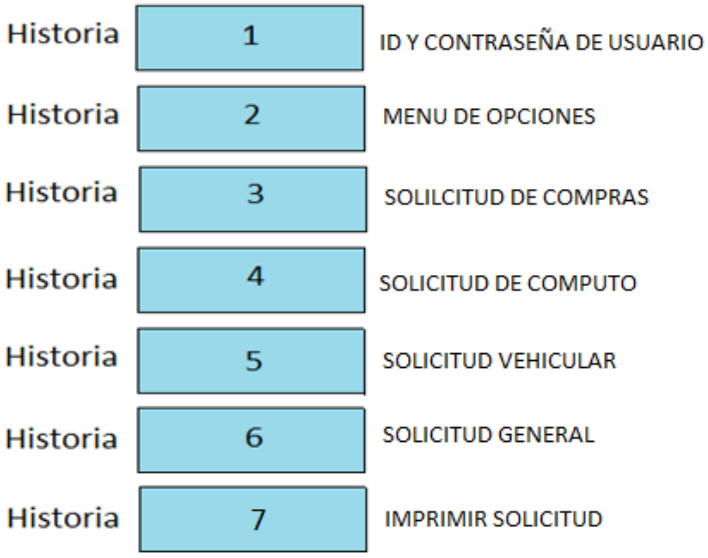

Figura 2.1. Pila de producto Backlog.

2.2.3 Pila de Sprint de la SDRSOT.

Seleccionamos los elementos 1,3 y 7 de la pila de producto como prioridad, considerando que son las más importantes y que de esta manera podremos finalizar el proyecto en tiempo y forma. Ahora nuestra nueva pila, se llamará, pila de sprint (pila priorizada de elementos) [7], tal como lo muestra la figura 2.2.

Estos elementos se irán agregando a la nueva pila (pila de sprint) con forme se van concluyendo los anteriores en cada iteración (recalcando que durante el desarrollo del proyecto, podemos mover los elementos de la pila de sprint y formar una nueva pila si es necesario, dependiendo de la importancia que tenga el elemento).

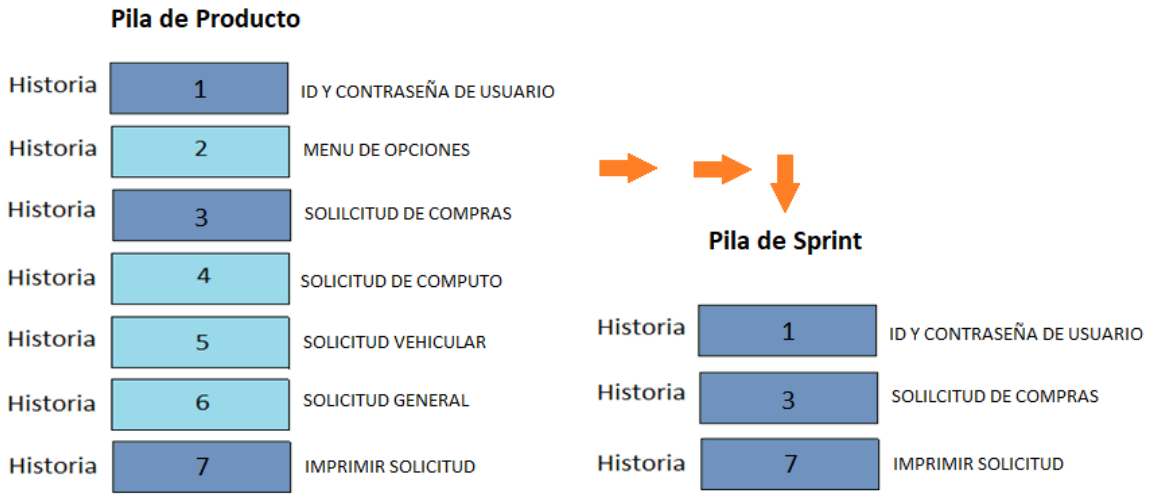

Figura 2.2 Selección de elementos en la pila de Sprint 1.

Ahora podemos observar en la figura 2.3, que se obtienen dos pilas de Sprint; la prima pila de Sprint con los elementos (1, 3 y 7) y la segunda con los elementos (2, 4, 5 y 6), en las cuales se tendrá que trabajar en cada una, y así finalizar cada iteración (proceso) en el tiempo que se ha estimado en cada una de las pilas de Spint.

Figura 2.3 Pilas de Sprint 1 y 2.

2.2.4 Técnica de estimación - Ojo de buen cubero.

Hasta ahora se puede percibir que la pila no es muy grande, puesto que con las técnicas de estimación que maneja Scrum, el más adecuado para su desarrollo, es el ojo del buen cubero [7]; es decir solo basta con una reunión para identificar las historias con las que empezaremos a trabajar (1, 3 y 7).

2.2.5 Planificación de sprint de la SDRSOT.

Lo siguiente es dividir las funcionalidades de la pila de producto en tareas, y sub tareas; lo más detallado posible, así como estimar el tiempo que se le dará a cada una de las pilas para identificar cada punto, y de esta forma trabajar en su desarrollo con la finalidad de alcanzar los objetivos en cuanto al tiempo estimado de cada pila de sprint [7].

Veamos la planificación de la primera pila de sprint con los elementos (1, 3 y 7) tal como se ve en la tabla siguiente (tabla 2.2).

Planificación de Sprint 2 (SDRSOT)				
Nº	**HISTORIAS**	**TAREAS**	**SUB TAREAS**	**T.E.**
1	**PAGINA PRINCIPAL ID Y CONTRASEÑA.**	▪ CREAR BASE DE DATOS	BOOTSTRAP	2 HR
		▪ CONEXIÓN A LA BASE DE DATOS		1 HR
		▪ DISEÑAR INTERFAZ	XAMPP	3 HRS
		▪ PROBAR INTERFAZ		1 HR
		▪ CREAR FORMULARIO	JAVA SCRIPT	2 HR
		▪ CONFIGURAR RECAPTCHA		1 HR
		▪ CONFIGURAR SESIONES	CODIGO PHP	4 HRS
		▪ PAGINA RESPONSIVE.		2 HRS
PROBAR ELEMENTO DE LA PILA				
3	**SOLICITUD DE COMPRAS**	▪ CREAR INTERFAZ	BOOTSTRAP	1 HR
		CREAR FORMULARIO		2 HS
		▪ VALIDAR FORMULARIO.	CODIGO PHP / CONSULTAS SQL	4 H
		▪ IMPLEMENTAR JAVASCRIPT		2 HR
		▪ IMPLENTAR PAGINA	ESTILOS CSS	1 HR
7	**GENERAR SOLICITUD DE IMPRESION**	▪ CREAR INTERFAZ	BOOTSTRAP	1 HR
		▪ CODIGO PHP	CONSULTAS SQL	4 HRS
		▪ IMPLEMENTAR JAVASCRIPT		2 HRS
		▪ IMPLEMENTAR PAGINA RESPONSIVE	ESTILOS CSS	1 HR

Tabla 2.2 Planificación de la pila de sprint 1.

Veamos la siguiente planificación de la segunda pila de Sprint con los elementos (2, 4, 5 y 6), tal como se ve en la tabla 2.3.

Planificación de Sprint 2 (SDRSOT)				
2	MENU DE OPCIONES	▪ DISEÑAR INTERFAZ		2 HRS
		▪ PROBAR LA INTERFAZ		1 HR
PROBAR ELEMENTO DE LA PILA				
4	SOLICITUD COMPUTO	▪ CREAR INTERFAZ	BOOTSTRAP	1 HR
		▪ CREAR FORMULARIO		2 HR
		▪ VALIDAR FORMULARIO	CODIGO PHP / CONSULTAS SQL	4 HRS
		▪ IMPLEMENTAR JAVASCRIPT		2 HRS
		▪ IMPLEMENTAR PAGINA RESPONSIVE	ESTILOS CSS	1 HR
PROBAR ELEMENTO DE LA PILA				
5	SOLICITUD VEHICULAR	▪ CREAR INTERFAZ	BOOTSTRAP	1 HR
		▪ CRERAR FORMULARIO		2 HRS
		▪ VALIDAR FORMULARIO	CODIGO PHP / CONSULTAS SQL	4 HRS
		▪ IMPLEMENTAR JAVASCRIPT		2 HRS
		▪ IMPLEMENTAR PAGINA RESPONSIVE	ESTILOS CSS	1 HR
PROBAR ELEMENTO DE LA PILA				
6	SOLICITUD GENERAL	▪ CREAR INTERFAZ	BOOTSTRAP	1 HR
		▪ CREAR FORMULARIO		2 HRS
		▪ VALIDAR FORMULARIO	CODIGO PHP / CONSULTAS SQL	4 HRS
		▪ IMPLEMENTAR JAVASCRIPT		2 HRS
		▪ IMPLEMETAR PAGINA RESPONSIVE	ESTILOS CSS	1 H
PROBAR PILA				

Tabla 2.3 Planificación de la pila de Sprint 2.

2.2.6 Forma práctica de planificación de la pila de Sprint con Trello para la SDRSOT.

Para llevar a cabo la planificación en una forma práctica utilizando el tablón de tareas [7], donde se verán los avances de cada elemento de la pila. Para llevar a cabo esta dinámica ocupamos la herramienta Trello que es un tablón virtual que permite registrar actividades con tarjetas virtuales, organizarlas, agregar listas, agregar comentarios, imágenes, enlaces etc.

Se muestra la lista de las etiquetas (tareas) que van en cada elemento (historia), identificados por un color en específico (vea la representación en la figura 2.4), estas serán agregadas en cada elemento dependiendo de lo que requiera cada una.

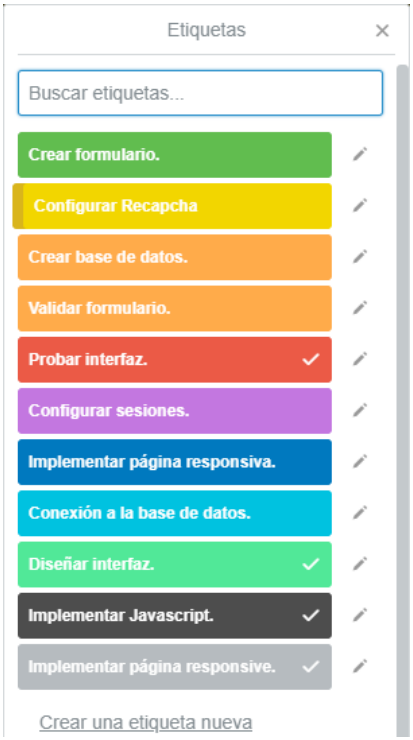

Figura 2.4 Lista de tareas

En la figura 2.5 podemos observar las tres divisiones del tablón como son: lista de tareas (historias), en proceso y hecho (terminado), como se puede ver aun no hay ninguna tarea en proceso.

