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

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

Sartenejas, Octubre de 2018

V. HERNÁNDEZ 2018	DESARROLLO DE LA VERSIÓN 2 DE LA APLICACIÓN WEB CPI	USB INGENIERIA 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 de la Computación

Sartenejas, Octubre de 2018

Página reservada para el acta de evaluación

AGRADECIMIENTOS

sdfsdsdgsdgwqeqeqw

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

RESUMEN

El presente informe describe las actividades realizadas durante el proyecto de pasantía larga en el período Abril-Septiembre 2018 en la empresa IKêls Consulting, el cual consistió en desarrollar la versión 2 de la aplicación web CPI que permite realizar comparaciones de precios entre productos ofrecidos por diferentes tiendas de ventas al detal. La primera versión de esta aplicación fue desarrollada hace más de diez años usando tecnología Classic ASP y VBScript por lo que la empresa tenía la necesidad de realizar un proceso de re-ingeniería para poder adaptar la solución a una plataforma y modelos más modernos. La aplicación cuenta con tres módulos iniciales, un primer módulo de administración de cadenas, productos y categorías, dedicado a la creación, edición y eliminación de registros de estas entidades, así como listar los productos pertenecientes a una cadena. El segundo módulo de administración de precios de productos. Y un tercer módulo para la generación de reportes, el mismo consiste en mostrar los reportes comparativos entre los precios de productos entre las diferentes cadenas, reportes históricos y por último un reporte de

los productos por categorías de una cadena. La metodología utilizada en el desarrollo del proyecto fue Scrum, ya que se deseaba un desarrollo iterativo. Para lograr los objetivos, se emplearon diversas herramientas y tecnologías, pero principalmente la plataforma Umbraco, la cual está construida sobre la plataforma ASP.NET de Microsoft, C#, HTML, CSS, JavaScript, Highcharts y SQL Server. Una vez culminado el proyecto, se obtuvo una versión de la aplicación con tres módulos con una serie de funcionalidades que pueden ser resumidas en las siguientes: crear, modificar y eliminar cadenas, productos, categorías y precios de productos, generación de reportes comparativos e importación por lote de precios mediante un formato de Microsoft Excel.

Palabras cláves: Aplicación Web, Reingeniería, Módulos, Desarrollo Iterativo, Plataforma Umbraco.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

INTRODUCCIÓN

iKêls Consulting [1] es una empresa especializada en el desarrollo de aplicaciones y sitios web dedicada al sector corporativo utilizando Umbraco como la plataforma principal de manejo de contenido (CMS) [2]. Umbraco es una herramienta de gestión de contenido de código abierto que desarrollado sobre ASP.NET [3] para sistemas Windows de Microsoft y usando el lenguaje de programación C# [4], cuenta con una gran aceptación a nivel mundial y una amplia comunidad de desarrolladores.

Antecedentes

iKêls posee entre sus activos una aplicación web llamada CPI que permite realizar comparaciones de precios entre productos ofrecidos por diferentes tiendas de ventas al detal, sin embargo esta aplicación aplicación se desarrolló hace más de 10 años usando tecnología Classic ASP y VBScript por lo que la empresa necesita realizar un proceso completo de re-ingeniería para adaptar esta solución a plataformas y modelos de gestión más modernos. (No estoy segura si colocar este párrafo como antecedentes)

Planteamiento del problema

Las cadenas de supermercados operan en un entorno comercial muy competitivo con márgenes muy pequeños de rentabilidad, por lo que para poder diferenciarse del resto de opciones en el mercado, deben confeccionar una estructura de precios que resulte simultáneamente atractiva para atraer a sus clientes así como rentable para mantener el negocio a flote. Para lograr lo anterior es imprescindible conocer los rangos dónde se mueve el precio de cada producto de manera constante. Debido al volumen de datos involucrados (miles de productos en varios competidores) se necesita una aplicación que permita consolidar y analizar la información de manera rápida, sencilla y efectiva.

Justificación e importancia

Los departamentos de análisis de precios de las cadenas minoristas invierten la mayor parte de sus recursos en actividades de investigación. Al disponer de una herramienta que facilite su trabajo permitirá responder más rápidamente ante los cambios del mercado así como cubrir un mayor volumen de productos en un menor tiempo. Poder identificar de manera oportuna aquellos productos donde se requiere ajustes de precios (ya sea para competir con otras cadenas o para no perder dinero), puede ser el factor clave para definir los resultados netos de la empresa.

Objetivo general

Crear una nueva versión de la aplicación web CPI integrada con la plataforma de manejo de contenido Umbraco para el ingreso y administración de precios de productos.

Objetivos específicos

- Crear el módulo de administración de cadenas, productos y categorías.
  - Desarrollar interfaz para creación, edición y eliminación de registros.
  - Asignar productos a categorías
  - Listar productos con mecanismo de búsqueda.
- Crear el módulo de administración de precios.
  - Desarrollar interfaz para creación, edición y eliminación de los registros de precios.
  - Importar un lote de precios usando un archivo en formato Microsoft Excel
- Crear el módulo de generación de reportes.
  - Crear reporte para comparar precios de productos por categorías
  - Crear reporte comparativo entre precios de productos de diferentes cadenas.
  - Crear reporte histórico de precios de un producto

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 [1] 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 [2]) 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* (Software como servicio) [5].

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. A continuación se muestra el organigrama de la empresa.

1.5. Organigrama

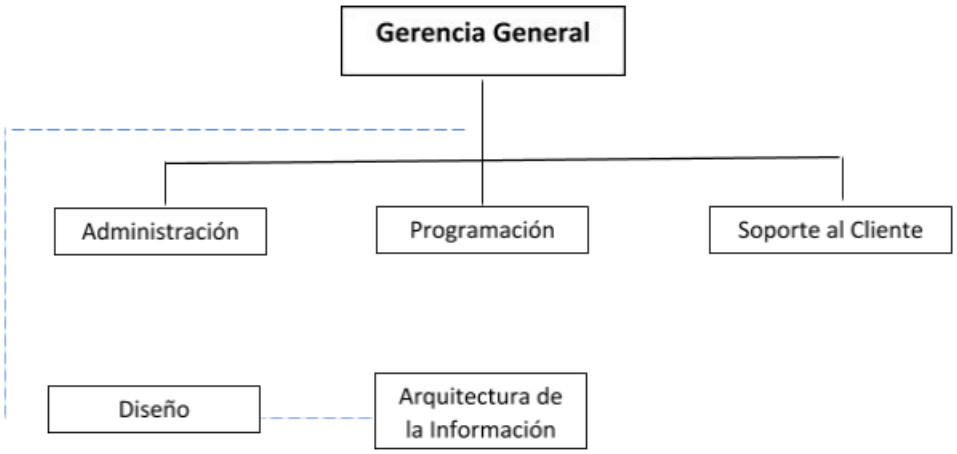

Figura 1.1: Organigrama de la Empresa. Fuente: Elaboración propia.

CAPÍTULO II

MARCO TEÓRICO

En el presente capítulo se definen las bases teóricas sobre las cuáles se apoya el proyecto. Para el desarrollo de la aplicación *CPI* se utilizó Umbraco [6] como plataforma principal para de manejo de contenido (CMS) [2] y utilizando el patrón MVC [7].

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 [2].

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 [8].

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 [7].

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 [9] que implementa este patrón.

2.1.4. API

Un API, llamado así por sus siglas en inglés para *Application Programming Interface* (Interfaz de Programación de Aplicaciones), 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 con algún software externo [10].

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 [11].

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 [12].

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 que realiza el cliente, y las respuestas que envía 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 componentes:

   - Identificador del recurso deseado (en el caso de aplicaciones web este identificador puede ser el URL).
   - Respuesta del servidor que debe incluir suficiente información para que el cliente pueda modificar el recurso (la información necesaria para que el servidor pueda llevar a cabo la solicitud y toda respuesta del servidor debe tener la información necesaria para que el cliente la entienda.
   - Usar hipermedia como motor del estado de la aplicación, lo que quiere decir que 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.

2.1.7. SAAS

Un Software como Servicio o SAAS, por sus siglas en inglés *Software as a Service*, es un modelo para la distribución de software donde los clientes acceden al software a través de Internet. En SaaS, un proveedor de servicios aloja la aplicación en su centro de datos y un cliente accede a ella a través de un navegador web estándar [5].

2.1.8. Modelo "4+1"de Kruchten

El modelo "4+1" de Kruchten, es un modelo de vistas diseñado por el profesor Philippe Kruchten y que encaja con el estándar "IEEE 1471-2000" que se utiliza para describir la arquitectura de un sistema software intensivo basado en el uso de múltiples vistas que deben estar relacionadas entre sí (ver Figura 2.1) [13]. Cada una de estas vistas permite abordar por separado las inquietudes de los distintos ïnteresados"de la arquitectura: usuario final, desarrolladores, ingenieros de sistemas, gerentes de proyectos, entre otros, y manejar por separado los requisitos funcionales y no funcionales. Las vistas que abarca este modelo son:

- Vista Lógica: En esta vista se representa la funcionalidad que el sistema proporciona a los usuarios finales, es decir, se ha de representar lo que el sistema debe hacer y las funciones y servicios que ofrece.
- Vista de Despliegue: En esta vista se muestra el sistema desde la perspectiva de un programador y se ocupa de la gestión del software, es decir, se muestra como esta dividido el sistema de software en componentes y las dependencias que hay entre estos componentes.
- Vista de Procesos: En esta vista se muestran los procesos que hay en el sistema y la forma en la que se comunican estos procesos, se muestra desde la perspectiva de un integrador de sistemas, el flujo de trabajo paso a paso de negocio y operaciones de los componentes que conforman el sistema.
- Vista Física: Se muestra desde la perspectiva de un ingeniero de sistemas todos los componentes físicos del sistema, así como las conexiones físicas entre esos componentes que conforman la solución.
- Vista de Escenarios: Esta vista va a ser representada por los casos de uso software y va a tener la función de unir y relacionar las otras 4 vistas.

