UNIVERSIDAD SIMÓN BOLÍVAR

DECANATO DE ESTUDIOS PROFESIONALES

COORDINACIÓN DE INGENIERÍA DE LA COMPUTACIÓN

DESARROLLO DE LA VERSIÓN 2 DE LA APLICACIÓN WEB EFUEL

Por:

Ricardo Münch

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

R. MÜNCH 2018	DESARROLLO DE LA VERSIÓN 2 DE LA APLICACIÓN WEB EFUEL	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 EFUEL

Por:

Ricardo Münch

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 describe a la empresa en la cual se desarrolló el proyecto de pasantía. 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. Definiremos el tipo de sistema sobre el cual se desarrolló la aplicación, los patrones de diseño usados para la construcción de la aplicación y algunos conceptos útiles para entender algunos componentes del sistema desarrollado.

2.1. Bases Teóricas

2.1.1. CMS

Un CMS, por sus siglas en inglés *Content Management System* (Sistema de Gestión de Contenidos), es una aplicación de software que provee algún nivel de automatización a las tareas de manejo de contenido. Un CMS permite a los usuarios crear nuevo contenido, editar contenido existente, y hacer el contenido accesible al público. [1]

Desde el punto de vista de un editor un CMS consta, básicamente, de 2 partes: una interfaz para la edición de contenido (referido como el *back-end*, esto es, la capa de acceso a los datos de la aplicación) y una interfaz para mostrar el contenido publicado (referido como el *front-end*, es decir, la capa de presentación de la aplicación).

El uso de un CMS facilita las tareas de mantenimiento y de generación de contenido, especialmente para usuarios que no tienen preparación técnica especial. Los usuarios de la aplicación eFuel serán, en su mayoría, personal sin conocimientos especializados en el área de computación, ésta es una de las razones por las cuales resulta conveniente desarrollar el sistema sobre un CMS.

2.1.2. API

Un API, siglas en inglés para *Application Programming Interface*, es un conjunto de comandos, funciones, protocolos y objetos que exponen los datos de una aplicación de software, es decir, establecen las reglas y los mecanismos a través de los cuales se puede tener acceso a ellos. [4]

Normalmente, aplicaciones externas disponen del API de una aplicación para obtener datos de esta última y usarlos para proveer algún servicio a sus usuarios. Para el proyecto presente se desarrolló un API para acceder a algunos de los datos de la misma.

2.1.3. Servicio Web

Un servicio web es un sistema de software diseñado para soportar interacción máquina-máquina a través de una red, proveen una vía estándar para la interoperabilidad entre distintas aplicaciones de software ejecutadas en distintas plataformas y ambientes. [2] Típicamente, los sistemas externos tienen acceso al servicio web a través de un API.

El proyecto presente incluye el desarrollo de un módulo de servicios web para el acceso a algunos datos de la aplicación, por ejemplo, datos para generar tablas informativas.

2.1.4. Arquitectura

A continuación se describen los modelos de arquitectura (física y de software) en los cuales se bassa el funcionamiento y el desarrollo de la aplicación.

2.1.4.1. Modelo Cliente-Servidor

Arquitectura de redes de computadoras que divide el trabajo entre 2 entidades: un cliente y un servidor. El cliente le envía una solicitud al servidor y espera una respuesta. Por otra parte, el servidor recibe la solicitud, lleva a cabo el trabajo requerido y devuelve una respuesta al cliente. [16] El servidor mantiene una relación de uno-a-muchos con los clientes. 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 y el proyecto presente, siendo una aplicación web, usa este patrón, por lo que es importante tenerlo en mente para entender (y desarrollar) algunas de las funcionalidades de la aplicación.

2.1.4.2. MVC

MVC, siglas para Modelo-Vista-Controlador, es un modelo de diseño de aplicaciones compuesto por 3 partes: el Modelo (datos), la Vista (presentación de los datos e interfaz con el usuario) y el Controlador (proceso que maneja la entrada de los usuarios y el acceso a los datos). [7] Al separar la aplicación en estos 3 componentes se reduce la complejidad del diseño arquitectónico y se incrementa la reusabilidad, flexibilidad 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.

Actualmente, este patrón es ampliamente utilizado en el desarrollo de aplicaciones web ya que resulta muy natural acoplarlo con el modelo cliente-servidor que utiliza la web. El sistema eFuel fue desarrollado sobre una plataforma de desarrollo web que implementa el patrón MVC y, en consecuencia, el código tiene la estructura descrita por este patrón.

