UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE INGENIERIA DE LA COMPUTACIÓNINGENIERÍA DE LA COMPUTACIÓN

DESARROLLO DE LA VERSIÓN 2 DE LA APLICACIÓN WEB CPI

Por:
Valentina Hernández

INFORME DE PASANTÍA
Presentado ante la ilustre Universidad Simón Bolívar
como requisito parcial para optar al título de Ingeniero de la Computación
Ingeniero en Computación

Sartenejas, Septiembre de 2018

V. HERNÁNDEZ 2018	DESARROLLO DE LA VERSIÓN 2 DE LA APLICACIÓN WEB CPI	USB INGENIERÍA DE LA COMPUTACIÓN

UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE INGENIERÍA DE LA COMPUTACIÓN

DESARROLLO DE LA VERSIÓN 2 DE LA APLICACIÓN WEB CPI

Por:
Valentina Hernández

Realizado con la asesoría de:

Tutor Académico: Prof. Soraya Carrasquel
Tutor Industrial: Ing. José Cerqueiro

INFORME DE PASANTÍA
Presentado ante la ilustre Universidad Simón Bolívar
como requisito parcial para optar al título de
Ingeniero en Computación

Sartenejas, Septiembre de 2018

Página reservada para el acta de evaluación

RESUMEN

Es una exposición clara del tema tratado en el trabajo, de los objetivos, de la metodología utilizada, de los resultados relevantes obtenidos y de las conclusiones. Mismo tipo de fuente seleccionado con tamaño 12 e interlineado sencillo en el párrafo. El resumen no debe exceder de trescientas (300) palabras escritas.
Palabras cláves: palabras, cláves, separadas por coma, cinco máximo.

ÍNDICE GENERAL

INTRODUCCIÓN

Introducción aquí.

CAPÍTULO I

ENTORNO EMPRESARIAL

En este capítulo se presenta una descripción del entorno en el cual se desarrolló el proyecto de pasantía en la empresa iKêls Consulting. Comprende una breve reseña histórica, su misión y visión, la estructura organizacional y el área a la cual el pasante estuvo asignado.

1.1. Antecedentes de la empresa

IKêls Consulting se creó el año 2008 como una empresa dedicada al desarrollo de soluciones en el área de Sistemas de Información. Los fundadores contaban con una amplia trayectoria en los procesos y tecnología para la elaboración de documentación técnica avanzada (por ejemplo normas ISO para construcción de plantas petroquímicas).

Para aprovechar la experiencia previa los productos y servicios se concentran en el área de aplicaciones web (por ejemplo, sistemas de manejo de contenido o CMS) para el sector corporativo atendiendo a un selecto grupo de clientes con presencia local e internacional.

Actualmente, las actividades principales se concentran en:

- Construcción de portales web en múltiples idiomas y que pueden ser administrados por sus propios dueños. Esto incluye la programación de módulos especiales para integrar información desde y hacia sistemas externos, desplegar datos de manera amigable o generar notificaciones automáticas dependientes de actividades de los visitantes u otros eventos.

- Apoyo en la gestión de contenido de portales web.

- Consultoría y gestión para optimizar las variables asociadas al rendimiento y desempeño de las páginas web. Teniendo especial interés en el monitoreo de presencia en buscadores, evaluación del perfil de los visitantes y garantizar un nivel adecuado de usabilidad en diferentes dispositivos, etc.

- Desarrollo de productos personalizados que complementen las ventajas y facilidades de los dispositivos móviles en sincronización con mecanismos de soporte en servidores web.
- Desarrollo de soluciones especializadas para ofrecer bajo el modelo SaaS o Software as a Service.

1.2. Misión

Proveer productos y servicios en el área de sistemas de información que permitan una comunicación efectiva de nuestros clientes con su público y también sirva como plataforma de trabajo donde se aprovechen las innovaciones y ventajas de las tecnologías más modernas.

1.3. Visión

Deseamos ser un proveedor confiable, que ofrece un alto valor agregado en cada producto o servicio que prestamos a nuestros clientes.

1.4. Ubicación del pasante

El proyecto de pasantía pertenece al grupo de desarrollo de aplicaciones y cuenta con la dirección del Presidente de la Empresa y con el apoyo de los ingenieros líderes del grupo.

CAPÍTULO II

MARCO TEÓRICO

En el presente capítulo se definen las bases teóricas sobre las cuáles se apoya el proyecto.

2.1. Bases Teóricas

2.1.1. CMS

Un Sistema de Gestión de Contenido o CMS, por sus siglas en inglés *Content Management System*, es un paquete de software que brinda cierto nivel de automatización de las tareas requeridas para administrar el contenido de manera efectiva. Un CMS permite a los usuarios crear nuevo contenido, editar contenido existente, y hacer el contenido accesible al público [1].