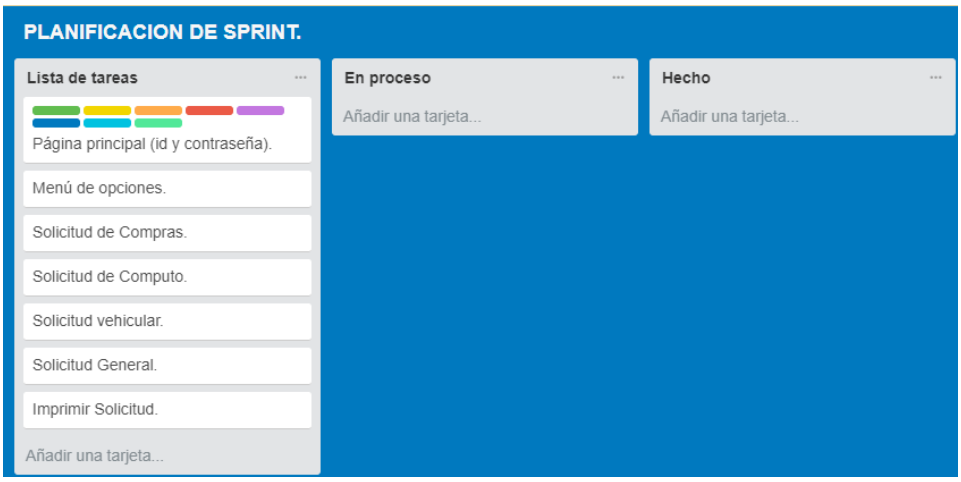

Figura 2.5 Lista de elementos de la pila de sprint.

En la figura 2.6, se puede apreciar el primer elemento de la pila de Sprint en proceso junto a sus respectivas tareas, es decir que se está trabajando en el primer elemento (*página principal id y contraseña).*

Figura 2.6 Elementos en proceso.

Ahora vea los avances en el tablón de tareas (figura 2.7), se tiene dos tareas en proceso y una totalmente terminada.

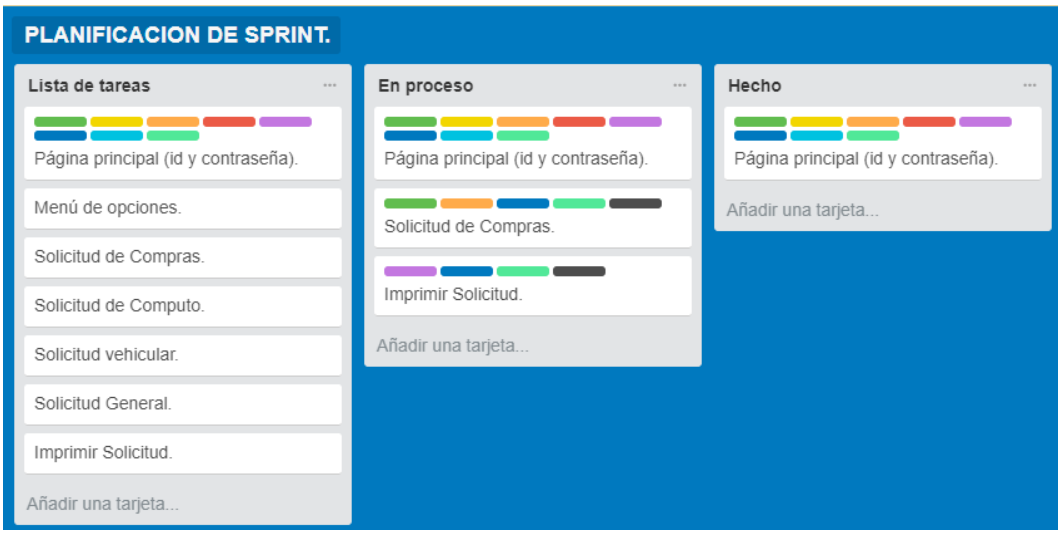

Figura 2.7 Elementos terminados.

En la última división de la tabla podemos observar que ha quedado perfectamente finalizado cada elemento, y el tablón ha quedado totalmente completo en sus tres fases, vea la figura 2.8.

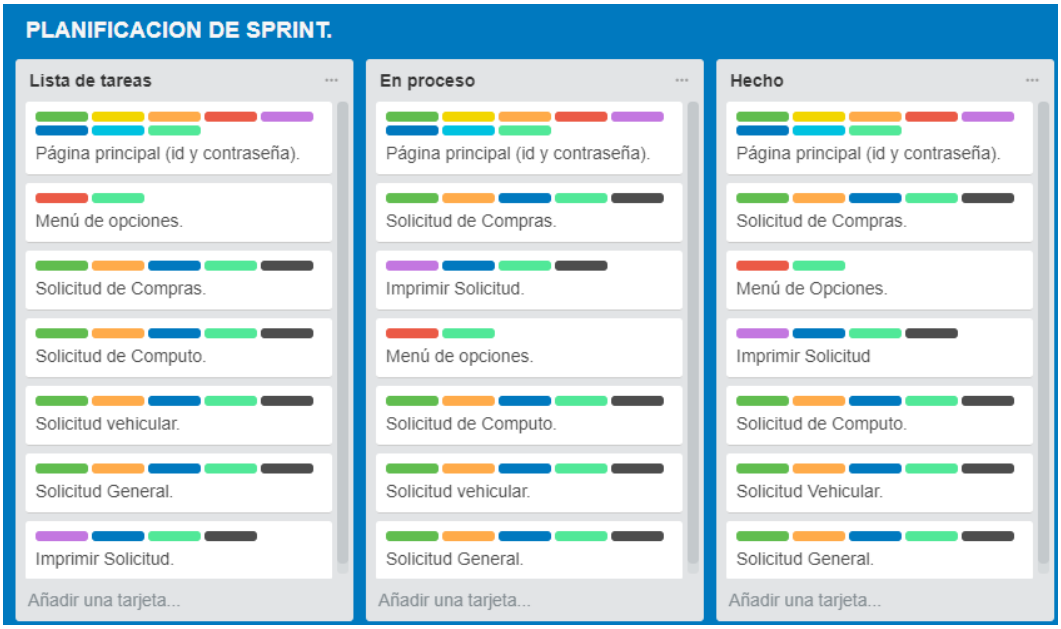

Figura 2.8 Pila de Sprint terminados.

Es importante mencionar que por cada elemento terminado, se tiene que hacer una demo de Sprint, una pequeña prueba de funcionalidad, para comprobar que el funcionamiento sea el correcto y así continuar con el siguiente elemento hasta finalizar todo el proceso.

2.3 Diseño del software aplicando UML (Lenguaje Unificado de Modelado).

Empezamos con la construcción del diseño del sistema (modelado del sistema), que propone UML, con base al análisis y planteamiento del problema, de tal forma que sea entendible para los usuarios y la interacción que tienen los usuarios con el sistema.

2.3.1 Utilización de Diagrama de casos de uso mediante UML.

Utilizamos el diagrama de casos de uso, donde representamos los requerimientos del sistema (solicitudes), en un conjunto de casos de uso; así como los actores participantes y la interacción entre ellos; en este caso es un actor (Colaborador) que interactúa directamente con el sistema. Todo esto para

tener una idea más clara de lo que esperamos que haga el sistema, de lo establecido en anteriormente. Vea la figura 2.9.

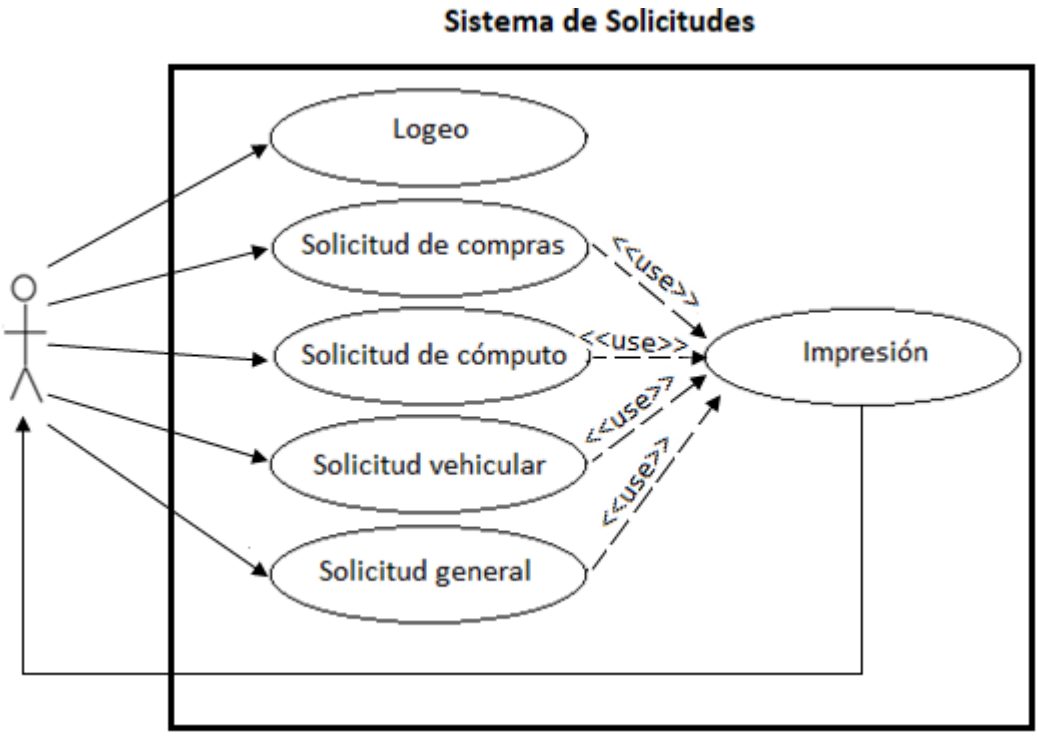

Figura 2.9 Diagrama de casos de uso para la SDRSOT.