2.1.4.3. REST

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

Las restricciones definidas para los sistemas REST son las siguientes:

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

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

3. **Permite el uso de memoria caché** la respuesta debe estar explícita o implícitamente etiquetada como *cacheable* (se puede guardar en alguna memoria caché) o *non-cacheable* (no se puede guardar en ninguna memoria caché). La ventaja del

uso de memoria caché es que se pueden eliminar parcial o completamente algunas interacciones mejorando así el rendimiento del sistema.

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

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

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

CAPÍTULO III

MARCO TECNOLÓGICO

En este capítulo se definen las tecnologías y herramientas utilizadas para el desarrollo del sistema. Entre estas se incluyen los *frameworks* utilizados para el desarrollo (ASP.NET, ASP.NET MVC, ASP.NET Web API y Umbraco), los lenguajes utilizados (C# para el *backend*, y HTML, CSS y JavaScript para el *frontend*, como es común en el desarrollo web), la herramienta de control deversiones (Git) y el entorno de desarrollo (Visual Studio).

3.1. Frameworks

3.1.1. ASP.NET

ASP.NET es un modelo de desarrollo Web unificado que incluye los servicios necesarios para construir aplicaciones Web empresariales 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. [9]

3.1.2. ASP.NET MVC

ASP.NET MVC es un marco de trabajo (o *framework* en inglés) para construir aplicaciones web escalables, basadas en estándares y usando patrones de diseño bien establecidos (el patrón MVC) y el poder de ASP.NET y .NET Framework. [10]

Se utilizó este marco de trabajo para el desarrollo del módulo **EF_Core** (ver sección *sección de arquitectura*) del proyecto, allí están implementados los controladores (ver sección 2.1.4.2) de la aplicación web.

3.1.3. ASP.NET Web Api

ASP.NET Web API es un marco de trabajo que facilita la construcción de servicios HTTP que llegan a una amplia gama de clientes, incluyendo navegadores y dispositivos móbiles. ASP.NET Web API es una plataforma ideal para construir aplicaciones RESTful en .NET Framework. [11]

Este marco de trabajo se utilizó para el desarrollo del módulo **EF_API** (ver sección *sección de arquitectura*) del proyecto, allí están implementados los servicios del API para acceder a información del sistema (ver sección 2.1.3) de la aplicación web.

3.1.4. Umbraco

Umbraco es un Sistema de Gestión de Contenido gratuito y de código abierto construido sobre ASP.NET, fue desarrollado en Dinamarca y su primera versión fue lanzada en el año 2005. Umbraco cuenta con funcionalidades que son necesarias para el sistema: la autenticación de usuarios, permisología, interfaces para realizar operaciones sobre la base de datos, entre otras. La plataforma también cuenta con un repositorio de paquetes y extensiones que proveen otras funcionalidades útiles para la aplicación web y provee una variedad de librerías y de módulos que facilitan el desarrollo del sistema.

Cabe destacar que la empresa tiene varios años de experiencia desarrollando aplicaciones web con Umbraco y varios de sus empleados tienen certificaciones y un nivel alto de conocimiento de la plataforma lo cual facilitó el desarrollo del sistema. La aplicación web eFuel fue desarrollada con Umbraco v7.10.

3.2. Lenguajes

A continuación se definen brevemente los lenguajes de programación utilizados para el desarrollo del proyecto divididos en *front-end* y *back-end*.

3.2.1. *Front-end*

3.2.1.1. HTML

HTML (siglas en inglés para Lenguaje de Marcado de HiperTexto), es el lenguaje de marcado central de la *World Wide Web* (red informática mundial). Originalmente, HTML

fue diseñado principalmente para describir semánticamente documentos científicos. Sin embargo, la generalidad de su diseño ha permitido que sea adaptado, en los años siguientes, para describir otros tipos de documentos y hasta aplicaciones. [14]

Es el lenguaje que se usa para definir la estructura y el contenido de una página web, en el caso de este proyecto se usa para describir las vistas del sistema.