Para los editores el CMS les permite crear contenido nuevo, editar contenido existente, realizar procesos en el contenido mediante una interfaz de edición (referida como *back-end* que es la capa de acceso a los datos de la aplicación) y finalmente les permite colocar este contenido a disposición de otros usuarios en una interfaz (referida como el *front-end*, es decir, la capa de presentación de la aplicación).

Un CMS permite el control del contenido, esto se refiere a que mantiene un constante seguimiento del contenido (donde se encuentra el contenido, quién puede acceder a él, y cómo se relaciona con otros contenidos). Por otro lado permite la reutilización del contenido (usar contenido en más de un lugar).

Una de las mayores ventajas de el uso de CMS es que facilita las tareas mencionadas anteriormente para usuarios que no tienen preparación técnica. En el caso de la aplicación CPI, la mayoría de los usuarios no poseen conocimientos especializados en el área de computación, por lo cual es conveniente el desarrollo del sistema sobre un CMS

El CMS sobre el cual se trabajó para el desarrollo de la aplicación cuenta con funcionalidades que ya están implementadas que son necesarias para ésta, como la autenticación

para los usuarios, permisología, interfaces que facilitan el manejo de la base de datos, entre otras cosas. Cuenta con gran variedad de paquetes, librerías y módulos que facilitan el desarrollo de la aplicación.

2.1.2. Modelo Cliente-Servidor

Arquitectura de redes de computadoras ampliamente utilizado y que forma la base del uso de redes en gran medida, consta de dos entidades: un cliente y un servidor. El cliente le envía una solicitud al servidor y espera una respuesta. Luego, el servidor recibe la solicitud, lleva a cabo el trabajo requerido, o busca los datos solicitados y devuelve una respuesta al cliente [11].

El servidor mantiene una relación de uno-a-muchos con los clientes, por otro lado ambos términos pueden ser vistos como "roles", pues es posible que una máquina o proceso ejecute labores tanto de cliente como de servidor (por ejemplo, un servidor puede enviar una petición a otro servidor, si el mismo carece de los recursos que le fueron solicitados, convirtiéndose así en un cliente). Es importante destacar que los términos "cliente" y "servidor" pueden referirse tanto a máquinas como a programas o procesos. Esta arquitectura es ampliamente utilizada en aplicaciones web.

2.1.3. MVC

MVC, siglas para Modelo-Vista-Controlador, es un patrón de software utilizado ampliamente en la actualidad. Consiste en separar los datos de la aplicación (Modelo), la interfaz con el usuario (Vista), y la lógica de control (Controlador) en tres componentes distintos. Al realizar esta separación se reduce la complejidad del diseño arquitectónico y se incrementa la flexibilidad, la reusabilidad y mantenimiento del código. Adicionalmente, se pueden realizar cambios sobre un componente sin afectar a los demás, lo cual permite que cada componente tenga ciclos de desarrollo independientes del resto. [6].

Cabe destacar que este patrón es usado frecuentemente en el desarrollo de aplicaciones web, por lo que resulta bastante sencillo aplicarlo al modelo cliente-servidor utilizado en la aplicación. CPI fue desarrollado usando una plataforma de desarrollo web basada en .NET que implementa este patrón.

2.1.4. API

Un API, llamado así por sus siglas en inglés para *Application Programming Interface*, es un conjunto de comandos, funciones, protocolos y objetos que permiten exponer los datos de una aplicación de software, establecen las reglas y los mecanismos a través de los cuales se puede tener acceso a estos datos. También permite la interacción entre con algún software externo.[3]

El API sirve como intermediario entre dos aplicaciones, por ejemplo una aplicación dispone del API de otra para obtener los datos que se encuentran en la última y usarlos para proveer algún servicio a sus usuarios. En el caso de CPI se desarrolló un API para poder acceder a datos de la aplicación.

2.1.5. Servicio Web

Un servicio web es una aplicación o fuente de datos a la que se puede acceder a través de un protocolo web estándar, diseñado para soportar interacción máquina-máquina a través de una red, proveen una vía estándar para la comunicación entre distintas aplicaciones de software ejecutadas en distintas plataformas y ambientes. La mayoría de los servicios web proporcionan un API, para que se puedan acceder a los datos. [4]

Para la aplicación CPI en el desarrollo se incluyó un módulo de servicios web, el cual permite el acceso a datos de la aplicación.