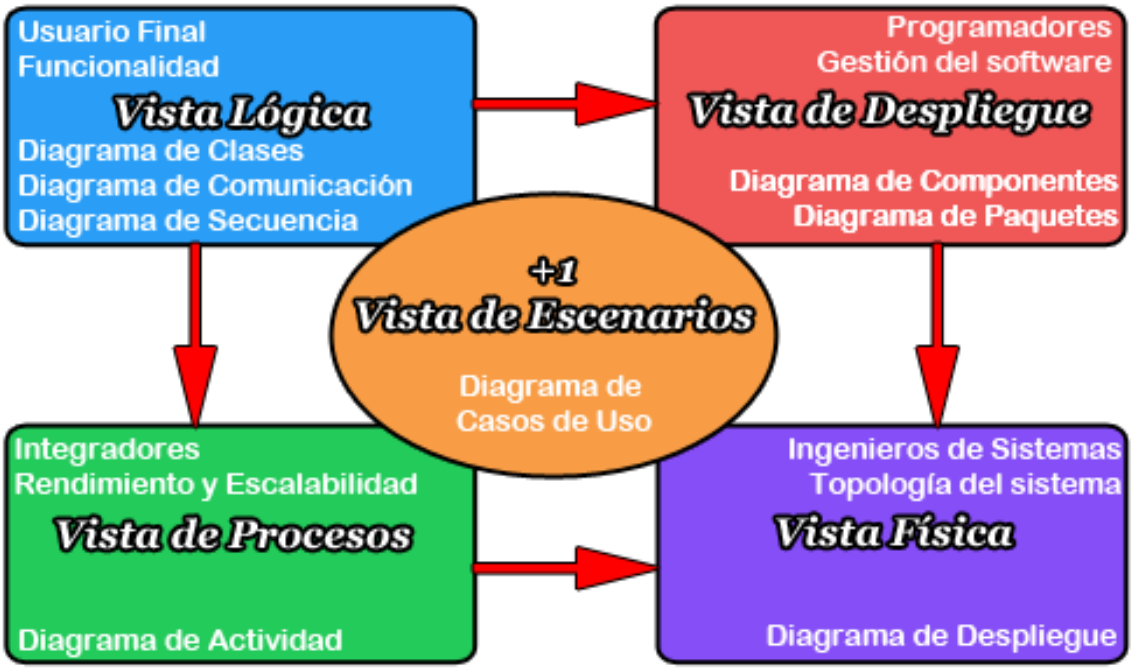

Figura 2.1: Modelo “4+1” de Kruchten

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 [4].

3.1.2. HTML

HTML por sus siglas en inglés *Hypertext Markup Language* (Lenguaje Marcado de Hipertexto), 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 [14].

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 [15].

3.1.4. JavaScript

JavaScript es un lenguaje de programación interpretado comúnmente 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 [16].

3.1.5. ASP.NET Razor

Razor es un lenguaje de marcado, no es un lenguaje de programación, que permite incluir código basado en servidor (Visual Basic y C#) en páginas web.

El código basado en servidor puede crear contenido web dinámico sobre la marcha, mientras que una página web se escribe en el navegador. Cuando se hace un llamado a una página web, el servidor ejecuta el código basado en servidor dentro de la página antes de devolver la página al navegador, y al ejecutarse en el servidor, este código puede realizar tareas complejas.

Razor se basa en ASP.NET y está diseñado para crear aplicaciones web. Tiene el poder del marcado ASP.NET tradicional, pero es más fácil de usar y más fácil de aprender [17].

3.2. Frameworks

3.2.1. .NET

.NET es una plataforma de desarrollo de código abierto, multiplataforma y gratuita para crear muchos tipos diferentes de aplicaciones [9].

3.2.2. ASP.NET

ASP.NET es un modelo de desarrollo Web unificado que incluye los servicios necesarios para crear aplicaciones Web de clase empresarial con un mínimo de codificación. ASP.NET es parte de .NET Framework y al codificar las aplicaciones en ASP.NET se tiene acceso a las clases de .NET Framework [3].

3.2.3. ASP.NET MVC

ASP.NET MVC es un marco de trabajo para crear aplicaciones web escalables y basadas en estándares que usan patrones de diseño bien establecidos (patrón MVC) y el poder de ASP.NET y .NET Framework [18].

Para el desarrollo de uno de los módulos de la aplicación **CPI_Core**, se utilizó este marco de trabajo, en donde se encuentran los controladores de la misma.

3.2.4. ASP.NET Web Api

ASP.NET Web API es un marco de trabajo que facilita la creación de servicios HTTP que llegan a una amplia gama de clientes, incluidos navegadores y dispositivos móviles. ASP.NET Web API es una plataforma ideal para crear aplicaciones RESTful en .NET Framework [19].

3.2.5. Umbraco

Umbraco es un Sistema de Gestión de Contenido de código abierto gratuito desarrollado sobre ASP.NET, la primera versión de código abierto de Umbraco se lanzó el 16 de febrero del año 2005 [6].

El principal objetivo de esta plataforma es brindar flexibilidad para que se puedan editar y hacer las cosas de la manera en la que se necesiten realizar, tiene todas las funcionalidades necesarias para desarrollar aplicaciones web como: la autenticación de usuarios, permisología, interfaces para manejar la base de datos, entre muchas más. Además para agregar funcionalidades extras a la versión básica de Umbraco cuenta con un repositorio de paquetes y extensiones que proveen variedad de librerías y módulos para facilitar el desarrollo del sistema.

IKêls Consulting [1] tiene varios años de experiencia desarrollando aplicaciones web usando Umbraco, por lo que resultó de mucha ayuda para el desarrollo de la aplicación. *CPI* fue desarrollada sobre Umbraco v7.10.4

3.3. Control de Versiones

3.3.1. Git

Git es un sistema de control de versiones distribuido de código abierto y gratuito, diseñado para manejar desde proyectos pequeños hasta proyectos muy grandes con rapidez y eficiencia [20]. Fue diseñado por Linus Torvalds en el año 2005 pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones para facilitar el trabajo de varios desarrolladores sobre un mismo código fuente. Permite llevar el registro de los cambios de archivos y sincronizar el trabajo que varias personas realizan sobre archivos compartidos.

3.4. Manejador de Base de Datos

3.4.1. SQL Server

SQL Server es una parte central de la plataforma de datos de Microsoft. SQL Server es un líder de la industria en sistemas operativos de gestión de bases de datos (*ODBMS*)[21]. Entre las tecnologías que tiene SQL Server la que se utilizó para la aplicación fue el motor de base de datos, el cual es el servicio principal para almacenar, procesar y proteger datos. El motor de base de datos proporciona acceso controlado y rápido procesamiento de transacciones para cumplir con los requisitos de las aplicaciones que requieren los datos más exigentes dentro de su empresa.

3.5. Entorno de Trabajo

3.5.1. Visual Studio

Visual Studio es un entorno de desarrollo integrado que permite editar, depurar, compilar código para luego publicar una aplicación [22]. Este entorno de desarrollo (*IDE*)

incluye una gran cantidad de funcionalidades para facilitar el desarrollo de software, entre las cuales están la depuración del código con información detallada de las variables y otras entidades del programa, instrucciones de compilación complejas para la aplicación, descarga y actualización paquetes y librerías, integración con Git, *IntelliSense* de Microsoft como ayuda de codificación y la depuración de una aplicación para ver el valor de una variable durante la ejecución del programa y para la completación de código usando el contexto de la aplicación (clases, relaciones y métodos), entre otras.

3.6. JSON

JSON por sus siglas en inglés *JavaScript Object Notation* es una sintaxis de texto que facilita el intercambio de datos estructurados entre todos los lenguajes de programación. Está basado en un subconjunto del lenguaje JavaScript. Utiliza en su sintaxis llaves, corchetes, dos puntos y comas lo cual es útil en muchos contextos y aplicaciones [23]. Está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

3.7. Ajax

Ajax por sus siglas en inglés *Asynchronous JavaScript And XML* es una combinación de tecnologías de desarrollo web utilizadas para crear sitios web dinámicos. Los sitios que utilizan Ajax combinan JavaScript y XML para mostrar contenido dinámico [24]. Con "asíncrono" se refiere a la forma en la que se realizan las solicitudes al servidor web, ya que al enviar una solicitud al servidor web, éste puede recibir datos que luego pueden ser mostrados en la página web.

3.8. Highcharts

Highcharts es una biblioteca de gráficos escrita en JavaScript puro, que ofrece una manera fácil de agregar gráficos interactivos a su sitio web o aplicación web. Fue lanzado en el año 2009, [25].

En la aplicación *CPI* se usó para realizar los gráficos de barra, boxplot y dispersión que muestran los reportes de precios de los productos, de los productos por categorías, y el histórico de precios de un producto.

3.9. DataTables

DataTables es un complemento para la biblioteca jQuery de Javascript. Es una herramienta altamente flexible, y su objetivo es mejorar la accesibilidad de los datos en las tablas HTML [26].

Este complemento fue usado para cada una de las tablas de la aplicación *CPI*.

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 Scrum. 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 [27].

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 [27].

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*

4.3.2. Equipo de Desarrollo

El Equipo de Desarrollo está conformado por profesionales que realizan el trabajo de entregar un Incremento (ver Sección 4.5.3) de producto "Terminado" que potencialmente se puede poner en producción al final de cada Sprint (ver Sección 4.4.1). Cabe destacar que solo los miembros del Equipo de Desarrollo participan en la creación del Incremento, ellos se organizan y gestionan su propio trabajo. Los Equipos de Desarrollo tienen las siguientes características:

- Son autoorganizados. Nadie (ni siquiera el Scrum Master) indica al Equipo de Desarrollo cómo convertir elementos de la Lista del Producto en Incrementos de funcionalidad potencialmente desplegables.
- Los Equipos de Desarrollo son multifuncionales, esto es, como equipo cuentan con todas las habilidades necesarias para crear un Incremento de producto.
- Scrum no reconoce títulos para los miembros de un Equipo de Desarrollo independientemente del trabajo que realice cada persona
- Los Miembros individuales del Equipo de Desarrollo pueden tener habilidades especializadas y áreas en las que estén más enfocados, pero la responsabilidad recae en el Equipo de Desarrollo como un todo.

4.3.3. Scrum Master

El Facilitador (o *Scrum Master*) es el responsable de promover y apoyar Scrum como está definido en la Guía de Scrum [27]. Esto lo logran ayudando a todo el Equipo Scrum a entender la teoría, práctica, reglas y valores de Scrum [27].

4.4. Eventos de Scrum

Para minimizar y regularizar la necesidad de reuniones no definidas en Scrum existen eventos predefinidos. Los eventos son bloques de tiempo (*label-boxes*), de tal modo que todos tienen una duración máxima. En el caso de un Sprint (ver Sección 4.4.1), su duración es fija y no puede acortarse ni alargarse, el resto de eventos pueden terminar antes siempre y cuando se cumplan con el objetivo del evento [27].

4.4.1. Sprint

El Sprint es el corazón de Scrum, es un bloque de tiempo de aproximadamente un mes de duración durante el cual se crea un Incremento (ver Sección 4.5.3) del producto "Terminado" utilizable y potencialmente desplegable. Cada nuevo Sprint comienza inmediatamente después de la finalización del Sprint anterior [27].

Los Sprint están compuestos por la Planificación del Sprint (*Sprint Planning*), los Scrums Diarios (*Daily Scrums*), el trabajo de desarrollo, la Revisión del Sprint (*Sprint Review*), y la Retrospectiva del Sprint (*Sprint Retrospective*).

4.5. Artefactos de Scrum

Los artefactos de Scrum representan trabajo o valor en diversas formas que son útiles para proporcionar transparencia y oportunidades para la inspección y adaptación. Los artefactos definidos por Scrum están diseñados específicamente para maximizar la transparencia de la información clave, necesaria para asegurar que todos tengan el mismo entendimiento del artefacto [27].