2.3.2 Base de datos del software (solicitudes) utilizando el diagrama de clases (modelo entidad relación).

Uno de los métodos aplicados en el desarrollo de software que requiera de una base de datos [15], [16], es el modelo entidad relación (E/R) definido en el diagrama de clases, representado mediante un esquema lógico que define una base de datos, esto quiere decir que se hace un reconocimiento de entidades y sus relaciones entres ellas. Por lo tanto es utilizado en este proyecto de software (solicitudes) como a continuación se presenta (figura 2.10).

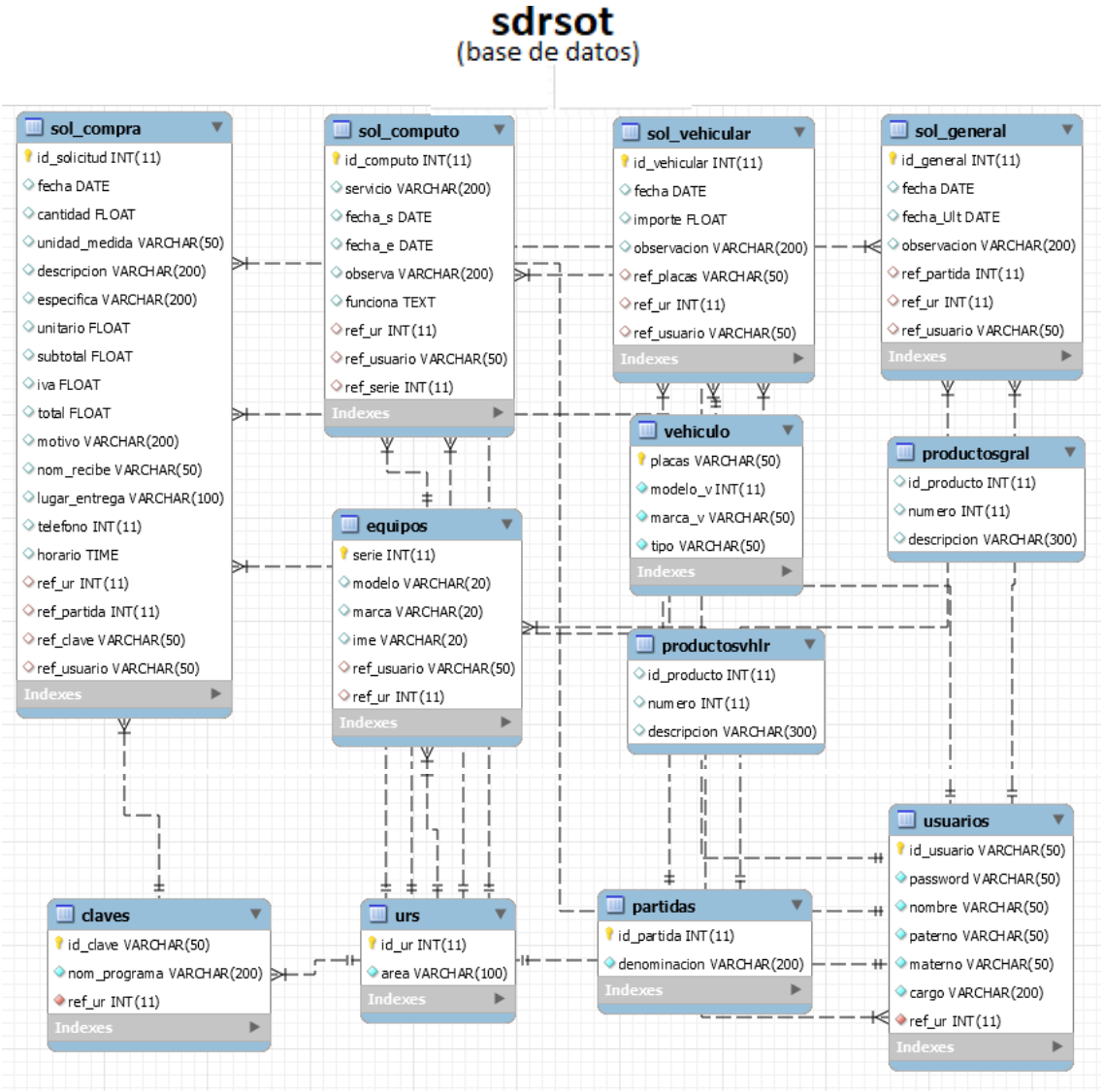

Figura 2.10 Esquema entidad relación (base de datos).

2.4 Software (solicitudes) aplicando Modelo Vista Controlador.

Veamos las representaciones de cada uno de los elementos que conforman el MVC contenida en Codeigniter [3], y la forma de interactuar entre ellos, para finalmente tener como resultado una página web. La capeta de proyecto (Codeigniter) se llama ***solicitudess.***

2.4.1 Modelo (models) del software.

Para utilizar esta parte del modelo en el controlador [3], se requiere que la base de datos exista, ya que desde esta carpeta se gestionan todas las consultas a la base de datos; puesto que el modelo, no es más que una carpeta en donde irán todos los archivos de gestión. El primer paso es hacer la conexión a la

base de datos, desde la carpeta *solicitudess/config/database.php,* en donde se coloca el nombre de la base de datos en este caso ***sdrsot****, c*omo se puede observar en la figura 2.11.

database.php

```php
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

$active_group = 'default';
$query_builder = TRUE;

$db['default'] = array(
    'dsn'   => '',
    'hostname' => 'localhost',
    'username' => 'root',
    'password' => '',
    'database' => 'sdrsot',
    'dbdriver' => 'mysqli',
    'dbprefix' => '',
    'pconnect' => FALSE,
    'db_debug' => (ENVIRONMENT !== 'production'),
    'cache_on' => FALSE,
    'cachedir' => '',
    'char_set' => 'utf8',
    'dbcollat' => 'utf8_general_ci',
    'swap_pre' => '',
    'encrypt' => FALSE,
    'compress' => FALSE,
    'stricton' => FALSE,
    'failover' => array(),
    'save_queries' => TRUE
);
```

Figura 2.11 Conexión a la base de datos.

Una vez configurada la conexión a la base de datos, se procede a hacer todas las consultas que sean requeridas para el funcionamiento del software, desde la carpeta *application/models/bases_models.php*, veamos el siguiente código (figura 2.12) que genera la primera consulta a la base de datos, en donde condiciona que los datos que ingrese el usuario como son su *id* y *contraseña* deben coincidir con el registro de la base de datos.

bases_model.php

```php
<?php
defined('BASEPATH') OR exit('No direct script access allowed');
class bases_model extends CI_Model
{

        function __construct()
        {parent::__construct();}

        public function validaLogin($data)
        {
            $cadena="select * from usuarios where id_usuario='".$data['login']."'
            and password='".$data['password']."'";
            $query = $this->db->query($cadena);

            if ($query->num_rows() > 0)
            {
                return $query;
            }
            else
            {
                return FALSE;
            }
    }   }
```

Figura 2.12 Consulta a la base de datos.

2.4.2 Vista (views) del software.

En la ruta *solicitudess/application/views,* van todas las vistas que se requieren del software (solicitudess); sin olvidar que una vista es la parte visual hacia el usuario [3]. En esta parte se hace una interacción entre las herramientas CSS, HTML, Javascript y PHP. Es importante mencionar que una vista tiene una estructura que lo conforman que son: cabecera (header), cuerpo (body) y pie de página (footer).

El primer archivo llamado ***header.php*** como se ve en la figura 2.13. El cual representa la cabecera y apertura del documento HTML gestiona las herramientas externas a codeigniter que son necesarias para hacer una página dinámica, como lo es bootstrap que trae consigo a (Jquery). En este mismo archivo podemos ver una parte del código que genera un estilo (CSS), en donde es posible afectar a todas las vistas que se han creado. En caso de que solo se quiera afectar a un solo archivo se tendrá que especificar por medio de un **id** ó **class**.

head.php

```
<!DOCTYPE html>
<html>
<head>
  <title>solicitudes</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/css/
  bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.6/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/js/bootstrap.min.js"></script>
  <script type="text/javascript"></script>
  <style type="text/css">
    h1{
    padding: 3px;
    text-align: center;
    color: #34495E;
    }
    p.center {
    text-align: center;
    color: black;
    }
    th {
    background-color: #94D4BC;
    color: navy;
    text-align: center;
    }
  </style>
</head>
```

Figura 2.13 Cabecera del documento HTML (head).

El siguiente código es el cuerpo del documento (body) veamos el código de la figura 2.14, que genera campos de de texto de un formulario; esta vista se llama ***login_usuario.***

login_usuario.php

```
<body>
  <div class="container-fluid bg-5 text-center">
    <form action="<?php echo base_url();?>index.php/Welcome/validaLogin" method="POST">
          <h1>    Ingrese sus Datos</h1>
          <label for="login">Login:</label>
          <?php echo form_error('login');?>
          <input type="text" class="form-control" id="login" placeholder="Ingresa Login" name="login">
          <label for="pwd">Password:</label>
          <?php echo form_error('pwd');?>
          <input type="password" class="form-control" id="pwd" placeholder="Ingresa password" name="pwd">
          <div class="g-recaptcha" data-sitekey="6Lcx7TAUAAAAAM1H0WPSPbNzYdh4hDABZA9b-dTH"></div>
          <br>
        <button type="submit" id="registra" class="btn btn-primary">Enviar Datos</button>
    </form>
  </div>
</body>
```

Figura 2.14 Cuerpo del documento HTML (body).

El siguiente código es el pie de página y cierre del documento HTML, tal como se ve en la figura 2.15.

```php
<footer class="container-fluid bg-4 text-center" style="background-color: #FAA41F">
    <div style="background-color: #DFC25D">
        <p>Todos los derechos reservados <a href="https://www.w3schools.com"
        data-toggle="tooltip" title="Visit w3schools">&copy</a>Francisca López Velázquez</p>
    </div>
</footer>
</html>
```

Figura 2.15 Pie de página (footer).