2.1.6. REST

Transferencia de Estado Representacional o REST, por sus siglas en inglés, es un estilo de arquitectura para sistemas de hipermedia distribuidos (como la *World Wide Web* o red informática mundial) que define una serie de restricciones que, cuando se aplican en conjunto, enfatizan la escalabilidad de interacciones entre componentes, la generalidad de las interfaces y el despliegue independiente de componentes. [5]

Las restricciones definidas para los sistemas REST son las siguientes:

1. **Separación Cliente-Servidor** el cliente y el servidor actúan independientemente, la interacción entre ellos ocurre solo a través de solicitudes, realizadas por el cliente, y respuestas, enviadas por el servidor como una reacción a una solicitud. El servidor solo envía información cuando ésta es solicitada por algún cliente.

2. **Sin estado (*stateless* en inglés)** el servidor no guarda información de ningún usuario que use los servicios del sistema. Cada solicitud individual contiene toda la información necesaria para ser ejecutada y enviar una respuesta, independientemente de todas las demás solicitudes que sean atendidas.

3. **Permite el uso de memoria caché** la respuesta debe estar explícita o implícitamente etiquetada como *cacheable* (se puede guardar en alguna memoria caché) o *non-cacheable* (no se puede guardar en ninguna memoria caché). La ventaja del uso de memoria caché es que se pueden eliminar parcial o completamente algunas interacciones mejorando así el rendimiento del sistema.

4. **Interfaz uniforme** cada solicitud al servidor debe tener los mismos 4 componentes: un identificador del recurso deseado, en el caso de aplicaciones web este identificador puede ser el URL; las respuesta del servidor debe incluir suficiente información para que el cliente pueda modificar el recurso; toda solicitud debe contener la información necesaria para que el servidor la pueda llevar a cabo y toda respuesta del servidor debe tener la información necesaria para que el cliente la entienda; uso de hipermedia como motor del estado de la aplicación, es decir, el servidor debe poder informar al cliente las formas en las que puede cambiar el estado de la aplicación a través de enlaces de hipermedia, una página web específica se puede considerar un estado de la aplicación y enlaces incluidos en esa página se consideran transiciones a otros estados de la aplicación.

5. **Sistema por capas** entre el cliente que realiza una solicitud y el servidor que envía la respuesta final puede haber varios servidores, por ejemplo, puede haber un servidor que proporcione una capa de seguridad, otro una capa de memoria caché, y otros con otras funcionalidades. Estos servidores intermedios no deben afectar ni la solicitud ni la respuesta. Además, cada transacción (envío de la solicitud por el cliente y la subsiguiente respuesta) debe ser transparente para el cliente, es decir, el cliente no tiene conocimiento de las capas intermedias por las que pasan la solicitud y la respuesta.

Se dice que un servicio web o el API de un servicio web es *RESTful* cuando cumple con todas estas restricciones. El proyecto presente se desarrolló adhiriéndose a estas restricciones.

CAPÍTULO III

MARCO TECNOLÓGICO

En el presente capítulo se muestran las herramientas necesarias para el desarrollo del proyecto, de tal manera que el lector pueda comprender los conceptos y tecnologías asociadas con la elaboración del mismo.

3.1. Lenguajes

3.1.1. C#

C# es un lenguaje de programación desarrollado por Microsoft introducido en el año 2002, tiene seguridad de tipos y es orientado a objetos, el cual que permite a los desarrolladores crear una variedad de aplicaciones seguras y robustas que son ejecutadas en .NET Framework. Se puede usar C# para crear aplicaciones cliente de Windows, servicios CML, componentes distribuidos, aplicaciones de bases de datos, aplicaciones cliente-servidor como es el caso de CPI (la cual usa este lenguaje para desarrollar el código del *back-end*) , entre otras. [8]

3.1.2. HTML

HTML por sus siglas en inglés *Hypertext Markup Language* , es el lenguaje de marcado central de la *World Wide Web* que es la red informática mundial. Originalmente, HTML fue diseñado principalmente para describir semánticamente documentos científicos. Sin embargo, la generalidad de su diseño ha permitido que se adapte, en la actualidad, para describir otros tipos de documentos e incluso aplicaciones. [9]

Todas las vistas de la aplicación CPI fueron definidas por este lenguaje.

3.1.3. CSS

CSS por sus siglas en inglés *Cascading Style Sheets* (Hojas de Estilo en Cascada), es el lenguaje para describir la presentación de páginas web (colores, diseños, fuentes, etc), permite a los usuarios agregar estilos a documentos estructurados como HTML. Al permitir la separacion del estilo de presentacion del contenido de los documentos, se facilita el mantenimiento de los sitios, el intercambio de hojas de estilo entre páginas y la personalización de páginas en diferentes entornos.[7]

3.1.4. JavaScript

JavaScript es un lenguaje de programación interpretado conmunmente utilizado para el desarrollo web. Originalmente fue desarrollado por Netscape para crear contenido dinámico, control de multimedia, animación de imágenes, entre otras cosas a los sitios web.

