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 [6].

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. [5].

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. [**?**]

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 este capítulo se definen las tecnologías y herramientas utilizadas para el desarrollo del sistema.

3.1. Lenguajes

3.1.1. C#

3.1.2. HTML

3.1.3. CSS

3.1.4. JavaScript

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

CAPÍTULO IV

MARCO METODOLÓGICO

Para poder cumplir con los objetivos del proyecto fue necesario utilizar la metodología de desarrollo de software llamada *Proceso Unificado de Rational* o RUP (por sus siglas en inglés) de IBM. A continuación se presentará brevemente la estructura de la metodología en cuestión, incluyendo sus fases y actividades asociadas.

4.1. La metodología RUP

El Proceso Unificado de Rational es un proceso de ingeniería el cual provee un enfoque disciplinado hacia la asignación de tareas dentro de una organización orientada al desarrollo. Su objetivo es asegurar la producción de software de alta calidad que cumpla con las necesidades de sus usuarios finales dentro de tiempo y presupuesto predecible. [Kru00]

El Proceso Unificado de Rational captura varias de las *mejores prácticas* en el desarrollo de software moderno de una manera que se puedan adaptar a una gama amplia de proyectos y organizaciones. Las (6) prácticas principales cubiertas por el proyecto son:

1. Desarrollar software de manera iterativa.
2. Administrar requerimientos.
3. Utilizar arquitecturas basadas en componentes,.
4. Modelar el software de manera visual.
5. Verificar la calidad continuamente.
6. Mantener el control sobre los cambios realizados al software.

La Figura x muestra la arquitecutra general del Proceso Unificado de Rational. El modelo esta compuesto por dos dimensiones: el horizontal representa el tiempo y se ven reflejados los distintos ciclos de vida del software a medida que este se va desarollando;

mientras que en el eje vertical se reflejan los principales flujos de trabajo, los cuales se encargan de agrupar actividades segun su naturaleza. Por otra parte, el eje horizontal representa el aspecto dinámico del software a medida que este es aplicado. Se ve reflejado en la forma de ciclos, fases, iteraciones e hitos. A su vez, el eje vertical muestra el aspecto estático del software, describiéndolo en términos de actividades, flujos de trabajo, artefactos y trabajadores.

4.2. Fases de RUP

RUP divide el proceso de desarollo en cuatro fases. Dentro de cada fase se puede realizar una cantidad variable de iteraciones dependiendo de la envergadura del proyecto. Cada fase hace hincapié sobre distintas actividades asociadas al desarrollo del proyecto, y producen como resultado una serie de artefactos e hitos (que deben cumplirse) con la culminación de la fase. Las fases del proceso se discuten a continuación.

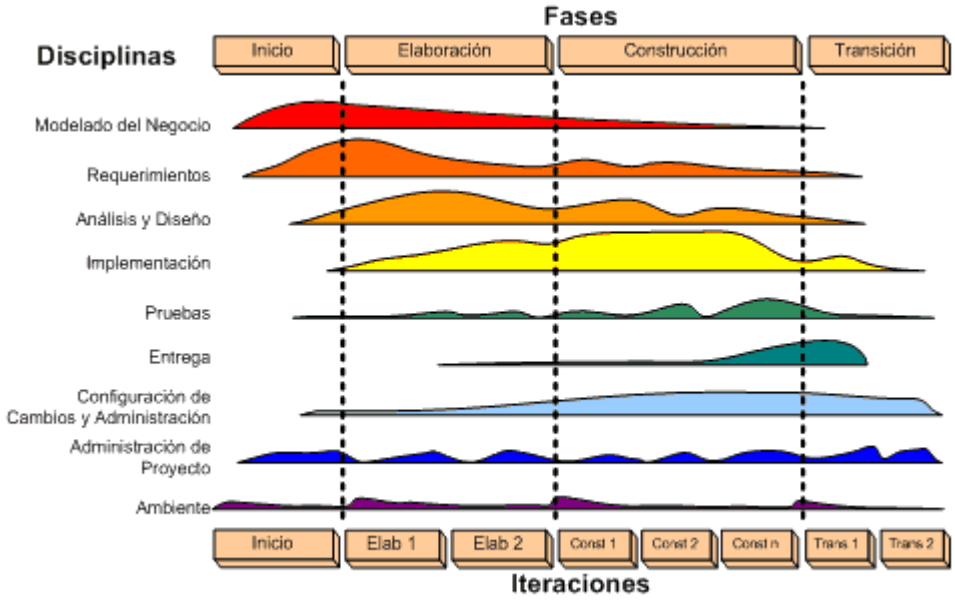

Figura 4.1: Arquitectura general del Proceso de Unificación Rational.