2.4.3 Controlador (controller) del software.

Esta es la parte importante del MVC ya que este se encarga de controlar las diferentes operaciones que se solicitan; es decir, si es una consulta a la base de datos, una inserción a una tabla de la base de datos, o lo que se quiera mostrar en una vista, permite entradas y salidas dependiendo de lo que se quiera realizar y si estos son correctos; El controlador [3] está definido por funciones que tienen como tarea llamar a todos los archivos creados ya sean modelos o vistas y es ubicado en: solicitudess/application/controllers/Welcome.

El siguiente código hace el llamado la vista que hemos creado, la vista estructurada HTML (head, body y footer), como se pude observar en la figura 2.16, el cual es leído por el controlador con el nombre ***Welcome.***

```php
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Welcome extends CI_Controller {

    function __construct()
    {
     parent::__construct();
     $this->load->model('bases_model');
    }
    public function index()
    {
        $this->load->view('head');
        $this->load->view('login_usuario');
        $this->load->view('footer');
    }
}
```

Figura 2.16 Controlador.

2.4.4 Prevención a ataques XSS del software (solicitudes).

Es importante mencionar que cuando se habla de formularios y sobre todo de validar a un usuario, tenemos que tener cierta seguridad en la entrada de datos [2]. Lo que utilizamos para este software es lo siguiente:

1. Configurar en solicitudess/application/config/config.php que sea verdadero, $config['global_xss_filtering'] = True.
2. Declarar la librería en solicitudess/application/controllers/Welcome.php colocando $this->load->library ('encryption') en el constructor (construct) del controlador.
3. Asignar una llave a la aplicación de codeigniter desde la ruta solicitudess/application/config/config.php, la llave tendrá que ser lo más larga posible, entre mas caracteres tenga mas difícil será el riesgo, $config['encryption_key'] = 'LLAVE'.
4. Reiniciar los servidores XAMPP.

En el siguiente esquema en la figura 2.17 podemos observar la utilización de la librería ***encrypt,*** y aplicado en los campos de texto del formulario, esto quiere decir que el software estará protegido ante inyecciones Javascript, los cuales pueden tener contenido pornográfico, textos inadecuados los cuales estarían insertando en la base de datos, o peor a un, borrar datos de la base de datos [2].

Hay muchos otros medios para proteger nuestras páginas; como xss_clean(), strip_tags, entre otros, sin embargo lo que siempre es recomendable es utilizar la encriptación por **HAS**, para que nuestra sitio web este lo más protegido posible y evitar posibles errores [2].

Welcome.php

```php
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Welcome extends CI_Controller {

    function __construct()
    {
     parent::__construct();
     $this->load->model('bases_model');
     $this->load->library('encrypt');
    }
    public function index()
    {
        $this->load->view('head');
        $this->load->view('login_usuario');
        $this->load->view('footer');
    }
    public function validaLogin()
    {
          $encrypted_string = $this->encrypt->encode($this->input->post('login'));
          var_dump($encrypted_string);
          var_dump($this->encrypt->decode($encrypted_string));
          $this->form_validation->set_rules('login','login','trim|required');
          $this->form_validation->set_rules('pwd','pwd','trim|required');
          $this->form_validation->set_rules('g-recaptcha-response','recaptcha validation','
              trim|required|callback_validate_captcha');
          $this->form_validation->set_message('required','<div class="alert alert-danger">
            El campo %s es  obligatorio</div>');

            if ($this->form_validation->run() != FALSE)
            {
                  $login=$this->input->post('login',TRUE);
                  $password=$this->input->post('pwd',TRUE);
                  $data = array(
                      'login'=> $login,'password'=>$password);

                  $sql=$this->bases_model->validaLogin($data);
                    if ($sql==FALSE)
                    {
                    {
                    session_destroy();
                    header("Location: index.php");
                    }
                    }
                    $_SESSION['tiempo'] = time();
                        $datasession="";
                        $line ="";
                        $line2="";
                        $line3="";
                          foreach($sql->result() as $fila)
                          {
                          foreach($sql->result() as $fila)
                          {
                                $line2 = $fila->cargo ;
                          }
                          foreach($sql->result() as $fila)
                          {
                                $line3 = $fila->id_usuario ;
                          }
                              $_SESSION["usuario"]= $line;
                              $_SESSION["cargo"]= $line2;
                              $_SESSION["loginn"]= $line3;
                              redirect('/Welcome/menu/', 'location');
                    }
            }
            else
            {
                  redirect('/Welcome/index/', 'location');
            }
    }
}
```

Figura 2.17 Librería para prevenir ataques XSS.

Conclusión del capítulo.

Hemos encaminado a resolver el problema con una de las mejores metodologías como lo es Scrum, el cual ha ayudado a identificar las historias de usuario, es decir, tener claro los requerimientos del sistema en base a las necesidades del usuario bajo un análisis profundo e implementarlo en cada proceso de desarrollo.

En cuanto al sistema, tal como se puede observar en este capítulo, se llevo a cabo la realización del software desde el diseño utilizando UML (Lengua Unificado de Modelado), hasta la codificación, con la utilización de Codeigniter aplicando MVC, hemos comprobado que es una de las mejores herramientas para el desarrollo de páginas web, en este caso de la sistematización de documento para la SDROT (Secretaría de Desarrollo Rural, Sustentabilidad y Ordenamiento Territorial).

Capítulo 3 Prototipo del software.

En todo desarrollo de software se requiere de un proceso para su realización y lo más importante para su ejecución; llevada a cabo por medio de un conjunto de instrucciones y una serie de pruebas conocida como prototipo, que permite una representación limitada de un producto y de esta manera explorar su uso. Veamos la siguiente representación en un conjunto de procesos que se llevaron a cabo mediante las pruebas en el funcionamiento de dicho sistema.

3.1 Pasos para ingresar al sistema (solicitudes).

Tenemos cuatro solicitudes disponibles con diferentes funcionalidades, por lo que se necesita un guía para su indagación y de esta manera facilite al usuario el uso correspondiente a cada formato de solicitud.

3.1.1 Entrar al sistema. Para tener acceso al sistema el usuario deberá abrir un explorador y tendrá que ingresar la siguiente URL: http://localhost:8080/solicitudess/ tal como se ve en la figura 3.1.

Figura 3.1 URL para entrar al sistema.

3.1.2 Página de inicio (login y password). Primera pantalla del sistema como se ve en la figura 3.2. El usuario deberá ingresar su NIP y contraseña para poder acceder al sistema y de esta manera poder realizar cualquier acción; en caso de que estos datos no sean válidos, el sistema no lo dejara pasar a la siguiente página, ya que este sistema cuenta con niveles seguridad en cuanto a validación de usuario; una de estas como se puede observar en la imagen es recapcha de Google, que verifica si el usuario no es un robot.

Figura 3.2 Página de validación de usuario.

3.1.3 Página del menú de opciones. Es el menú de opciones; una vez que el usuario se haya validado y que sus datos sean los correctos, pasará a la siguiente página, el cual nos muestra un menú de opciones que indica las solicitudes disponibles a realizar (compras, computo, vehicular y general), como se ve en la figura 3.3.

Figura 3.3 Página de menú de opciones.

3.1.4 Llenar formato de solicitud de compras. El usuario dará click a la primera opción del menú (Figura 3.3), enseguida el usuario pasara a la página de ***solicitud de compras*** (figura 3.5)

1. Al llenar esta solicitud, el usuario encontrará un apartado de campos de operaciones al ingresar cierta *cantidad* y multiplicado por *unitario* obtendrá el resultado automáticamente; para realizar este proceso el usuario dará click en el botón *subtotal.* Como lo muestra la figura 3.4.

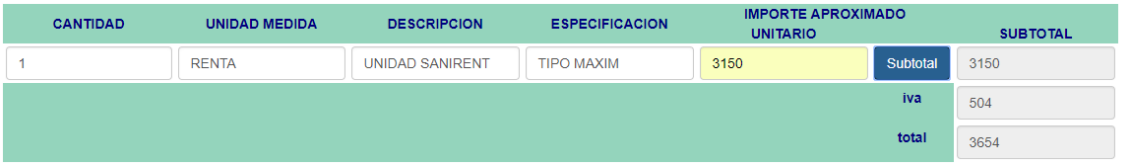

Figura 3.4 Campos de operaciones automáticos.

2. Una vez que el usuario haya llenado la solicitud correspondiente tal como lo muestra la figura 3.5, lo que procede es dar click en el botón enviar para ser guardados en la base de datos (si faltó llenar algún campo de texto, la página no dejara enviar los datos, hasta que todos los campos estén completamente llenos) o bien puede dar click en el botón limpiar para volver a llenar los datos correctamente.

Figura 3.5 Formato de solicitud de Compras.

3. Esta página también cuenta con un panel de navegación que estará disponible en todas las solicitudes (compras, computo, vehicular y general), este panel tiene la opción de *CERRAR SESION* como lo muestra la figura 3.6 es decir; si el usuario no desea realizar ninguna otra acción después de haber elaborado dicha solicitud dentro del sistema tendrá que dar click en la opción *CERRAR SESION*, y enseguida estará fuera del sistema y lo llevará a la página de inicio (página de logeo).

Figura 3.6 Panel de navegación.

3.1.5 Generar formato de impresión de la solicitud de compras. Una vez que los datos de la solicitud de compras han sido enviados a la base datos, automáticamente nos mostrará el registro de la persona que hiso dicha solicitud, en una nueva página que estará lista para ser impresa, el usuario deberá dar click en el icono de la impresora para ser enviada, tal como se ve en la figura 3.7.

Figura 3.7 Formato para mandar a imprimir.

3.1.6 Enviar solicitud a impresora (solicitud de compras). Una vez que el usuario ha dado click en el botón de imprimir, enseguida lo enviara a la impresora que esté configurada a la red para finalizar el proceso de impresión de la solicitud de compras. Vea la figura 3.8.