4.5.1. Lista de Producto

La Lista de Producto es una lista ordenada de los objetivos y requisitos que son necesario en el producto. El Dueño de Producto (Product Owner) es el responsable de esta lista, incluyendo su contenido, disponibilidad y ordenación. La Lista de Producto va cambiando constantemente a medida de que el producto y el entorno en el que se usará también lo hacen, es decir cambia mientras se van identificando necesidades para que el producto pueda ser adecuado, competitivo y útil. La Lista de Producto enumera todas las características, funcionalidades, requisitos, mejoras y correcciones que constituyen cambios a realizarse sobre el producto para entregas futuras [27].

4.5.2. Lista de Pendientes del Sprint

La lista de Pendientes del Sprint es el conjunto de elementos seleccionados para realizar en el Sprint de la Lista de Productos, más un plan para entregar el Incremento (ver Sección 4.5.3 del producto y cumplir los objetivos del Sprint [27]. Esta lista la realiza el Equipo de Desarrollo y se basa en la funcionalidad que formará parte del próximo Incremento.

4.5.3. Incremento

El Incremento es la suma de todos los elementos de la Lista de Producto completados durante un Sprint y el valor de los incrementos de todos los Sprints anteriores. Al final de un Sprint el nuevo Incremento debe estar "Terminado", lo cual significa que está en condiciones de ser utilizado y que cumple la Definición de "Terminado" del Equipo Scrum (cada equipo tiene su propia definición de un producto o Incremento terminado). El incremento es un paso hacia una visión o meta. El incremento debe estar en condiciones de utilizarse sin importar si el Dueño de Producto decide liberarlo o no [27].

CAPÍTULO V

DESARROLLO

En este capítulo se presentan las actividades realizadas durante el desarrollo del proyecto correspondientes al período de pasantías Abril-Septiembre 2018. El proyecto inicialmente tenía planteado el desarrollo de cuatro módulos:

1. Módulo de administración de cadenas, tiendas, zonas, productos y categorías: El alcance inicial de este módulo consistía en el desarrollar la interfaz gráfica para crear, editar y eliminar los registros de cada una de estas entidades, asignar tiendas a cadenas, asignar productos a categorías, y desarrollar la funcionalidad para listar los registros de productos con mecanismo de búsqueda. Para el desarrollo de este módulo se tuvo una duración de 1 Sprint 4.4.1 sin embargo no se logró completar todo el módulo en ese tiempo, y se quedó por desarrollar las entidades tiendas y zonas para el último Sprint ya que las entidades de mayor relevancia ya estaban implementadas, pero como se tuvo otros retrasos durante el desarrollo del proyecto, el Dueño del Producto decidió dejar estas entidades para una próxima versión de la aplicación, debido a que con las entidades implementadas se podían utilizar todas las funcionalidades planificadas.

2. Módulo de administración de precios: Este módulo incluye la interfaz de creación, edición y eliminación de registros, el desarrollo de un mecanismo para importar los precios en lote usando un archivo en formato Microsoft Excel y de igual manera importar por lote mediante webservice desde una fuente externa en formato JSON, sin embargo ésta última no se realizó ya que este módulo estaba planificado para ser desarrollado en 1 Sprint, y no se tuvo el tiempo suficiente para desarrollar esta funcionalidad, por lo que el Dueño del Producto decidió dejarlo para una próxima versión ya que con un método de importación de precios bastaba para esta versión de la aplicación.

3. Módulo de generación de reportes: Para este módulo se planteó el desarrollo de la funcionalidad de generar reportes comparativos para:

- Gráfico de dispersión de precios / categorías.
- Gráfico de barras para un producto seleccionado separando precio normal y oferta.
- Gráfico tipo box-plot para el histórico de precios de un producto.

Implementar un mecanismo automático para generar reportes de manera frecuente y enviar alertas de correo con el resultado, y finalmente un mecanismo para exportar los reportes en un formato tipo tabla en Microsoft Excel, sin embargo estas dos últimas funcionalidades se decidieron dejar para una próxima versión del sistema, ya que el desarrollo de la primera funcionalidad tomó más tiempo del pensado, debido a su complejidad.

4. Módulo de preferencias para usuario: Este módulo no se desarrolló, debido a que el mismo tenía directa relación con las dos últimas funcionalidades del módulo anterior y al no desarrollarlas, este módulo no tenía sentido.

El proyecto se dividió en 3 Fases :

- Fase de Inducción, en la cual se hace la introducción del proyecto, el pasante se familiarizó con las herramientas y tecnologías necesarias para realizar el proyecto.
- Fase de Desarrollo la cual se dividió en 5 Sprints, en esta fase se hace la implementación de los diferentes módulos de la aplicación.
- Fase de Documentación en la cual se realiza parte de la documentación necesaria para la aplicación.

A continuación, se explicará en detalle en qué consiste cada Fase.

5.1. Inducción

Durante esta Fase el pasante realizó un proceso de familiarización con la empresa junto con el estudio de herramientas y lenguajes de trabajo que fueron empleados a lo largo del desarrollo del proyecto. Realizó varios cursos tutoriales (Umbraco Course level 1, Learn ASP.NET MVC de Microsoft Virtual Academy, entre otros), y utilizó recursos (guía de Umbraco) que la empresa proporcionó para tener los conocimientos necesarios para llevar a cabo el proyecto. Además investigó sobre herramientas como: ASP.NET, SQL Server, Highcharts y Visual Studio.

La duración de esta fase fue de una semana, sin embargo, el pasante tuvo que mantenerse en constante búsqueda de recursos e investigar sobre la marcha sobre herramientas o conceptos nuevos, los cuales se fueron necesitando en el camino. Todo este proceso fue sumamente útil y de gran aprendizaje ya que la mayoría de las herramientas utilizadas para el desarrollo del proyecto eran nuevas y muchos de los conceptos se investigaron a profundidad.

5.2. Desarrollo

La Fase de desarrollo de la aplicación se realizó en cinco Sprints, los cuales tuvieron una duración en total de 17 semanas continuas. A continuación una descripción del trabajo realizado en cada uno.

5.2.1. Primer Sprint

Durante este Sprint se hizo el análisis y definición de los requerimientos de la aplicación, los cuales eran primero definir el alcance inicial de la aplicación (ver Figura 5.1 y Figura 5.2), definir la arquitectura a utilizar y por último la estructura de la base de datos. El pasante inició la familiarización de la versión actual de CPI junto con el dueño del producto para resolver las posibles dudas de la funcionalidad, o conceptos necesarios para entender la aplicación. Además se creó el repositorio de Git, el proyecto de Visual Studio y el sitio de Umbraco, y por último el Dueño del producto sugirió dos posibles plantillas de HTML que podrían ser utilizadas para las vistas de la aplicación y el pasante eligió la que era más adecuada visual y funcionalmente para la misma.

Actividades realizadas:

- Familiarización con la versión original de CPI. Esta actividad consistió principalmente en reuniones con el dueño del producto para aclarar dudas con respecto a los conceptos necesarios para entender la aplicación y sus funcionalidades.
- Elegir la arquitectura de la aplicación. El pasante junto con el Dueño del Producto decidieron utilizar el modelo de 4+1 Vistas de Philippe Kruchten [13], el cual propone que un sistema software se ha de documentar y mostrar con 4 vistas bien diferenciadas y que tienen que relacionar entre sí con una vista más, que es la denominada vista “+1”. Estas vistas son: vista lógica, vista de procesos, vista de despliegue, vista física y la vista “+1” vista de escenario.

- Definir los actores del sistema. En principio se definieron dos actores principales para la aplicación, *Admin* (el administrador del sistema) y *Retailer* (representantes de cadenas).
- Realizar una matriz de permisología de las funcionalidades de la aplicación. Lo que el pasante realizó fue listar todas las posibles funciones de cada uno de los módulos de aplicación y definir cuáles eran los permisos que tendrán cada uno de los actores.
- Elegir las entidades de la aplicación. El pasante basándose en la primera versión de la aplicación y junto el dueño del producto tomaron la decisión de cuáles serían las entidades que se manejarían desde la base de datos de Umbraco (cadenas) y cuáles en la base de datos de SQL Server (tiendas, zonas, productos, precios y categorías).
- Rediseñar la base de datos. Luego de decidir cuáles eran las entidades a trabajar en la base de datos de SQL Server el pasante junto con el Dueño del Producto decidieron cuál podría ser la nueva estructura de la base de datos. A continuación se muestra el Diagrama de la versión original de CPI (ver Figura 5.3) y el Diagrama ER de la versión 2 de CPI (ver Figura 5.4)
- Crear el proyecto en Visual Studio.
- Crear el repositorio de Git. Para el proyecto se creó un repositorio llamado CPI el cual se utilizó para el control de versiones del proyecto. Para utilizar correctamente el repositorio el pasante tuvo que refrescar los comandos.
- Crear el sitio de Umbraco.
- Elegir la plantilla a utilizar para las vistas de la aplicación. El pasante junto con el Dueño del Producto eligieron entre dos posibles plantillas, la elegida fue la que más se acopló a las necesidades de la aplicación.
- Investigar sobre Datatables ya que fue el pluggin (complemento) elegido por el Dueño del Producto para manejar dinámicamente las tablas.
- Listar las vistas que tendrá la aplicación, según los módulos que se eligieron como alcance inicial.

Duración: 2 semanas.