3.2.1.2. CSS

Hojas de Estilo en Cascada, o CSS por sus siglas en inglés (*Cascading Style Sheets*), es un lenguaje de hojas de estilo que permite a los autores y usuarios adjuntar estilos (*e.g.* fuentes y espaciados) a documentos estructurados (*e.g.* documentos de HTML). Al separar el estilo de presentación del contenido de los documentos, se simplifica la autoría y el mantenimiento de los sitios. [8]

3.2.1.3. JavaScript

JavaScript es un lenguaje de *scripting* o programación que permite la creación de contenido dinámico, control de multimedia, animación de imágenes [3]. *mencionar las llamadas al servidor para traer datos con el API*

3.2.2. *Back-end*

3.2.2.1. C#

C# es un lenguaje con seguridad de tipos y orientado a objetos que permite a desarrolladores construir una variedad de aplicaciones robustas y seguras que corren sobre .NET Framework. [12]

3.3. Control de versiones

3.3.1. Git

Git es un sistema de gestión de versiones rápido, escalable y distribuido con un rico conjunto de comandos que proveen operaciones de alto nivel y acceso completo a internos.

[6] Fue desarrollado por Linus Torvald en el año 2005 para facilitar el trabajo de varios desarrolladores sobre un mismo proyecto. Permite llevar el seguimiento de los cambios a un grupo de archivos y sincronizar varios repositorios en máquinas distintas.

Este es el sistema de gestión de versiones usado en la empresa en la que se desarrolló el proyecto.

3.4. Entorno de trabajo

3.4.1. Visual Studio

Visual Studio es un ambiente de desarrollo integrado que permite editar, depurar, compilar y publicar código. [13] Este ambiente de desarrollo incluye una gran cantidad de funcionalidades para facilitar el desarrollo de software: depuración del código paso a paso y con información detallada de las variable y otras entidades del programa, instrucciones de compilación complejas para la aplicación, descarga y actualización paquetes y librerías, integración con Git, *IntelliSense* de Microsoft para la completación de partes de código usando el contexto de la aplicación (clases y sus relaciones y métodos), entre otras.

Este entorno de trabajo está muy bien integrado con los *frameworks* utilizados para el proyecto (ASP.NET y sus extensiones/derivados), por lo que su uso resultó ventajoso y natural para el desarrollo del proyecto.

3.4.2. Trello

Trello es una herramienta de colaboración que organiza las tareas de un proyecto en tablas. Trello informa en qué se está trabajando, quién está trabajando en qué, y el estado de una tarea en el proceso de desarrollo.

Cada tabla tiene varias listas y cada lista contiene tarjetas. Una tabla corresponde a un proyecto, una lista corresponde al estado de una tarea y una tarjeta a una tarea, por ejemplo, en una tabla suele haber 3 listas: *Por hacer*, *Haciendo* y *Lista*. Las tarjetas contienen una descripción de la tarea, la o las personas asignadas y puede tener una fecha límite de entrega. A medida que se va completando trabajo se van moviendo las tarjetas de una lista a otra.

Resultó una herramienta útil para llevar el control de las tareas realizadas y por realizar. Se usó esta herramienta como apoyo para el uso de Scrum (ver capítulo IV).

CAPÍTULO IV

MARCO METODOLÓGICO

En este capítulo se describirá el marco de desarrollo Scrum que fue utilizado como metodología para el desarrollo del sistema, se dará una definición de Scrum y luego se definirán los roles de un equipo Scrum, los artefactos utilizados y los eventos que deben ocurrir durante el desarrollo.

4.1. Definición de Scrum

Scrum es un marco de trabajo dentro del cual la gente puede abordar un problema adaptativo complejo, mientras se entregan productos del mayor valor posible. Es un marco de trabajo para procesos que ha sido usado para gestionar el trabajo en productos complejos desde los inicios de la década de los 90. Scrum no es un proceso, ni una técnica o método definitivo. Más bien, es un marco de trabajo dentro del cual se pueden emplear varias técnicas y procesos. El marco de Scrum consiste en Equipos Scrum y sus roles, eventos, artefactos y reglas asociados. [15]

4.2. El equipo de Scrum

El equipo de Scrum está formado por el Dueño del Producto, el Equipo de Desarrollo y un *Scrum Master*. Los equipos de Scrum son auto-organizados y auto-suficientes. Los equipos auto-organizados escogen la mejor forma de llevar a cabo el trabajo, en vez de ser dirigidos por otras personas fuera del equipo. Los equipos auto-suficientes tienen todas las competencias necesarias para terminar el trabajo sin depender de otros fuera del equipo. [15] A continuación se describirán los roles de un equipo Scrum:

4.2.1. Dueño del Producto

Es el responsable de maximizar el valor del producto desarrollado por el Equipo de Desarrollo. Es el responsable de manejar la Lista de Objetivos del Producto (ver sección 4.4.1), esto incluye:

- Expresar los items de la Lista de Objetivos del Producto.
- Ordenar los items de la Lista de Objetivos para lograr los objetivos de la mejor manera posible.
- Optimizar el valor del trabajo realizado por el Equipo de Desarrollo (ver sección 4.2.2).
- Asegurarse de que la Lista de Objetivos del Producto sea visible, transparente y clara para todos.
- Asegurarse de que el Equipo de Desarrollo entienda los items a un nivel adecuado.