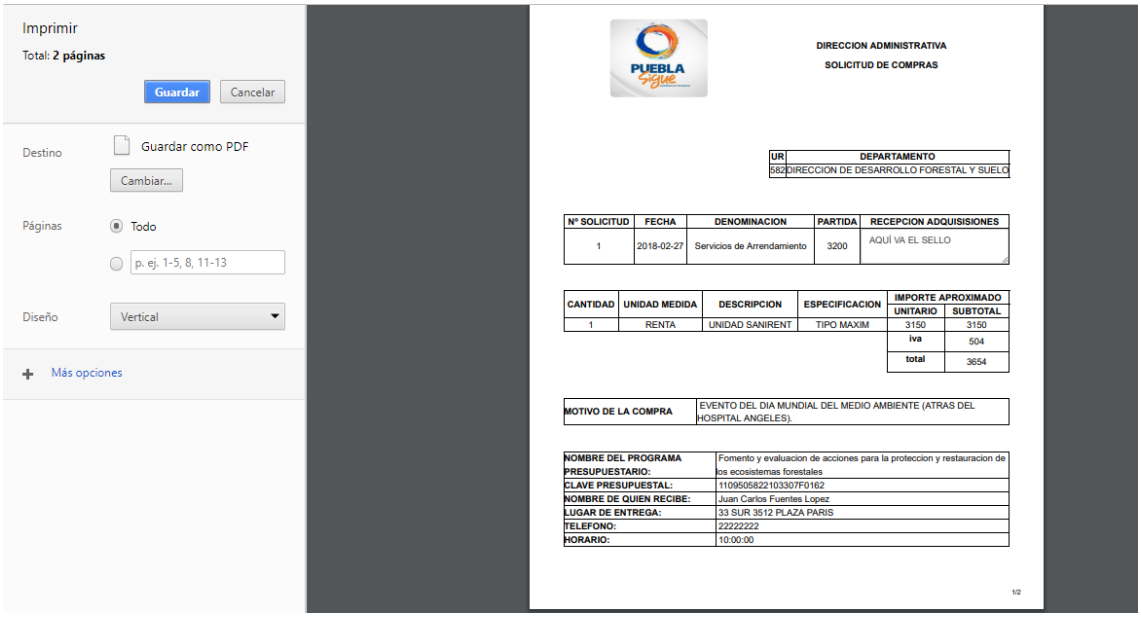

FIGURA 3.8 Impresión de hoja de solicitud de compras

3.1.7 Llenar formato de solicitud de cómputo. El usuario dará click a la segunda opción del menú (Figura 3.3) enseguida el usuario pasara a la página de ***solicitud de cómputo,*** vea la figura 3.9.

1. Una vez que el usuario ha llenado la solicitud correspondiente, lo que procede es dar click en el botón enviar para ser guardados en la base de datos (si faltó llenar algún campo de texto, la página no dejará enviar los datos, hasta que todos los campos sean llenados) o bien puede dar click en el botón limpiar para volver a llenar los datos correctamente.

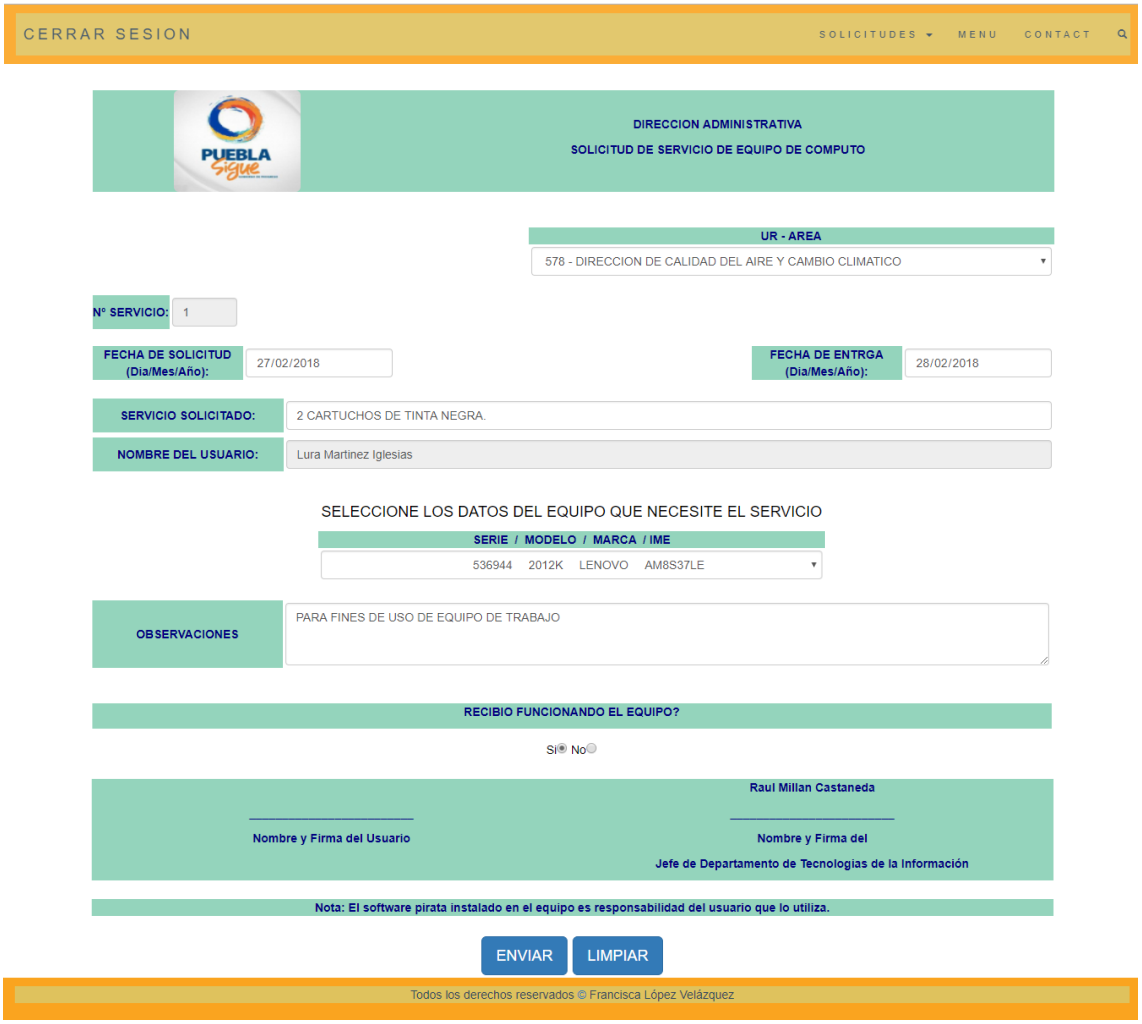

FIGURA 3.9 Formato de solicitud de cómputo.

3.1.8 Generar formato de impresión de la solicitud de cómputo. Una vez que los datos de la solicitud de cómputo, han sido enviados a la base datos, automáticamente nos mostrará el registro de la persona que hizo dicha solicitud, en una nueva página que estará lista para ser impreso, el usuario deberá dar click en el icono de la impresora para ser enviada, tal como se ve en la figura 3.10

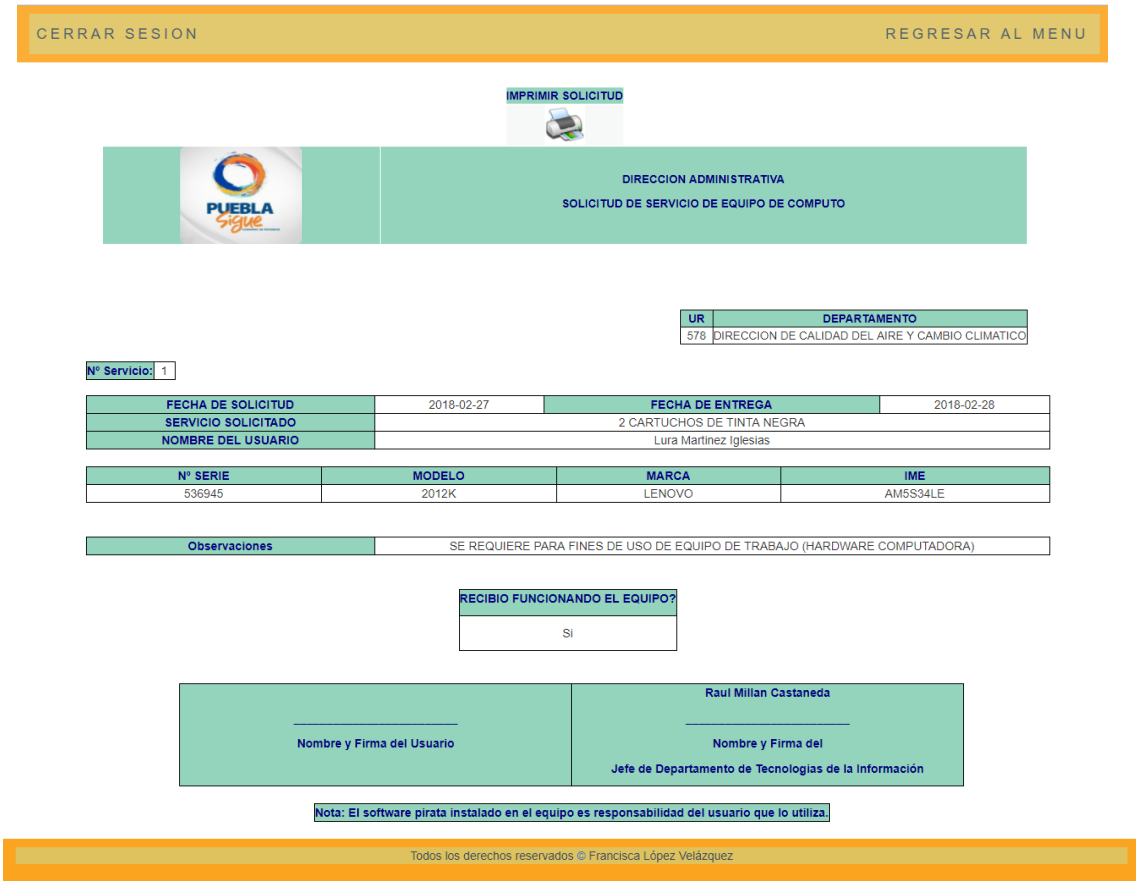

Figura 3.10 Formato para mandar a imprimir.

3.1.9 Enviar solicitud a impresora (solicitud de cómputo) Una vez que el usuario haya dado click en el botón de imprimir, enseguida lo enviará a la impresora que esté configurada a la red para finalizar el proceso de impresión de la solicitud de cómputo. Vea la figura 3.11.