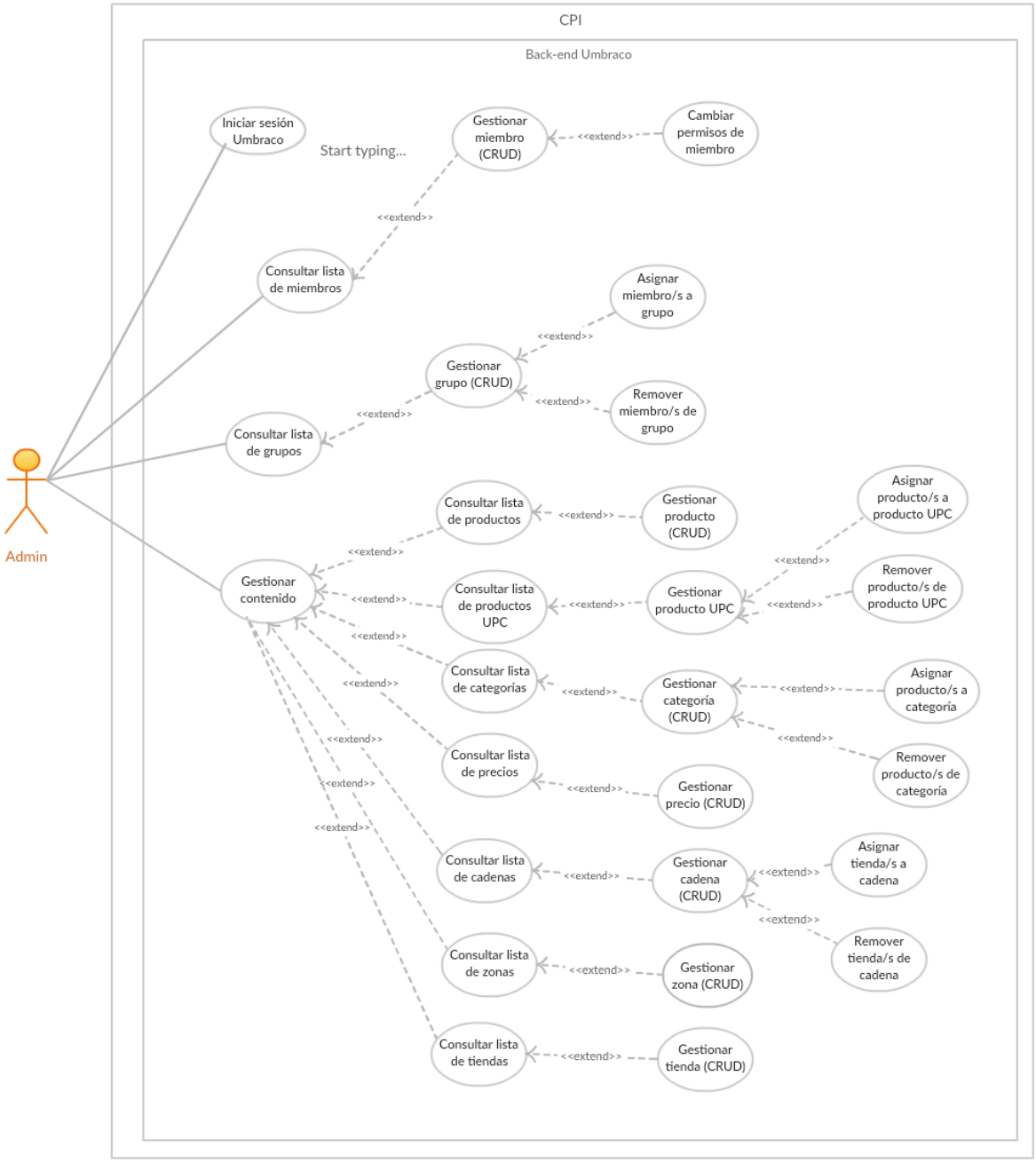

Figura 5.1: Diegrama Diagrama de Casos de Uso para Admin inicial. Elaboración propia.

Figura 5.2: Diagrama de Casos de Uso para Retailer inicial. Elaboración propia.

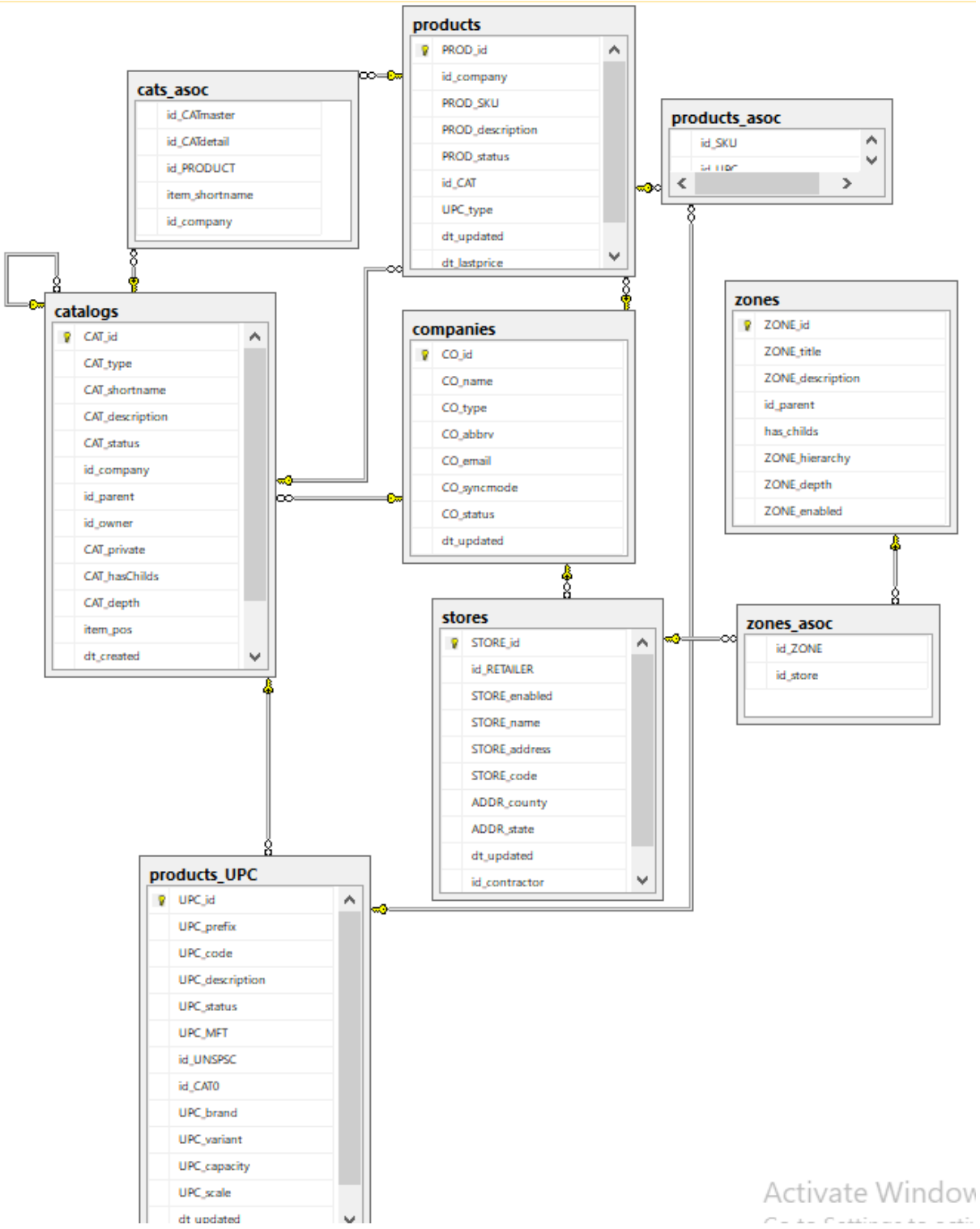

Figura 5.3: Diegrama ER de la base de datos de la versión 1 de CPI.

Figura 5.4: Diagrama de la base de datos de la nueva versión de CPI.

5.2.2. Segundo Sprint (Módulo de Administración de cadenas, productos y categorías)

Durante este Sprint se implementó la funcionalidad para el manejo de las principales entidades de la aplicación: cadenas, productos y categorías, en la cual se desarrolló una interfaz gráfica en el back end de Umbraco usando un paquete llamado Fluidity para

el manejo de estas entidades en la base de datos. Se definieron los Doctypes (tipos de documento) y Datatypes (tipos de datos) para crear el contenido en Umbraco. Se inició el desarrollo de las vistas de la aplicación. Y por último se empezó la implementación para la vista de listado de productos.

Actividades realizadas:

- Definir los Doctypes para las entidades principales. El pasante definió las propiedades que tendrían cada una de las entidades que se encuentran en Umbraco (ver en el Anexo A la Sección 8.2).
- Definir los Datatypes que usan las entidades principales. El pasante creo tipos de datos necesarios para las propiedades de las entidades a usarse en Umbraco (ver en el Anexo A la Sección 8.1).
- El Dueño del producto realizó una instalación local de la versión 1 de la aplicación CPI para que el pasante pudiera revisar a fondo la funcionalidad actual y además revisar el código fuente y la base de datos para ayudar al desarrollo de la nueva versión de la aplicación.
- El Dueño del Producto propuso dos paquetes de Umbraco para gestionar los datos de la base de datos de SQL Server (Fluidity o UI-Matic) y el pasante investigó sobre ambos y se decidió elegir Fluidity ya que era un paquete desarrollado más recientemente y además era compatible con la versión de Umbraco utilizada para el desarrollo de la aplicación. Luego se desarrolló la interfaz de Fluidity para creación, edición y eliminación de registros de las entidades que se guardan en la base de datos de SQL Server, para de esta manera poder manejar los registros desde el back end de Umbraco. Para poder realizar la interfaz y poder manejar correctamente los datos de las tablas de la base de datos, en algunos casos se tuvo que agregar un campo adicional a la tabla y el mismo se tomaba como clave primaria, ya que estas tablas tenían dos claves primarias y Fluidity no manejaba correctamente este tipo de casos.
- Se desarrollaron las principales partials views (vistas parciales) que tienen en común todas las vistas de la aplicación, como por ejemplo la barra de navegación y la barra lateral.
- Se realizó la implementación de la vista y la funcionalidad para listar los productos por cadena, y esto se hizo utilizando Datatables.

Duración: 3 semanas.

5.2.3. Tercer Sprint (Módulo de Generación de Reportes)

En este Sprint se desarrolló parte de la funcionalidad de generación de reportes comparativos. Este módulo cuenta con tres tipos de reportes, sin embargo únicamente se pudieron desarrollar dos de ellos (reporte de comparación de precios de un producto en distintas cadenas, y reporte histórico de precios de un producto) ya que el tiempo de desarrollo de los gráficos usando la herramienta Highcharts tomó más tiempo del que se tenía planificado porque al ser una herramienta que el pasante no utilizó anteriormente se necesitó de tiempo para aprender a utilizarla y luego para poder traer los datos correctamente de la base de datos (los cuales son los que se ven reflejados en cada uno de los gráficos) se necesitó de la traducción, entendimiento y mejora considerable del código de la versión antigua y esto se hizo bastante engorroso ya que los scripts no estaban bien documentados.

Actividades realizadas:

- Elección de la herramienta para hacer los gráficos necesarios para los reportes. El Dueño del Producto propuso dos herramientas para la generación de los gráficos, la primera era Chartjs que es de licencia gratuita y la segunda se llama Highcharts la cual es de licencia paga. El pasante para realizar la elección de la herramienta, realizó ejemplos de gráficos usando ambas opciones para comparar complejidad, facilidad de conseguir recursos y ejemplos para ayudar con el desarrollo, entre otras cosas, y llegó a la conclusión que la herramienta más apta para ser usada en la aplicación era Highcharts ya que contaba con todos los tipos de gráficos necesarios para la aplicación (Chartjs no contaba con gráfico boxplot) y cuenta con mayor documentación y ejemplos en los que se podía apoyar para el desarrollo de los gráficos.

- Revisión de la documentación y la Referencia API de JavaScript de Highcharts para poder comenzar con el desarrollo de los gráficos para los reportes.