El Dueño del Producto puede realizar el trabajo descrito arriba o puede delegarlo al Equipo de Desarrollo, sin embargo, siempre será responsable de este.

Para que el Dueño del Producto sea exitoso, toda la organización debe respetar sus decisiones. Estas decisiones son visibles en el contenido y el ordenamiento de la Lista de Objetivos del Producto. [15]

4.2.2. Equipo de Desarrollo

El Equipo de Desarrollo es un grupo de profesionales que realizan el trabajo para entregar un Incremento (ver sección 4.4.4) listo del producto para cada Iteración (ver sección 4.3.1). Los Equipos de Desarrollo están estructurados y tienen la potestad de gestionar su propio trabajo. Los Equipos de Desarrollo tienen las siguientes características:

- Están auto-organizados. Nadie le dice al equipo cómo debe llevar a cabo las tareas de la Lista de Objetivos del Producto.
- Son auto-suficientes, es decir, tienen todas las habilidades necesarias para llevar crear un Incremento.
- Scrum no tiene títulos para las personas que conforman el equipo, independientemente del trabajo que haga cada persona.

- Scrum no reconoce sub-equipos dentro del un Equipo de Desarrollo.
- Los miembros del equipo pueden tener habilidades específicas, pero la responsabilidad pertenece al equipo con un todo. [15]

4.2.3. Facilitador

El Facilitador (o *Scrum Master*) es el responsable de promover y apoyar Scrum como está definido en la Guía de Scrum (ver [15]). Esto lo logran ayudando a todo el Equipo Scrum a entender la teoría, práctica, reglas y valores de Scrum. [15]

4.3. Eventos de Scrum

4.3.1. La Iteración

4.4. Artefactos de Scrum

4.4.1. Lista de Objetivos del Producto

La Lista de Objetivos del Producto es una lista ordenada de todas las tareas que deben ser completadas para entregar el producto. Esta lista nunca está completa, la primera versión muestra los primeros requerimientos conocidos y mejor entendidos. [15] La Lista de Objetivos del Producto evoluciona junto con el producto para actualizar los requerimientos dependiendo de las exigencias del cliente y del mercado. El Dueño del Producto (ver sección 4.2.1) es el responsable de este artefacto.

4.4.2. Lista de Tareas de la Iteración

La Lista de Tareas de la Iteración es el conjunto de tareas de la lista de objetivos del producto seleccionadas para completar en una Iteración, puede verse como una predicción de lo que estará completado al llevarse a cabo la Iteración. [15] Esta lista va a ir cambiando a medida que el Equipo de Desarrollo identifique tareas y trabajo que debe ser realizado para completar los items durante el desarrollo de la Iteración, es decir, además de las tareas tomadas de la Lista de Objetivos del Producto, pueden surgir nuevas tareas que deben ser cumplidas para llevar a cabo el trabajo de la Iteración.

4.4.3. Tablero de tareas

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

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

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

4.4.4. Incremento

CAPÍTULO V

DESARROLLO

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] Center, Mozilla Development: *What is Javascript?*, 2018. `https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript`, visitado el 2018-09-21.

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

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

[6] Git: *git*, 2018. `https://git-scm.com/`, visitado el 2018-09-21.

[7] 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.

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

[9] Microsoft: *ASP.NET Overview*. `https://msdn.microsoft.com/en-us/library/4w3ex9c2.aspx`, visitado el 2018-09-19.

[10] Microsoft: *ASP.NET MVC 3.*, 2010. `https://docs.microsoft.com/en-us/aspnet/mvc/mvc3`, visitado el 2018-09-19.

[11] Microsoft: *ASP.NET Web API.*, 2010. https://msdn.microsoft.com/en-us/library/hh833994(v=vs.108).aspx, visitado el 2018-09-19.

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

[13] Microsoft: *Welcome to Visual Studio IDE.*, 2018. https://docs.microsoft.com/en-us/visualstudio/ide/visual-studio-ide?view=vs-2017, visitado el 2018-09-21.

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

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

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

[17] Umbraco: *The Flexible CMS*, 2018. https://umbraco.com/tour, visitado el 2018-09-26.