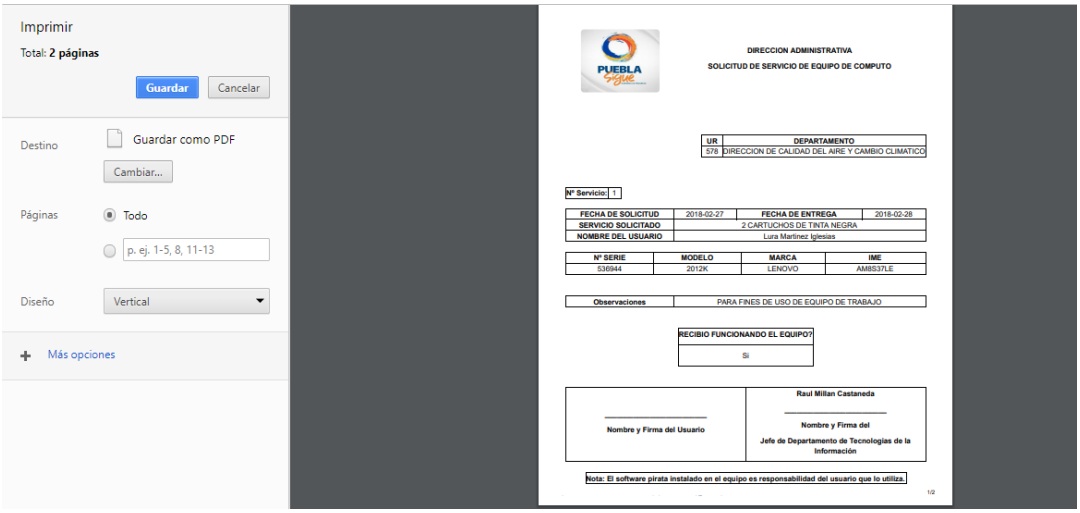

Figura 3.11 Impresión de hoja de solicitud de cómputo.

3.1.10 Llenar formato de solicitud vehicular. El usuario dará click a la tercera opción del menú (Figura 3.3), enseguida el usuario pasara a la página de ***solicitud vehicular,*** vea la figura 3.13.

1. Al llenar esta solicitud, el usuario encontrará un apartado de agregación de campos, con tan solo seleccionar y dar click en *agregar.* Como lo muestra la figura 3.12. También puede hacer click en el botón eliminar (*X*) si ya no requiere de un capo agregado.

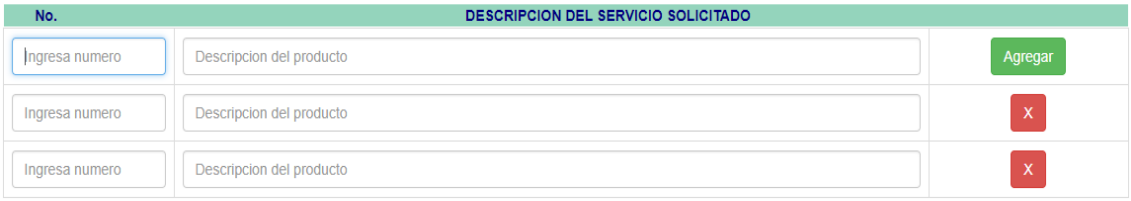

Figura 3.12 Campos de agregación.

2. Una vez que el usuario llene la solicitud correspondiente tal como se muestra en la figura 3.13. Lo que procede, es dar click en el botón enviar para ser guardados en la base de datos (si faltó llenar algún campo de texto, la página no dejará enviar los datos, hasta que todos los campos sean llenados) o bien puede dar click en el botón limpiar para volver a llenar los datos correctamente.

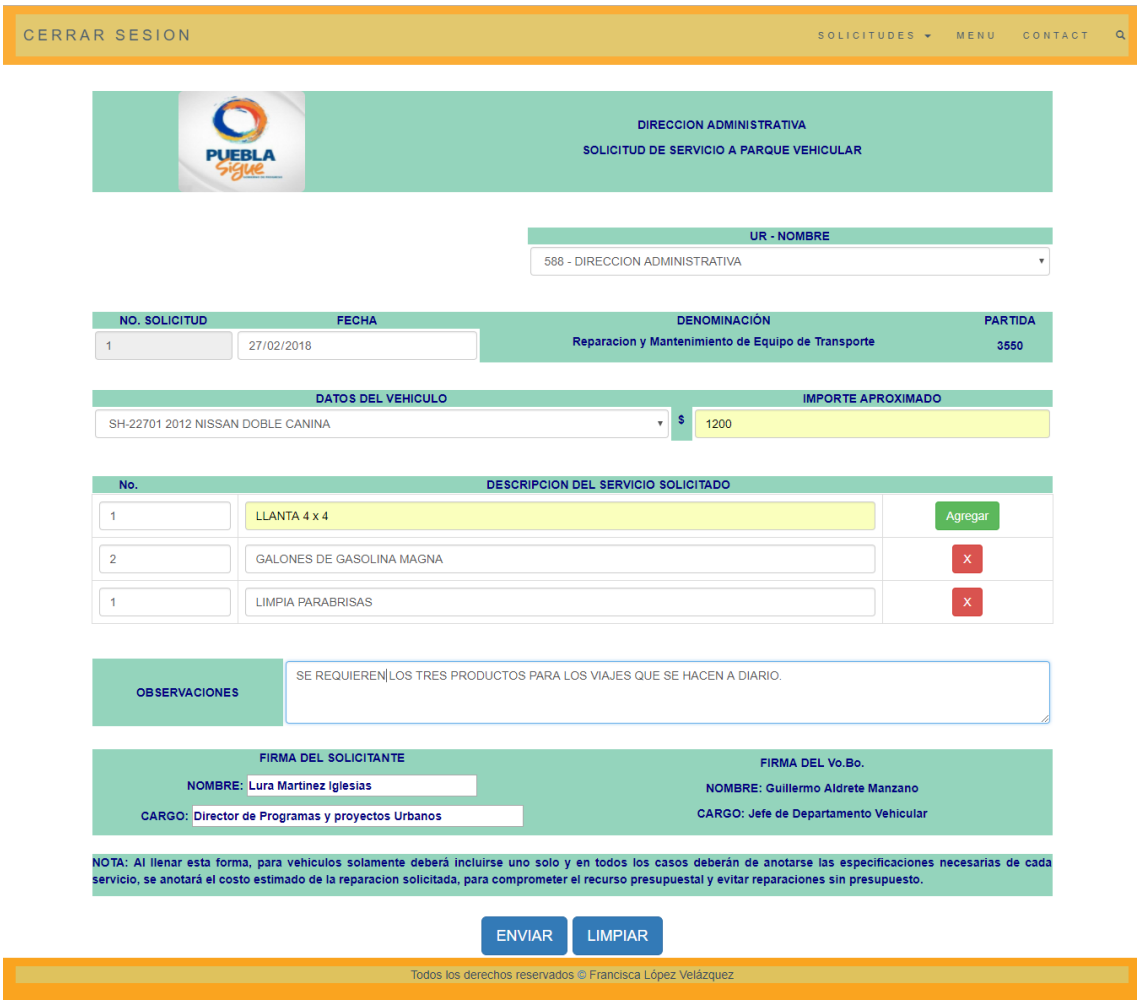

Figura 3.13 Formato de solicitud vehicular.

3.1.11 Generar formato de impresión de la solicitud vehicular. Una vez que los datos de la solicitud vehicular han sido enviados a la base datos, automáticamente nos mostrará el registro de la persona que hizo dicha solicitud, en una nueva página que estará lista para ser impreso, el usuario deberá dar click en el icono de la impresora para ser enviada, tal como se ve en la figura 3.14.

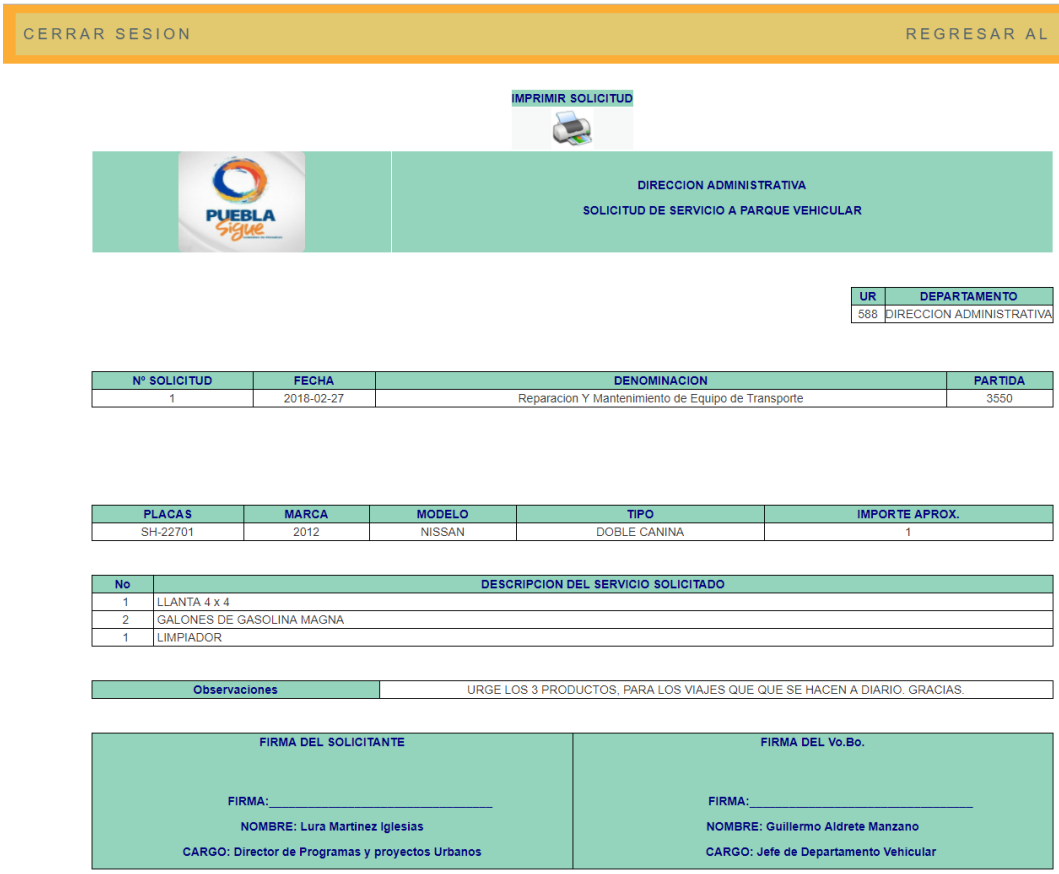

Figura 3.14 Formato para mandar a imprimir.