- Implementación del reporte de comparación de precios de un producto en común entre distintas cadenas. Este reporte se divide en dos partes, la primera es un gráfico de barras en el que se ven reflejados el precio normal y el precio por oferta del producto en las distintas cadenas, y la segunda parte es una tabla con información más detallada del producto (precio PVP, precio oferta, IVA del producto y número de tiendas dentro de la cadena en la que se encuentra disponible dicho producto).

- Implementación del reporte histórico de un producto en un período de tiempo, para esto se realizó un gráfico boxplot en el que se muestra una caja por fecha en el que

esté registrado el precio de ese producto dentro del período elegido.

Duración: 4 semanas.

5.2.4. Cuarto Sprint (Continuación con el Módulo de Generación de Reportes)

En este Sprint se realizó el reporte de precios de productos por categorías (gráfico de dispersión) que faltó por realizar en el Sprint anterior, y además se corrigieron detalles de los otros dos reportes ya implementados.

Actividades realizadas:

- Se agregó a la interfaz de Fluidity que ya se encontraba implementada la función para asignar productos a categorías.
- Agregar enlace de la tabla de productos al reporte de comparación de precios. A la lista de todos los productos se le agregó un enlace desde el código SKU del producto hacia el reporte de comparación de precios de productos de diferentes cadenas.
- Implementación del reporte de precios de productos por categorías. Primero para comenzar con la implementación se realizó un boceto con datos ficticios del gráfico de dispersión para que el pasante se familiarizara con la complejidad de este tipo de gráficos. Luego se realizó la traducción, mejora y entendimiento del código de la versión antigua de la aplicación para manejar los datos que se encuentran en la base de datos y reflejarlos en el gráfico.
- Se inició la construcción del diccionario de Umbraco con las palabras estáticas y que se repiten en cada una de las vistas de la aplicación.
- Se corrigieron detalles de los reportes anteriores. Para el reporte de comparación de precios de un producto se corrigió el formato de la fecha (pasó de utilizarse “mm/dd/yyyy” a el formato ISO “yyyymmdd”) de igual manera para el reporte histórico, se arreglaron detalles del código de HTML y eficiencia de los controladores para traer la información de la base de datos.

Duración: 4 semanas.

5.2.5. Quinto Sprint (Módulo de Administración de Precios y Corrección de estructura del proyecto)

Este es el último Sprint de desarrollo, en el mismo se implementó el módulo de administración de precios de productos. El alcance inicial de este módulo era realizar la interfaz gráfica para la creación, edición y eliminación de registros, importar precios por lote mediante un archivo en formato Microsoft Excel e importar mediante webservice desde una fuente externa en formato JSON, sin embargo la última funcionalidad no se pudo desarrollar ya que durante el sprint el pasante se dió cuenta de que había un error en la estructura, ya que la misma no estaba siguiendo el patrón MVC que se debía tener en la solución de Visual Studio. Este error en la estructura fue debido a la falta de comunicación entre el pasante y el Dueño del Producto sobre aspectos técnicos del proyecto, y para corregir este inconveniente se tomó tiempo del Sprint por lo que no alcanzó para completar el módulo y se decidió dejar esta funcionalidad para más adelante.

Actividades realizadas:

- Se implementó la autenticación, inicio y cierre de sesión para el front end de la aplicación. Para esto primero se definen los permisos de usuarios, en este caso, para el alcance de la aplicación únicamente se tiene el usuario *retailer*, y se creó un grupo de miembro en el back end de Umbraco en la sección Members llamado *cpi user*, luego se desarrolló la vista para el iniciar y cerrar sesión.

- Se desarrolló la interfaz gráfica para creación, edición y eliminación de registros usando el paquete de Umbraco Fluidity para poder manejar esta entidad directo de la plataforma, modificando directamente la base de datos de SQL Server.

- Se realizó la implementación para importar por lote los precios desde un archivo en formato Microsoft Excel.

- El código de la aplicación paso a revisión y de esta manera encontrar detalles de escritura y aplicar buenas prácticas de desarrollo. Por lo que se mejoraron nombres y declaraciones de variables, se agregaron mas comentarios para el entendimiento del código, y eliminación de librerías que no se usaban por las clases.

- Reorganización de controladores y vistas de la aplicación. Luego de revisar y corregir los errores encontrados, se consiguió un error en la estructura de la solución de Visual Studio, ya que no se estaba siguiendo el patrón MVC que se había escogido al prin-

cipio de la pasantía. Por lo que para resolver esto, se organizaron los controladores y las vistas de la aplicación.

Duración: 4 semanas.

A continuación se muestran los Diagramas de casos de Uso desarrollados al final de esta fase.

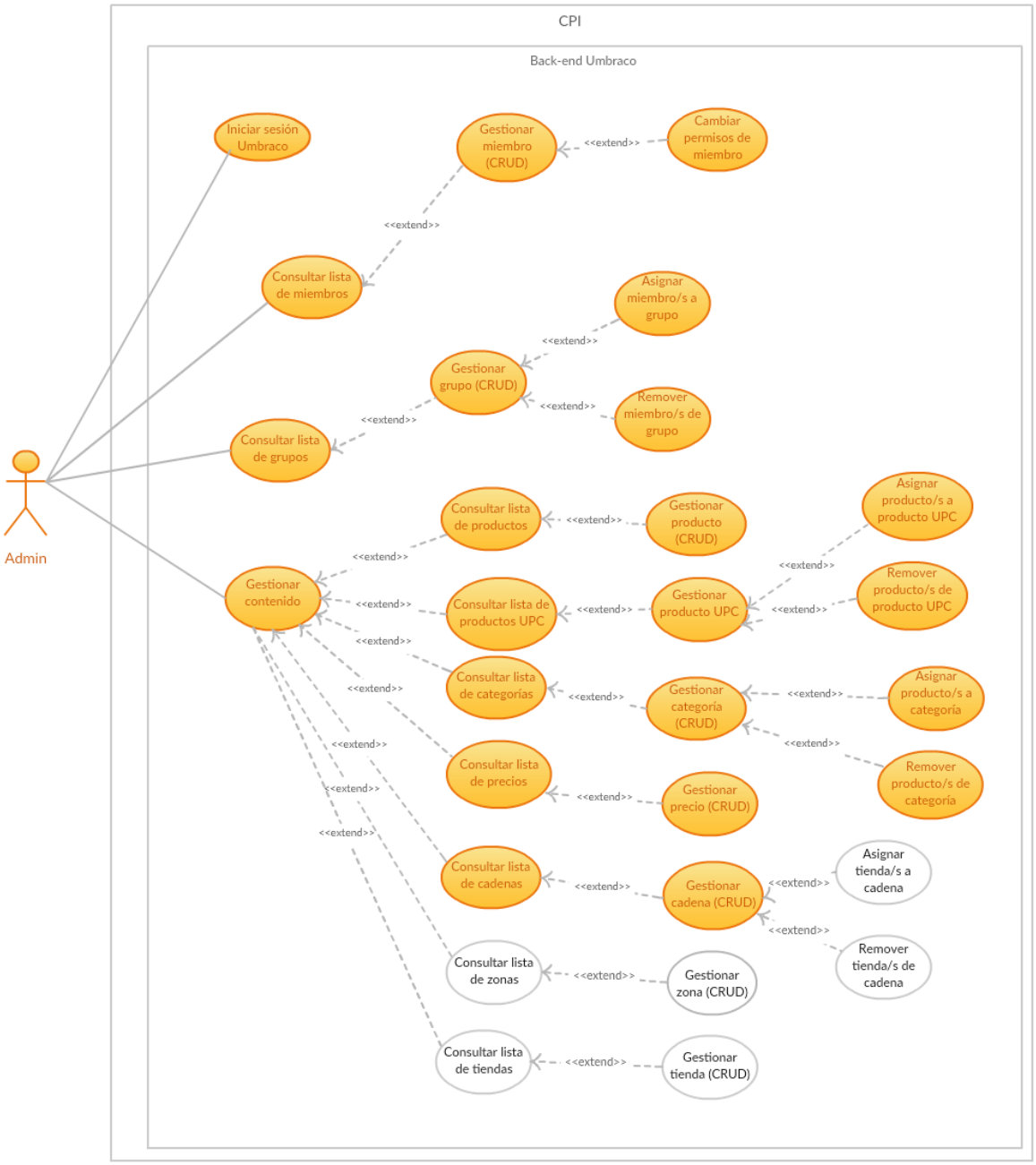

Figura 5.5: Diagrama de Casos de Uso desarrollados para el actor Admin. Elaboración propia.

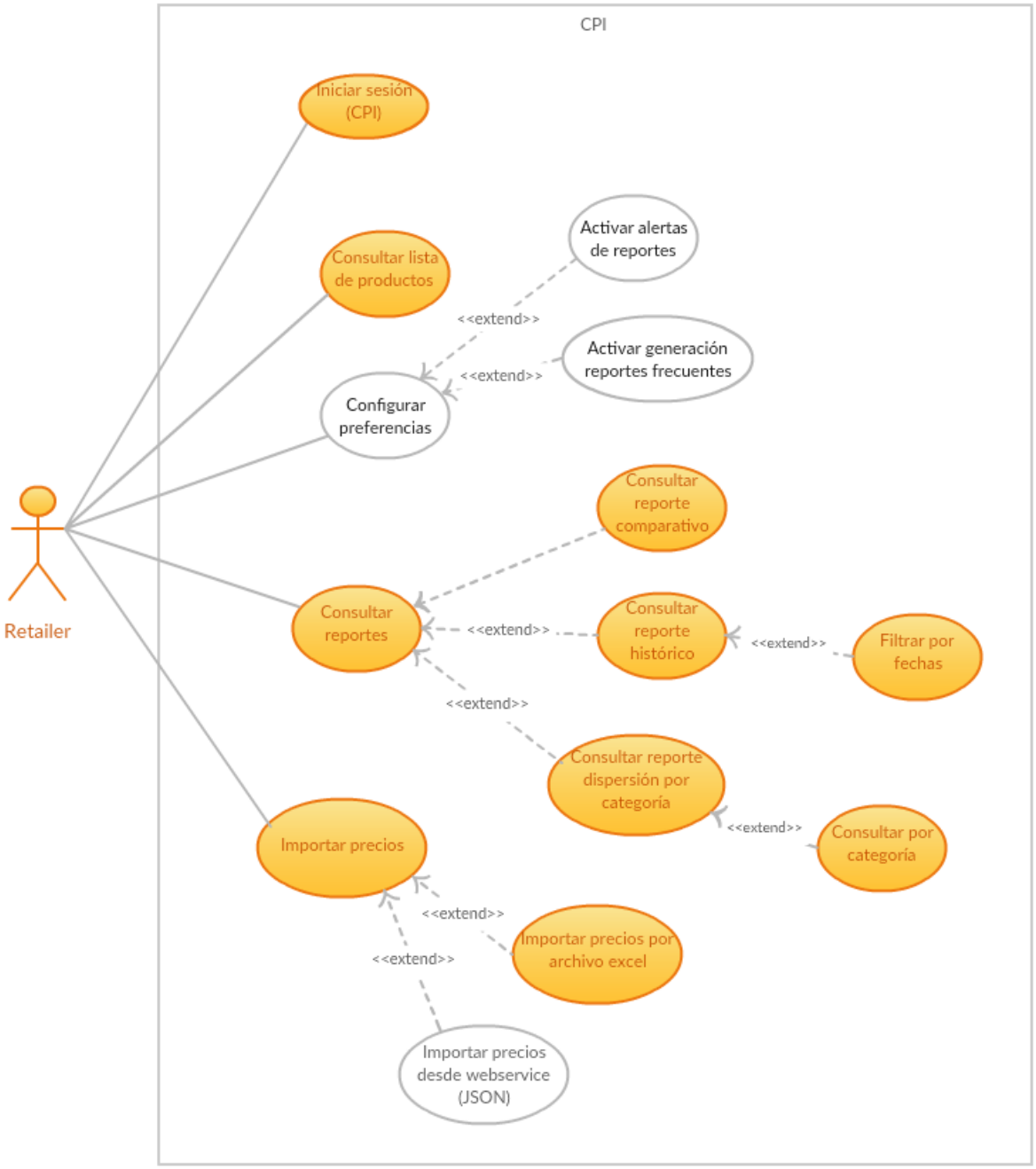

Figura 5.6: Diagrama de Casos de Uso desarrollados para el actor Retailer. Elaboración propia.