Es un lenguaje de *scripting* del lado del cliente, lo que significa que el código fuente es procesado por el navegador web del cliente y no en el servidor web. [2]

3.2. Entorno de trabajo

3.2.1. .NET

3.2.2. ASP.NET

3.2.3. ASP.NET MVC

3.2.4. ASP.NET Web Api

3.2.5. ASP.NET Razor

3.2.6. Umbraco

3.3.

3.3.1. SQL Server

3.3.2. JSON

3.3.3. Ajax

3.3.4. JQuery

3.3.5. DataTables

CAPÍTULO IV

MARCO METODOLÓGICO

En este capítulo se explicará la metodología que se utilizó para el desarrollo del proyecto, que fue necesaria usar para cumplir con los objetivos planteados, conocida como Srum. A continuación se describirá el marco de desarrollo, las actividades y resultados de cada una de las etapas de esta metodología.

4.1. ¿Qué es Scrum?

Scrum es un marco de trabajo para desarrollar, entregar y mantener productos complejos, en el cual las personas pueden abordar problemas complejos adaptativos, y a su vez entregar productos del máximo valor posible productiva y creativamente. Empezó a ser usado desde principios de los años 90. Scrum es un marco de trabajo dentro del cual se pueden emplear varios procesos y técnicas. El marco de trabajo Scrum consiste en los Equipos Scrum y sus roles, eventos, artefactos y reglas asociadas. [10]

Usando correctamente este marco de trabajo se puede mostrar la eficacia relativa de las técnicas de la gestión del producto y las técnicas de trabajo, de manera que a medida que va pasando el tiempo se pueda mejorar el producto, el equipo y el entorno de trabajo. Para que esto suceda es necesario que cada componente que interactúa dentro de Scrum cumpla con su propósito específico.

4.2. Usos de Scrum

Scrum inicialmente fue desarrollado para la gestión y desarrollo de productos. Se ha usado principalmente para:

1. Investigar e identificar mercados viables, tecnologías y capacidades de productos.
2. Desarrollar productos y mejoras.

3. Liberar productos y mejoras tantas veces como sea posible durante el día.
4. Desarrollar y mantener ambientes en la Nube (en línea, seguros, bajo demanda) y otros entornos operacionales para el uso de productos.
5. Mantener y renovar productos.

4.3. Equipo de Scrum

El equipo Scrum está conformado por el Dueño del Producto (*Product Owner*), el Equipo de desarrollo (*Development Team*) y un *Scrum Master*. Los equipos Scrum son autoorganizados y multifuncionales. Los equipos autoorganizados eligen la mejor forma de llevar a cabo su trabajo y no son dirigidos por personas externas al equipo. Los equipos multifuncionales tienen todas las competencias necesarias para llevar a cabo el trabajo sin depender de otras personas que no son parte del equipo. El modelo de equipo en Scrum está diseñado para optimizar la flexibilidad, la creatividad y la productividad. [10]

4.3.1. Dueño del Producto

Es el responsable de maximizar el valor del producto resultante del trabajo del Equipo de Desarrollo. Es la única persona responsable de gestionar la Lista del Producto (*Product Backlog*), esto incluye:

- Expresar claramente los elementos de la lista.
- Ordenar los elementos de la lista, para alcanzar objetivos y misiones de manera eficiente.
- Optimizar el valor del trabajo que el Equipo de Desarrollo realiza.
- Asegurar que la lista sea visible, transparente y clara para todos y que muestra aquello en lo que el equipo trabajará a continuación.
- Asegurar que el Equipo de Desarrollo entiende los elementos de la Lista del Producto al nivel necesario.
- Expresar los items del *Product Backlog*

4.4. Eventos de Scrum

4.5. Artefactos de Scrum

4.5.1. Lista de objetivos del producto

La lista de objetivos del producto es una lista ordenada de todas las tareas que deben ser completadas para entregar el producto. Esta lista nunca está completa, la primera versión muestra los primeros requerimientos conocidos y mejor entendidos. [10] La lista de objetivos del producto evoluciona junto con el producto para actualizar los requerimientos dependiendo de las exigencias del cliente y del mercado.