3.1.12 Enviar solicitud a impresora (solicitud vehicular). Una vez que el usuario haya dado click en el botón de imprimir, enseguida lo enviara a la impresora que esté configurada a la red para finalizar el proceso de impresión de la solicitud vehicular. Como se ve en la figura 3.15.

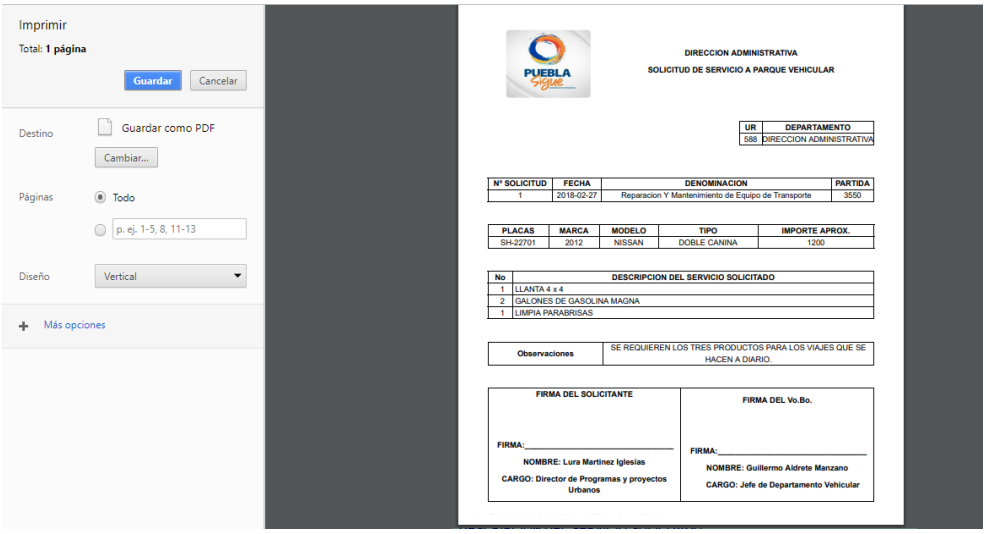

Figura 3.15 Impresión de hoja de solicitud vehicular

3.1.13 Llenar formato de solicitud general. El usuario dará click a la cuarta opción del menú (Figura 3.3), enseguida el usuario pasará a la página de ***solicitud general,*** vea la figura 3.17.

1. Al llenar esta solicitud, el usuario encontrará un apartado de agregación de campos, con tan solo seleccionar y dar click en *agregar.* Como lo muestra la figura 3.16. También puede hacer clic en el botón eliminar (*X*) si ya no requiere de un capo agregado.

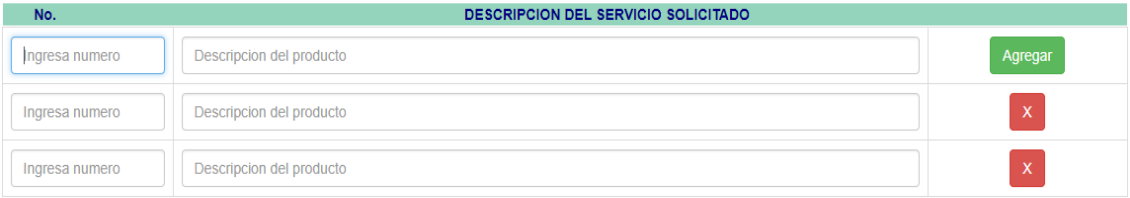

Figura 3.16 Campos de agregación.

2. Una vez que el usuario llene la solicitud correspondiente tal como lo muestra la figura 1.17. Lo que procede, es dar click en el botón enviar para ser guardados en la base de datos (si faltó llenar algún campo de texto, la página no dejará enviar los datos, hasta que todos los campos sean llenados) o bien puede dar click en el botón limpiar para volver a llenar los datos correctamente.

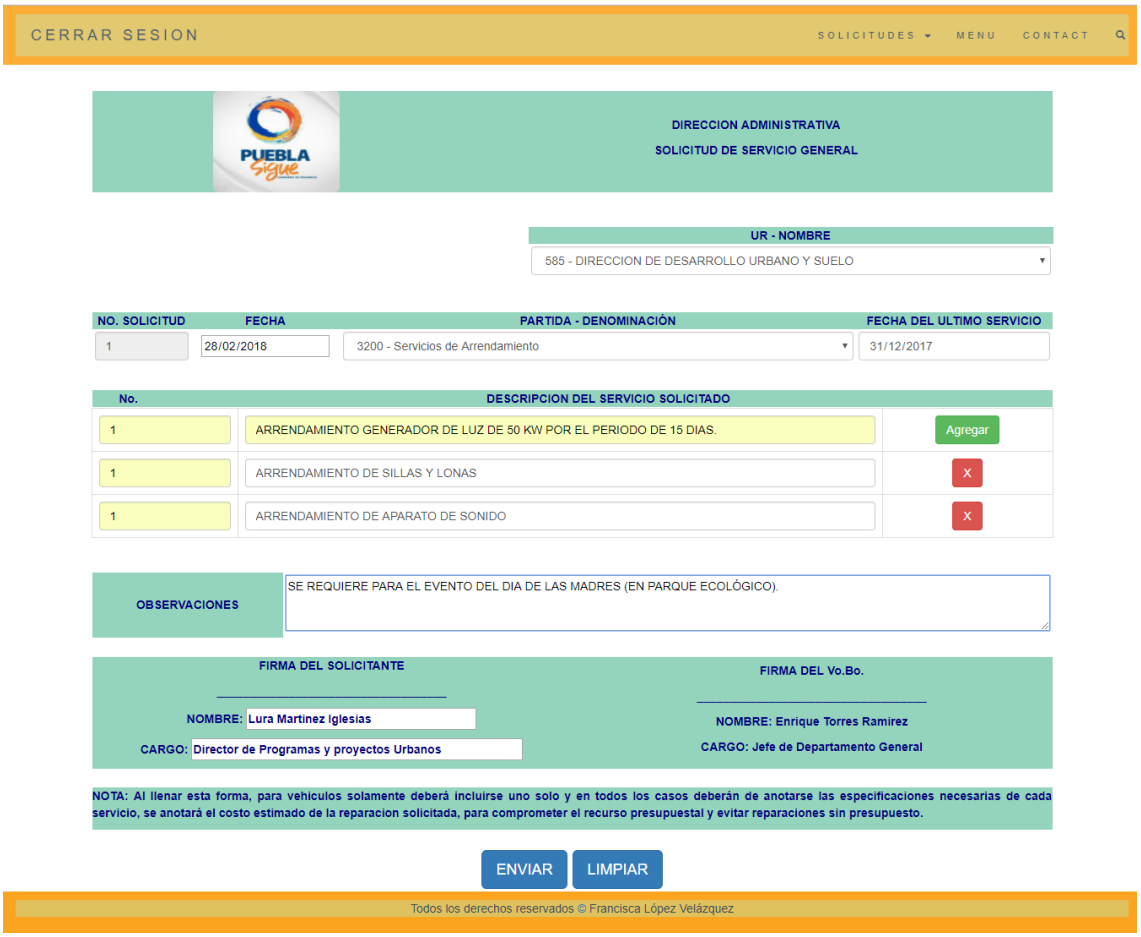

Figura 3.17 Formato de solicitud general.

3.1.14 Generar formato de impresión de la solicitud general. Una vez que los datos de la solicitud general han sido enviados a la base datos, automáticamente nos mostrará el registro de la persona que hizo dicha solicitud, en una nueva página que estará lista para ser impreso, el usuario deberá dar click en el icono de la impresora para ser enviada, tal como se ve en la figura 3.18.

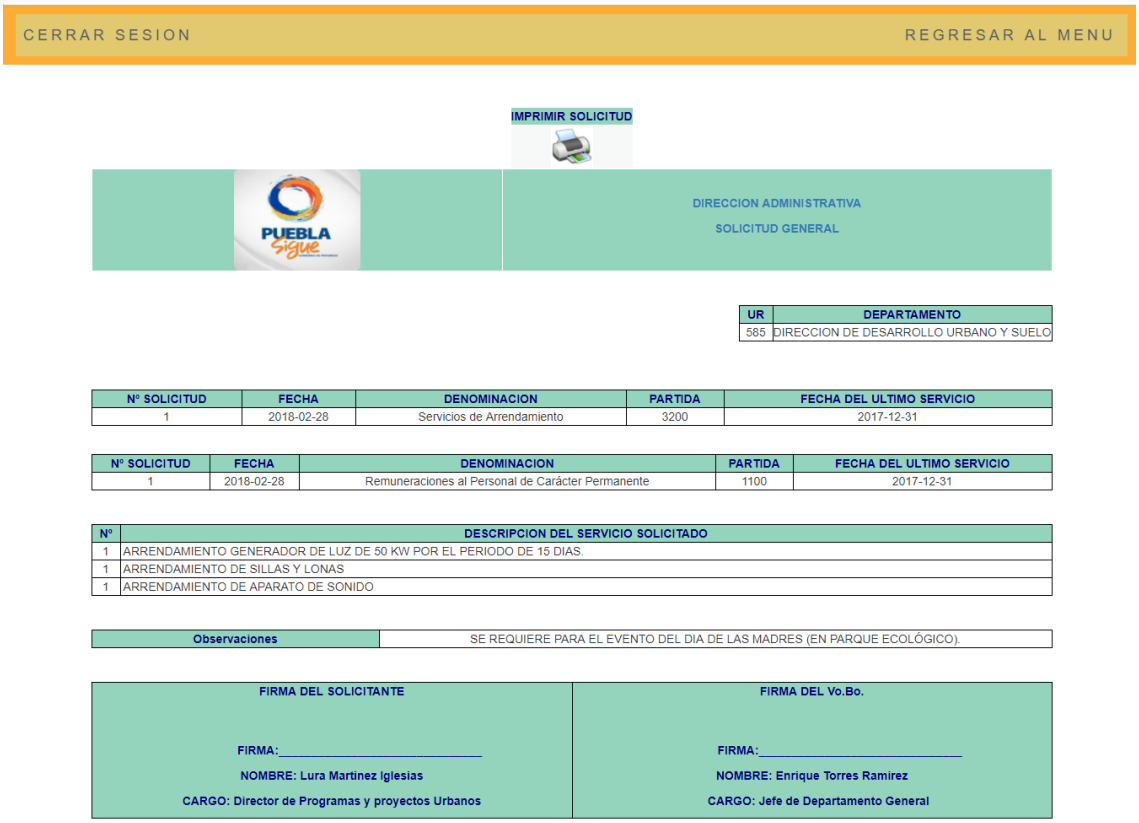

Figura 3.18 Formato para mandar a imprimir.