5.3. Documentación

Esta última fase del proyecto duró las últimas dos semanas del período de pasantía. Se realizó la documentación necesaria para la aplicación, se dejó expresado el estado final del sistema, se elaboraron los documentos de instalación de CPI y el Documento de Arquitectura de Software (ver Anexo A) donde se detallan los componentes y casos de uso desarrollados de la aplicación.

CONCLUSIONES Y RECOMENDACIONES

Durante este proyecto de pasantía se desarrolló una nueva versión de la aplicación web CPI que permite realizar comparaciones de precios entre productos ofrecidos por diferentes cadenas de ventas al detal. El proceso de desarrollo requirió realizar un proceso completo de re-ingeniería para adaptar la versión antigua a plataformas y modelos de gestión más modernos.

- El módulo de administración de cadenas, productos y categorías permite al usuario manejar a su gusto cada una de estas entidades, además de listar todos los productos disponibles en la cadena al que pertenece el mismo.
- El módulo de generación de reportes permite al usuario de la cadena realizar comparaciones de los productos de las otras cadenas, con el fin de poder confeccionar una estructura de precios que resulte simultáneamente atractiva para atraer a sus clientes así como rentable para mantener el negocio a flote.
- El módulo de administración de precios permite al usuario manejar a su gusto los precios de los productos pertenecientes a esta cadena.
- El uso de una aplicación Web brinda un servicio rápido de respuesta y accesibilidad desde cualquier parte de la internet. Además ofrece una interfaz amigable y fácil de usar.
- En el desarrollo se presentaron retrasos primero a pesar de hubo una fase de inducción del pasante a los conceptos y herramientas básicas necesarias para la implementación de la aplicación, una semana no fue suficiente para prepararlo para abarcar y cumplir con el alcance inicial del proyecto, ya que a medida de que se fue avanzando en el desarrollo de los módulos se tuvo la necesidad de realizar investigaciones o tutoriales para completar el conocimiento que hacía falta para completar los mismos, lo que muchas veces hizo que se retrasara el trabajo. Segundo a pesar que el pasante trató de tener al día los conocimientos para cumplir con el plan de trabajo, se tuvo una equivocación en la estructura de la solución de Visual Studio por lo que se tuvo que

tomar tiempo que estaba planificado para el desarrollo de otras funcionalidades para resolver este tipo de inconvenientes.

- Hay que tener presente que para el desarrollo de cualquier sistema es de gran importancia llevar al día la documentación del mismo, ya que a través de estos documentos los administradores o próximos desarrolladores pueden realizar un correcto mantenimiento y agregar módulos de una manera fácil y cualquier persona que necesite hacer uso del sistema lo pueda manejar sin ningún inconveniente.

-

- Finalmente, se puede concluir que gracias al presente trabajo de pasantía el pasante obtuvo una gran cantidad de enseñanzas y crecimiento como futuro profesional que complementaron y reforzaron los conocimientos logrados en la Universidad. Fue una oportunidad inigualable para poner en práctica una gran cantidad de conceptos y teorías aprendidas durante la estadía de la misa, así como para aprender que la práctica es muy distinta a la teoría.

5.4. Recomendaciones

A continuación se desglosan las recomendaciones sobre aquellos aspectos que podrían mejorarse:

- Se recomienda finalizar la implementación de las funcionalidades de los módulos que no se alcanzaron a desarrollar durante la pasantía. Y a su vez empezar el desarrollo de los módulos que no eran parte del alcance inicial con el fin de tener la nueva versión de CPI completa para lanzarla a producción sustituyendo a la antigua versión.

- Es recomendable realizar la documentación del sistema al mismo momento que se va avanzando con su desarrollo, a manera de realizarla con calidad, completitud y correctitud.

- Se recomienda a los desarrolladores del sistema mantener un continuo seguimiento de los avances del proyecto junto con el Dueño del Producto para verificar que todos los requerimientos del mismo están siendo cumplidos satisfactoriamente.

REFERENCIAS

[1] IKêls. (2018) Que hacemos. [Online]. Available: http://www.ikels.com/que-hacemos/

[2] D. Barker, *Web Content Management: Systems, Features, and Best Practices*. O'Reilly Media, 2016.

[3] Microsoft. Asp.net overview. [Online]. Available: https://msdn.microsoft.com/en-us/library/4w3ex9c2.aspx

[4] ——. (2015) Introduction to the c# language and the .net framework. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework

[5] Techopedia. Software as a service. [Online]. Available: https://www.techopedia.com/definition/155/software-as-a-service-saas

[6] Umbraco. (2018) The flexible cms. [Online]. Available: https://umbraco.com/tour

[7] G. E. Krasner and S. T. Pope, "A cookbook for using the model-view-controller user interface paradigm in smalltalk-80," *Journal of Object-Oriented Programming*, vol. 1, pp. 26–49, 1988.

[8] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*. Pearson, 2010.

[9] Microsoft. .net. [Online]. Available: https://www.microsoft.com/net/learn/dotnet/what-is-dotnet

[10] P. Christensson. (2016) Api definition. [Online]. Available: https://techterms.com/definition/api

[11] ——. (2017) Web service definition. [Online]. Available: https://techterms.com/definition/web_service

[12] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, 2000.

[13] P. Kruchten, "The 4+1 view model of architecture," *IEEE Software 12(6):45-50*, pp. 42–50, nov 1995.

[14] S. Moon, T. Leithead, A. Eicholz, A. Danilo, and S. Faulkner, "HTML 5.2," W3C, W3C Recommendation, Dec. 2017, https://www.w3.org/TR/2017/REC-html52-20171214/.

[15] H. W. Lie, C. Lilley, I. Jacobs, and B. Bos, "Cascading style sheets, level 2 (css2 specification)," W3C, W3C Recommendation, Apr. 2008, http://www.w3.org/TR/2008/REC-CSS2-20080411/.

[16] P. Christensson. (2014) Javascript definition. [Online]. Available: https://techterms.com/definition/javascript

[17] W3Schools. What is razor? [Online]. Available: https://www.w3schools.com/asp/razor_intro.asp

[18] Microsoft. (2010) Asp.net mvc 3. [Online]. Available: https://docs.microsoft.com/en-us/aspnet/mvc/mvc3

[19] ——. (2010) Asp.net web api. [Online]. Available: https://msdn.microsoft.com/en-us/library/hh833994(v=vs.108).aspx

[20] Git. (2018) git. [Online]. Available: https://git-scm.com/

[21] Microsoft. (2018) Sql server documentation. [Online]. Available: https://docs.microsoft.com/en-us/sql/sql-server/sql-server-technical-documentation?view=sql-server-2017

[22] ——. (2018) Welcome to visual studio ide. [Online]. Available: https://docs.microsoft.com/en-us/visualstudio/ide/visual-studio-ide?view=vs-2017

[23] "The json data interchange syntax," Dec. 2017. [Online]. Available: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

[24] P. Christensson. (2011) Ajax definition. [Online]. Available: https://techterms.com/definition/ajax

[25] HighSoft. Highcharts js. [Online]. Available: https://www.highcharts.com/about/

[26] DataTables. "data tables manual". [Online]. Available: https://datatables.net/manual/

[27] K. Schwaber and J. Sutherland. (2017) The scrum guide. [Online]. Available: https://www.scrumguides.org/scrum-guide.html

APÉNDICE A

DOCUMENTO DE ARQUITECTURA DE SOFTWARE

Documento de Arquitectura del Software

CPI

Versión: 1.0

Índice

Fecha	Versión	Descripción	Autores
29/09/2018	1.0	Documento de Arquitectura	Valentina Hernández

1 Introducción

1.1 Propósito

Este documento permitirá a los involucrados en el desarrollo de este sistema, tener una visión general de su arquitectura. Sirviendo como guía para el desarrollo, la identificación de las funcionalidades al dar una visión detallada de ellas, así como para dar una base informativa para el mantenimiento del sistema por parte del equipo de trabajo original o de terceros.

1.2 Alcance

A continuación se presenta una abstracción de la estructura que debe tener el sistema, el presente documento servirá de guía para los desarrolladores, usuarios y stakeholders para conocer los requerimientos del sistema, el diseño de la base de datos, así como mostrar sus funcionalidades y como están implementadas en el sistema.

Esta información es importante para dar una explicación clara de cómo está diseñado el sistema y así facilitar la mantenibilidad del mismo.

1.3 Definiciones, Siglas y Abreviaciones

1.4 Referencias

No hace referencia a ningún otro documento.

1.5 Vista Global

Este documento contiene todas las especificaciones de la arquitectura de la aplicación web *CPI* tanto a nivel de hardware como de software. Se introduce la representación arquitectónica del sistema, especificando en detalle los objetivos y restricciones de la misma, los casos de uso, procesos dentro del sistema, la implantación, implementación, los datos almacenados, tamaño, desempeño y calidad. También se explicará datos importantes del sistema mediante diagramas de actividades, de casos de uso, de clases, diagrama ERE, etc.

2 Representación Arquitectónica

La representación arquitectónica de *CPI* está basada en el modelo de 4+1 vistas de Philippe Kruchten. En el transcurso del documento se tratarán más a fondo los detalles de cada una.

3 Vista Lógica

La Vista Lógica muestra la estructura estática del sistema. En esta vista se representa la funcionalidad que el sistema proporcionará a los usuarios finales, es decir, se representa lo que el sistema debe hacer y las funciones y servicios que ofrece. Para esta vista se desarrollaron el Modelo Conceptual (ver Figura 1) y el Modelo Entidad-Relación de la Base de Datos de la aplicación (ver Figura 2).