4.2.1. Iniciación

Es la primera fase del Proceso Unificado de Rational, en la cual el ímpetu inicial para Desarrollar software (una idea, un prototipo, una solicitud de propuesta, una evolución sobre una generación previa) se materializa hasta el punto de obtener el respaldo suficiente (al menos dentro de la organización) para poder entrar en la etapa de elaboración. [Kru00] El objetivo principal de la fase de inciación es informar y mantener de acuerdo al público interesado (stakeholders) sobre los objetivos del ciclo de vida del proyecto. Otros objetivos de la fase incluyen:

- Establecer el alcance y condiciones límite del proyecto, incluyendo el concepto de operación, criterios de aceptación y descripciones sobre lo que se desea que contenga el producto.
- Desarrollar los casos de uso críticos del sistema, es decir, aquellos escenarios principales que definirán la funcionalidad del proyecto y en los cuales se dedicará el mayor esfuerzo de desarrollo.
- Estimar costos, tiempos de desarrollo y riesgos globales para el proyecto.

Los artefactos producidos para el cierre de la fase son los siguientes:

- Un documento de visión que muestre los principales requerimientos, funcionalidades y restricciones del proyecto.
- Un modelo de casos de uso incial, en donde se muestren los casos de uso y actores que se pueden identificar para la fase en cuestión
- Un glosario del proyecto inicial.
- Una propuesta de negocio incial que incliya: el contexto del negocio, criterios de éxito y predicciones financieras.
- Un plan de proyecto en donde se muestren las fases (e iteraciones) a llevarse a cabo.

El hito de la fase de inciación es la obtención y evaluación de los *Objetivos del Ciclo de Vida* del proyecto. Al alcanzar este hito debe realizarse una serie de evaluaciones que deben cumplise satisfactoriamente antes de continuar a la próxima fase del proyecto. Con este hito se debe validar que las partes interesadas del proyecto hayan llegado a un acuerdo sobre el alcance del mismo y las estimaciones de costos y tiempo. Adicionalmente, se debe

validar la compresión de requerimientos del cliente y que estos se vean plasmados en los casos de uso definidos. Por último, se debe validar la credibilidad de las prioridades y riesgos asociados al proceso de desarrollo. De todos los hitos presentes en la metodología, este es considerado como el más importante, ya que su resultado puede llevar a la cancelación del proyecto o a que el mismo tenga que ser re-planteado.

4.2.2. Elaboración

Es la fase del proyecto orientada a terminar de definir la visión del producto y la arquitectura a utilizar. [Kru00] La fase de elaboración tiene como objetivo analizar los problemas asociados al dominio de la aplicación, establecer una arquitectura sólida para el producto, elaborar un plan de proyecto definitivo y eliminar la mayor cantidad de elementos de alto riesgo. Cada decisión con respecto a la arquitectura del producto deben hacerse teniendo un conocimiento pleno del alcance del producto, su funcionalidad principal y requerimientos no funcionales (por ejemplo, requerimientos de desempeño). Los objetivos principales de la fase son:

- Definir y validar la arquitectura del producto de la manera más rápida posible.
- Definir y validar la visión del proyecto.
- Contruir un plan (de alta fidelidad) para la etapa de Construcción.
- Demostrar que la arquitectura desarrollada pueda soportar la visión por un costo y un tiempo razonable.

Algunos artefactos como resultado de la ejecución de la fase se muestran a continuación:

- Un modelo de casos de uso completado un 80

4.2.3. Construcción

4.2.4. Transición

4.3. Fases contempladas para el proyecto

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] Booth, David, McCabe, Francis, Ferris, Christopher, Newcomer, Eric, Orchard, David, Champion, Mike y Haas, Hugo: *Web Services Architecture*. W3C Note, W3C, Febrero 2004. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/.

[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] 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.

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