4.5.2. Lista de tareas de la iteración

La loita de tareas de la iteración es el conjunto de tareas de la lista de objetivos del producto seleccionadas para completar en un Sprint, puede verse como una predicción de lo que estará completado al llevarse a cabo es Sprint. [10] Esta lista va a ir cambiando a medida que se identifiquen tareas y trabajo que debe ser realizado para completar los items durante el desarrollo del Sprint.

4.5.3. Tablero de tareas

El tablero de tareas es una tabla utilizada para gestionar el estado de los objetivos de la lista de objetivos del producto, contiene 4 columnas:

- **Por hacer**: contiene las tareas que no se han empezado.
- **Haciendo**: contiene las tareas en las que se está trabajando actualmente.
- **Hechas**: contiene las tareas que se han completado.
- **Mejoras**: contiene tareas u objetivos que salen del alcance del proyecto actual y que pueden ser desarrolladas para una versión futura del producto.

Cada una de las tareas en este tablero puede ser asignada a una persona específica, también se pueden etiquetar dependiendo de la naturaleza de la tarea.

Para el proyecto presente se escribió una lista de objetivos del producto al inicio y se fue actualizando a medida que se avanzó en el desarrollo.

CAPÍTULO V

DESARROLLO

5.1. Fase I: Iniciación

5.1.1. Levantamiento de información

5.1.2. Plan de proyecto

5.1.3. Visión del Sistema

Para esta sección hay que hacer un *Documento de Visión del Sistema.*

5.1.4. Lista Inicial de Requerimientos

Para esta sección hay que hacer un *Listado de Requerimientos Funcionales.*

5.1.5. Modelo Inicial de Casos de Uso

Para esta sección hay que hacer un *Documento de Especificación de Requerimientos del Sistema.*

5.1.6. Glosario del Sistema

Para esta sección hay que hacer un *Glosario del Sistema.*

5.2. Fase II: Elaboración

5.2.1. Lista de Requerimientos Suplementarios

5.2.2. Descripción de la Arquitectura de Software

5.2.2.1. Vista Lógica

Modelo Conceptual

Diagrama de Clases

Modelo Entidad-Relación

5.2.2.2. Vista de Implementación

Aquí hay que hacer el *Diagrama de Componentes del Sistema* para mostrar la separación del sistema en las capas de MVC.

5.2.2.3. Vista de Implantación?

Hay que saber si esta vista es relevante para el sistema.

5.2.2.4. Vista de Procesos?

La misma pregunta que para la sección anterior.

5.2.2.5. Vista de Casos de Uso

Aquí hay que hacer una lista con los casos de uso más importantes detallados.

5.3. Fase III: Construcción

Describir el proceso de desarrollo en pasos.

5.4. Fase IV: Transición

5.4.1. Plan de Pruebas

CONCLUSIONES Y RECOMENDACIONES

Conclusiones aquí.

REFERENCIAS

[1] Barker, Deane: *Web Content Management: Systems, Features, and Best Practices*. O'Reilly Media, 2016, ISBN 978-1491908129.

[2] Christensson, Per: *JavaScript Definition.*, 2014. `https://techterms.com/definition/javascript`, visitado el 2018-09-25.

[3] Christensson, Per: *API Definition.*, 2016. `https://techterms.com/definition/api`, visitado el 2018-09-02.

[4] Christensson, Per: *Web Service Definition.*, 2017. `https://techterms.com/definition/web_service`, visitado el 2018-09-25.

[5] Fielding, Roy T.: *Architectural Styles and the Design of Network-based Software Architectures*. Tesis de Doctorado, University of California, 2000.

[6] Krasner, Glen E. y Pope, Stephen T.: *A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80*. Journal of Object-Oriented Programming, 1:26–49, 1988.

[7] Lie, Håkon Wium, Lilley, Chris, Jacobs, Ian y Bos, Bert: *Cascading Style Sheets, level 2 (CSS2 Specification)*. W3C Recommendation, W3C, Abril 2008. http://www.w3.org/TR/2008/REC-CSS2-20080411/.

[8] Microsoft: *Introduction to the C# Language and the .NET Framework.*, 2015. `https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework`, visitado el 2018-09-21.

[9] Moon, Sangwhan, Leithead, Travis, Eicholz, Arron, Danilo, Alex y Faulkner, Steve: *HTML 5.2*. W3C Recommendation, W3C, Diciembre 2017. https://www.w3.org/TR/2017/REC-html52-20171214/.

[10] Schwaber, Ken y Sutherland, Jeff: *The Scrum Guide*, 2017. `https://www.scrumguides.org/scrum-guide.html`.

[11] Tanenbaum, Andrew S. y Wetherall, David J.: *Computer Networks*. Pearson, 2010, ISBN 978-0132126953.