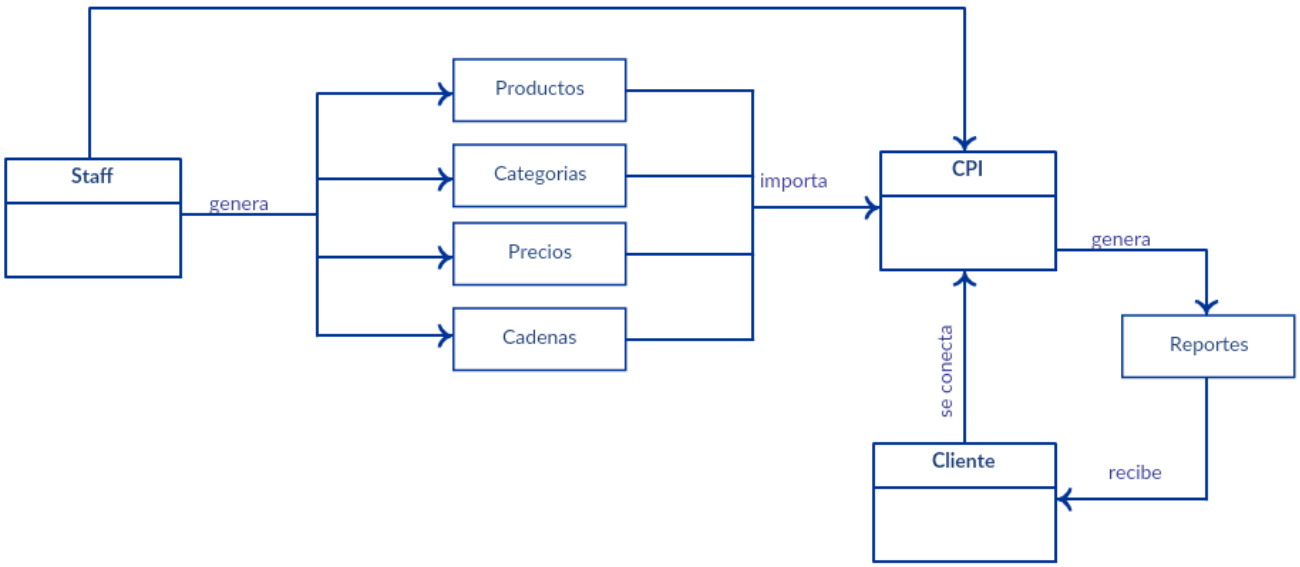

Figure 1: Modelo de Dominio. Elaboración propia.

Figure 2: Diagrama ER

4 Vista de Desarrollo

La Vista de Desarrollo de la aplicación se hizo a través de un diagrama de componentes, en donde se especifica la relación entre los mismos. Se tomaron en cuenta los componentes principales de la aplicación para facilitar al lector la comprensión del mismo. A continuación se muestra el Diagrama de componentes en la Figura 3.

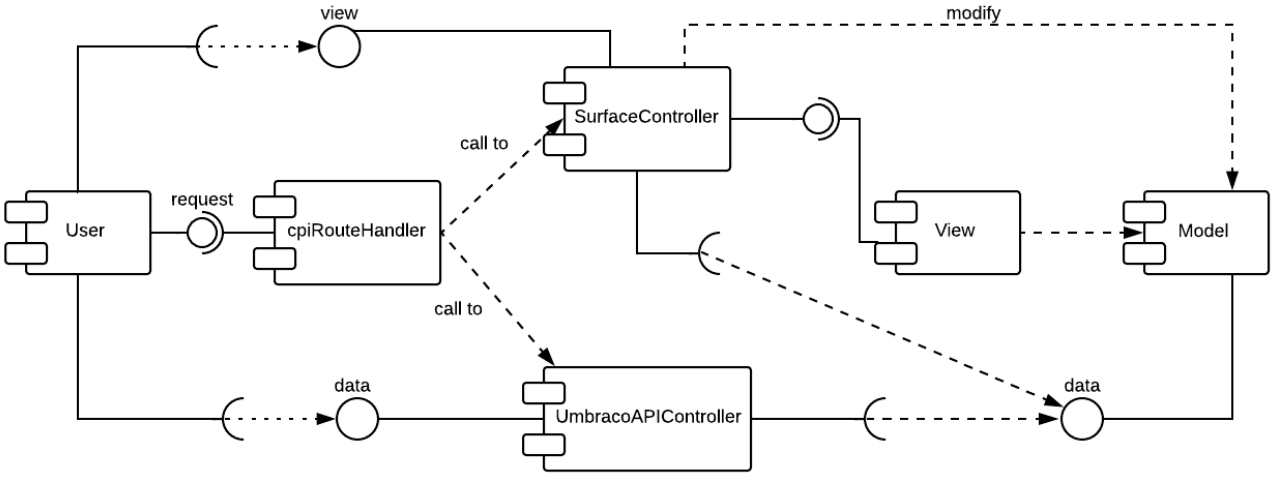

Figure 3: Diagrama de Componentes. Elaboración propia.

5 Vista de Procesos

En esta vista se muestran tanto los aspectos concurrentes del sistema a tiempo de ejecución como las interacciones que ocurren entre ellos. Sin embargo esta vista no fue desarrollada dado que la aplicación no resuelve aspectos de concurrencia de usuarios ni de datos.

6 Vista Física

La Vista Física muestra las plataformas de hardware y software necesarias para la ejecución del sistema. Esta vista se ve reflejada en la Figura 4.

En el diagrama se puede apreciar la disposición física de los componentes del software, partiendo desde el equipo del usuario representante de la cadena, los usuarios ingresan al Sistema por medio de un navegador Web en cualquier equipo de cómputo que tenga acceso a Internet. Dicho Sistema se encuentra alojado en un servidor web. Para cada petición de datos se accede al servidor de base de datos, el cual contiene la base datos de la aplicación.

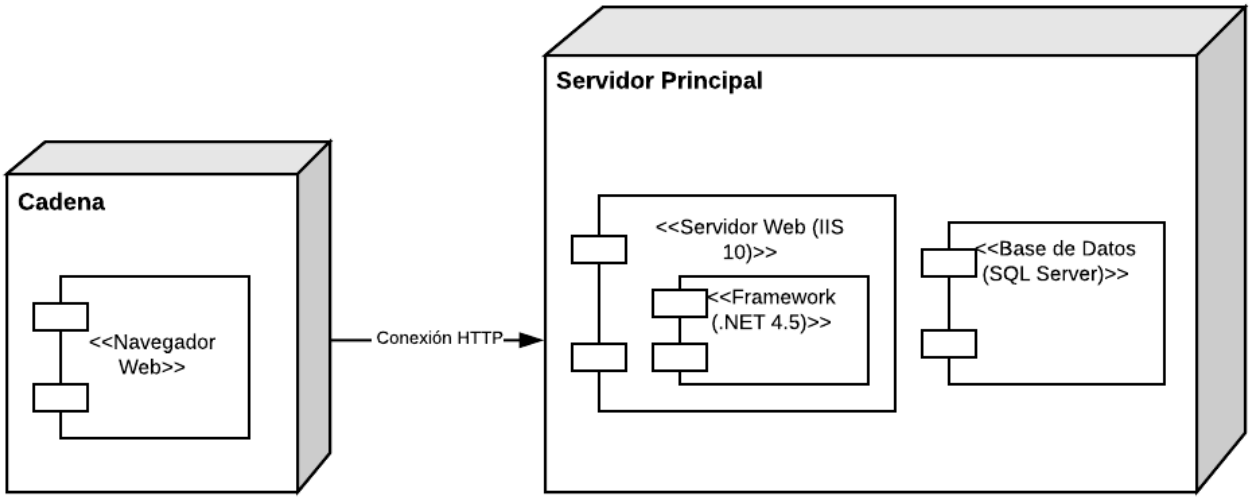

Figure 4: Diagrama de Despliegue. Elaboración propia.

7 Vista de Casos de Uso

En esta vista se describirá el sistema desde el punto de vista de los casos de uso. El sistema tiene 2 actores:

- *Admin*: Administrador de CPI. Es un usuario de Umbraco, lo que quiere decir que cuenta con las credenciales para poder ingresar al back-end de Umbraco y tiene los permisos necesarios para administrar las entidades y los grupos de CPI.
- *cpi-user*: Es un grupo de CPI, los pertenecientes a este grupo poseen credenciales para entrar al sistema.
- *Retailer*: Representa a una cadena. Solo tiene acceso a la información referente a las tiendas y productos que pertenecen a la cadena.

7.1 Resumen de Casos de Uso

ID del Caso de Uso	Caso de Uso	Actor
CU-1	Iniciar sesión (Umbraco)	Admin
CU-2	Consultar lista miembros	Admin
CU-3	Gestionar miembro (CRUD)	Admin
CU-4	Consultar grupos	Admin
CU-5	Gestionar grupo (CRUD)	Admin
CU-6	Asignar miembro/s a grupo	Admin
CU-7	Remover miembro/s de grupo (CRUD)	Admin
CU-8	Cambiar permisos de miembro	Admin
CU-9	Gestionar contenido	Admin
CU-10	Consultar lista de cadenas	Admin
CU-11	Gestionar cadena (CRUD)	Admin
CU-12	Consultar lista de productos	Admin
CU-13	Gestionar producto (CRUD)	Admin
CU-14	Consultar lista de productos UPC	Admin
CU-15	Gestionar producto UPC (CRUD)	Admin
CU-16	Asignar producto/s a producto UPC	Admin
CU-17	Remover producto/s de producto UPC	Admin
CU-18	Consultar lista de precios	Admin
CU-19	Consultar lista de tiendas	Admin
CU-20	Gestionar tienda (CRUD)	Admin
CU-21	Consultar lista de zonas	Admin
CU-22	Gestionar zonas (CRUD)	Admin
CU-23	Consultar lista de catálogo generales	Admin
CU-24	Gestionar catálogo (CRUD)	Admin
CU-25	Asignar producto/s a catálogo	Admin
CU-26	Remover producto/s de catálogo	Admin
CU-27	Iniciar sesión (CPI)	Retailer
CU-28	Consultar lista de productos	Retailer
CU-29	Importar precios desde archivo excel	Retailer
CU-30	Consultar precio producto	Retailer
CU-31	Consultar histórico de producto	Retailer
CU-32	Consultar gráfico dispersión por categoría	Retailer
CU-33	Filtrar gráfico dispersión por categoría	Retailer
CU-34	Consultar lista de catálogos personalizados	Retailer
CU-35	Gestionar catálogo (CRUD)	Retailer
CU-36	Asignar producto/s a catálogo	Retailer
CU-37	Remover producto/s de catálogo	Retailer

7.2 Diagrama de Casos de Uso

Se separaron los casos de uso en varios diagramas para facilitar la lectura.