3.1.16 **Enviar solicitud a impresora (solicitud general).** Una vez que el usuario haya dado click en el botón de imprimir, enseguida lo enviará a la impresora que esté configurada a la red para finalizar el proceso de impresión de la solicitud de general. Vea la figura 3.19.

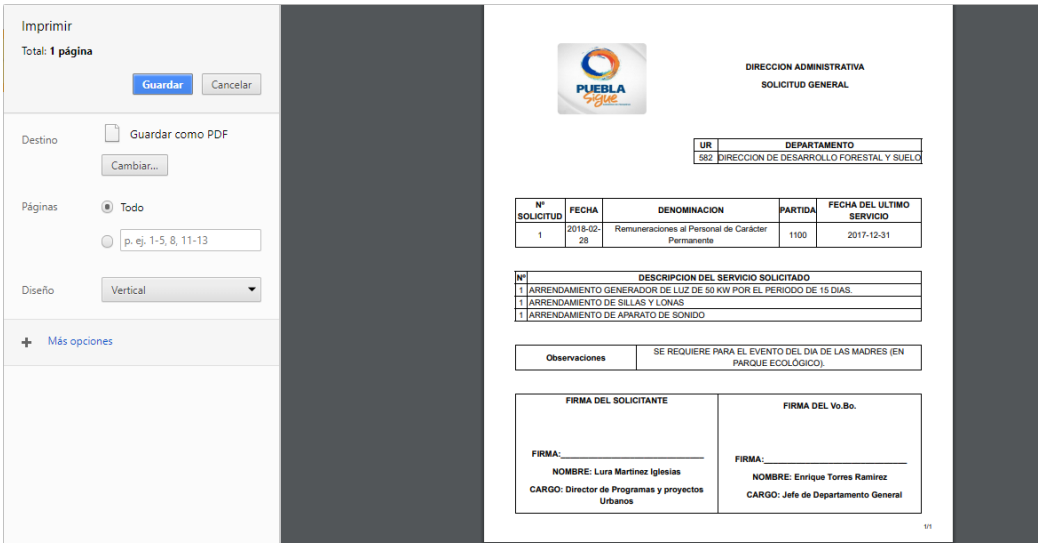

Figura 3.19 Impresión de hoja de solicitud vehicular

Conclusión del capítulo

La funcionalidad del sistema es exitosa. En este capítulo le presentamos el prototipo del software, en donde se pude apreciar la funcionalidad del sistema (manual de usuario), es decir los pasos que tiene que seguir para acceder al sistema y de esta manera el usuario pueda llenar el formato de la solicitud correspondiente con éxito.

Conclusión

Se resolvió el problema con una de las mejores metodologías como lo es Scrum, que aplica un método ágil, con objetivos específicos en cuanto a tiempo. Aplicar la metodología de Scrum ha sido la mejor opción para el desarrollo de este proyecto, alcanzando los tiempos estimados en cada iteración.

Referente al diseño fue modelado por el diagrama de casos de uso y el diagrama de clases (Modelo entidad relación) dentro de UML, los cuales fueron muy óptimos para su desarrollo.

Además con la utilización del framework que es Codeigniter debido a que es un framework flexible, el cual se pudo adaptar la arquitectura MVC (Modelo Vista Controlador), se ha logrado realizar un sistema (codificación) que cumple con los requisitos para hacer la solicitud correspondiente, facilitando la tareas diarias de trabajo de los colaboradores, ya que es un sistema que es fácil de acceder y fácil de utilizar.

Los resultados obtenidos con esta automatización de sistema, obtenemos lo siguiente.

- El proceso para realizar una solicitud, es mucho más fácil.
- Se ha hecho un respaldo de información mediante una base de datos.
- El sistema es flexible, se podrá modificar por algún otro desarrollador.
- El sistema es dinámico, porque contiene datos muy precisos, un ejemplo es hacer operaciones matemáticas, entre otros beneficios.
- La utilización del sistema puede ser desde su área de trabajo, con tan solo acceder al sistema (deberá estar conectado a una red).

Bibliografía

[1] Manaure A. (abril 23, 2010). "*CASO DE EXITO: Unilever reduce costos de almacenamiento y ambiente de datos mediante el uso de soluciones SAP",* mayo 1, 2017, The HAP Group. Sitio web: http://www.cioal.com/2010/04/23/caso-de-exitounilever-reduce-costos-de-almacenamiento-y-ambiente-de-datos-mediante-el-uso-de-soluciones-sap/

[2] Cibelli, C. (Diciembre 2012). *PHP Programación web avanzada para profesionales*. Paraguay 1307, PB, oficina 11: Alfaomega Grupo Editor Argentino.

[3] Griffiths, A. (Abril 2010). *CodeIgniter 1.7 Professional Development.* Birmingham, B27, 6PA, UK: Packt Publishing Ltd.

[4] Álvarez, M. Á. "*Manual de codeigniter, pdf*" URL http://www.izt.uam.mx/spring/wp-content/uploads/2012/07/manual-codeigniter.pdf

[5] Gauchat, J. D. (2012). *Documentos HTML5. En El gran libro de HTML5, CSS y Javascript (pp.15-18)*. Gran Via de les Corts Catalanes, 594 08007 Barcelona : Publidisa, Printed in Spain. Sitio web: https://gutl.jovenclub.cu/wp-content/uploads/2013/10/El+gran+libro+de+HTML5+CSS3+y+Javascrip.pdf

[6] Mills, C. (2013). *Practical CSS3: Develop and Design.* Berkeley, CA 94710, United States of America: Peachpit Press.

[7] H. kniberg. (2007). *Scrum y Xp desde las Trincheras*. Estados Unidos de América: Editor de infoQ.com. Sitio web: http://www.proyectalis.com/wp-content/uploads/2008/02/scrum-y-xp-desde-las-trincheras.pdf

[8] Meloni, J. C. (May 2012). *Installation QuickStart Guide with XAMPP. En PHP, MySQL and Apache*, in All One(pp.5-14). United States of America: Copyright © 2012 by Pearson Education, Inc.

[9] Quintero, J. B. & Anaya, R. (July/Dec. 2007). *MDA Y EL PAPEL DE LOS MODELOS EN EL PROCESO DE DESARROLLO DE SOFTWARE*, mayo 20, 2017, de Revista EIA Sitio web: http://www.scielo.org.co/scielo.php?pid=S1794-12372007000200011&script=sci_arttext&tlng=pt

[10] Nosedal, D. T. (Noviembre 18, 2015). *Implementación SAP EWM en Liverpool*. mayo, 2017, de ©2015 SAP SE or an SAP affiliate company. All rights reserved Sitio web: https://www.sap.com/latinamerica/documents/2015/11/d40a91c0-4d7c-0010-82c7-eda71af511fa.html

[11] Bazdresch, C. & Romo. D. (Diciembre, 2005). *EL IMPACTO DE LA CIENCIA Y LA TECNOLOGÍA EN EL DESARROLLO DE MÉXICO*, junio, 2017, de Centro de Investigación y Docencia Económicas Sitio web: https://www.ses.unam.mx/curso2008/pdf/Bazdresh.pdf

[12] Insulza, J. M., Ramdin A., Stevenson, B. J.R., Abreu, A., Hahn, S., Connolly, R., Conte, M. C., Harasic, O., Herrera, H., Redington, G., Vilariño, D.. (Noviembre 2005). *Ciencia, Tecnología, Ingeniería e Innovación para el Desarrollo, Una visión para las Américas en el siglo XXI*. Street, N.W., Washington: Copyright © 2005 por OEA.

[13] PMOinformatica.com. (Julio 10, 2013). *11 Reglas para administrar el Product Backlog en Scrum*. Junio, 2017, de Oficina de Proyectos de Informática Sitio web: http://www.pmoinformatica.com/2013/07/11-reglas-product-backlog-scrum.html

[12] Herrera, E. (Marzo 8, 2012). *Arrancar con HTML5 Curso de programación*. México: Alfaomega. [14] Quijado, J. L.. (Julio 2011). Domine JavaScript. Col. Del Valle. 03100, México D.F: Alfaomega Grupo Editor, S.A de C.V.

[15] López, I. (2013). *Base de datos*. Col. Del Valle. 03100, México.: Alfaomega Grupo Editor, S.A de C.V.

[16] Schneider, R.D. (june 2005). *Database Design and Tuning*. United States of America: Copyright 2005 by Pearson Education.

[17] Vaswani V.. (2010). *FUNDAMENTOS DE PHP*. México D.F, Delegacion Álvaro Obregon: Litográfica Ingramex.Torre A. International Standard Book Number: 978-970-10-7132-8

[18] Meloni J.C. (November 2011). *HTML5, CSS and JavaScript*. U.S. Corporate and Government sales : Copyright 2012 by Pearson Education.

[19] Charte, F. (2004). *Proyectos Profesionales PHP 5*. Madrid: EDICIONES ANAYAMULTIMEDIA (GRUPO ANAYA, S.A.), 2004. ISBN: 84-415-1770-3

[20] Taber M., Jordan D., Beaster A. Johnston S. (July 2004). *MySQL administrator's Guide*. United States of America: AB, U.S. Corporate and Government Sales, International Standard Book Number: 0-672-32634-5

[21] Schmuller, J. (November 9, 2001). *Aprendiendo UML en 24 Horas*. Naucalpan de Juárez, Edo. de México, Col. Fracc. Alce Blanco: Editorial División. Computación Sitio web: file:///C:/Users/Fran/Downloads/Prentice_Hall_Aprendiendo_UML_en_24_horas.pdf

[22] Sommerville, I. (2011). INGENIERÍA *DE SOFTWARE, Primera edición*. México: PEARSON EDUCACIÓN. Sitio web: ftp://april.frm.utn.edu.ar/METODOLOGIA%20DE%20SISTEMAS%20-%20TSP/LIBROS/Ingenieria%20de%20Software-Somerville.pdf