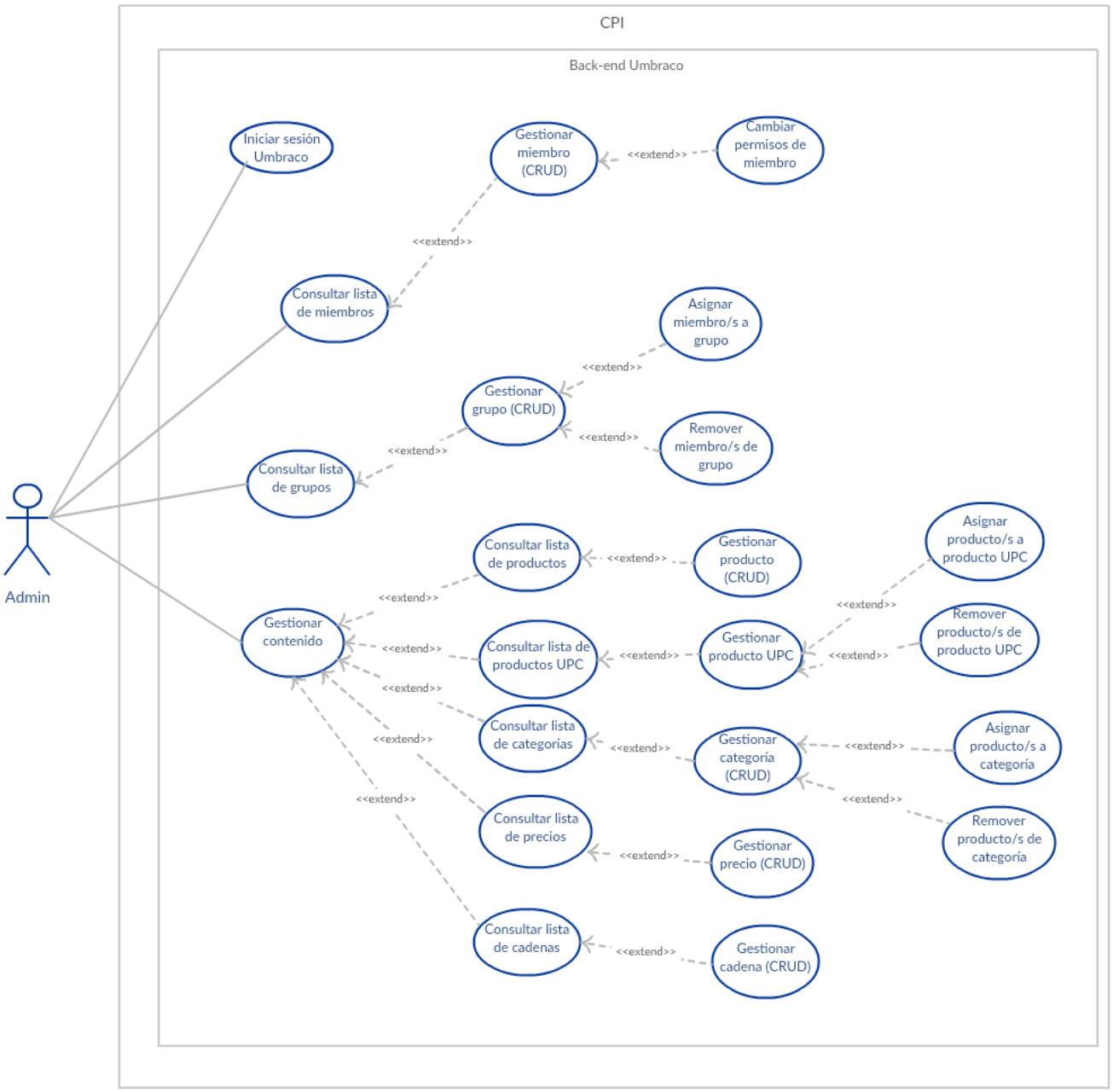

Figure 5: Casos de uso para usuario Admin

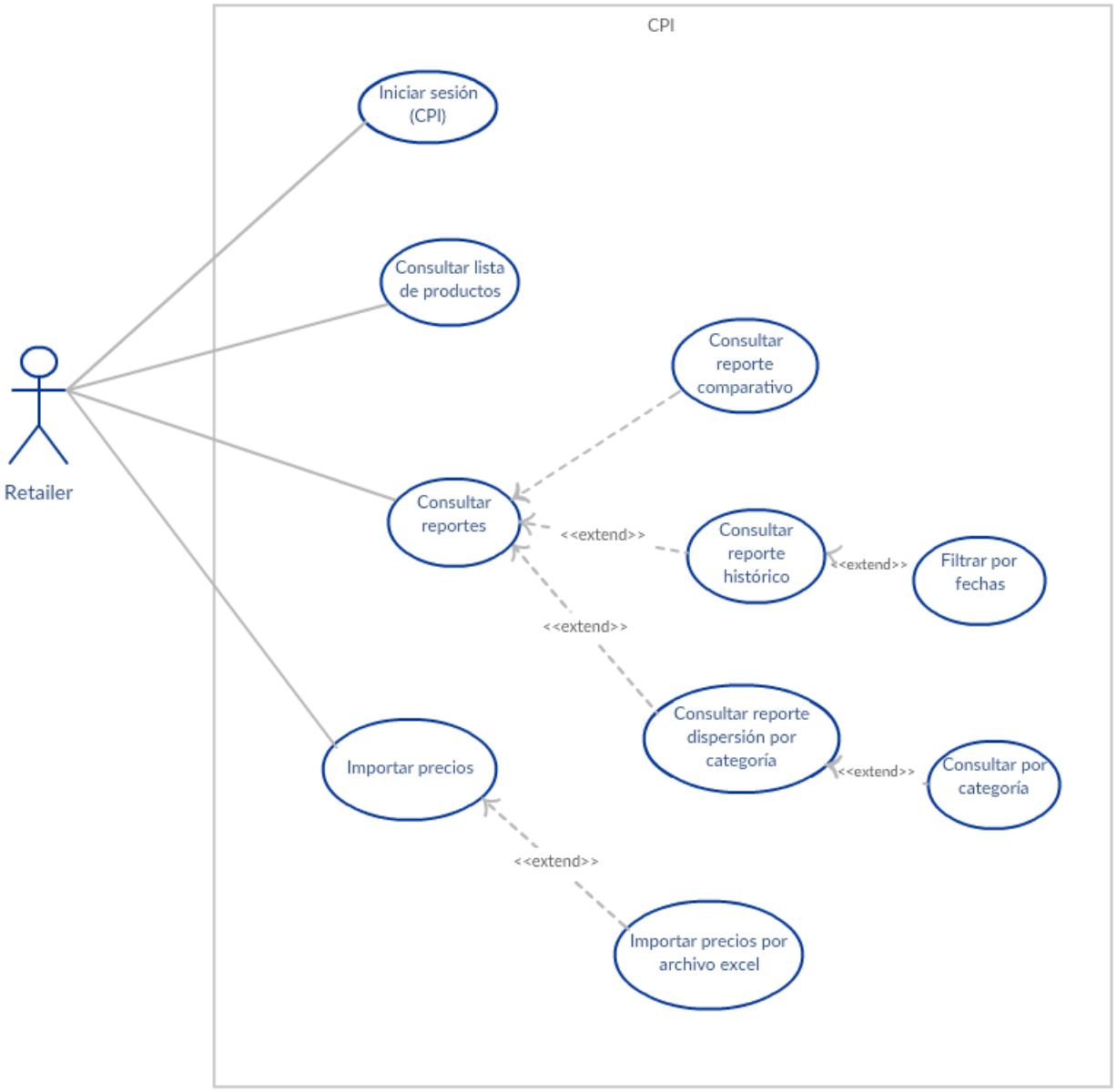

Figure 6: Casos de uso para usuario Retailer

7.3 Especificaciones de Casos de Uso

A continuación las narrativas de los casos de uso:

Caso de Uso: Iniciar sesión (Umbraco)	
Descripción: El usuario quiere ingresar al back end de Umbraco.	
Precondición: Ingresar la dirección correcta del sitio en la barra de navegación.	
Flujo básico:	
Actor	Sistema
1. El actor abre su navegador e introduce la dirección correspondiente al back end de Umbraco.	
	2. El servidor procesa la solicitud y envía al navegador del cliente una ventana para que el usuario se autentique.
3. El actor introduce su email y su contraseña.	
	4. El sistema valida la información del usuario y lo redirige al tablero (back end) de Umbraco.
Flujos alternos:	
Actor	Sistema
1. El actor abre su navegador e introduce la dirección correspondiente al back end de Umbraco.	
	2. El servidor procesa la solicitud y envía al navegador del cliente una ventana para que el usuario se autentique.
3. El actor introduce su email y su contraseña.	
	4. El sistema no valida la información del usuario y lo redirige a la misma página de inicio de sesión indicándole que los datos introducidos son incorrectos.
Poscondición: El usuario se encuentra en el tablero de Umbraco.	
Puntos de extensión: No se requiere de otros casos de uso.	

Caso de Uso: Consultar lista de miembros	
Descripción: El usuario quiere consultar la lista de miembros de CPI.	
Precondición: Haber ingresado al back end de Umbraco.	
Flujo básico:	
Actor	Sistema
1. El usuario le da click a la sección "Members" en el tablero de Umbraco.	
	2. El sistema muestra el panel de miembros.
3. El usuario le da click a la carpeta "Members".	
	4. El sistema muestra la lista de los miembros de CPI.

Flujos alternos:	
Actor	Sistema
1. El actor hace algo.	
	2. El sistema responde.
Poscondición: Aquí va la poscondición del caso de uso.	
Puntos de extensión: Aquí van los puntos de extensión del caso de uso.	

8 Componentes de Umbraco

En esta sección se muestran los componentes desarrollados para la parte Umbraco del sistema.

8.1 Datatypes

A continuación se muestra una lista de los Datatypes usados por la aplicación. Nota: Todos los Datatypes creados para CPI tienen el prefijo *_cpi::* en su nombre.

Datatype	**Función**
Attached images	Seleccionar una imagen
Cadena	Seleccionar una cadena
Categoria	Seleccionar una categoría
Email address	Introducir un correo electrónico
Fechas	Seleccionar una fecha
On/off	Elegir una opción o no
Productos	Seleccionar un producto
Textarea max150	Introducir texto de máximo 150 caracteres
Textarea max65	Introducir texto de máximo 65 caracteres
UPC	Seleccionar código UPC de un producto

Table 4: Datatypes

8.2 Doctypes

Doctype	Propiedades	Datatype
Cadena	Nombre	Textarea max150
	Abreviación	Textarea max65
	ID	Numeric
	Correo electrónico	Email Address
	Estado	On/off
	Fecha actualización	Date Picker

Table 5: Doctypes