UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

ANÁLISIS COMPARATIVO DE ALGUNOS FRAMEWORKS PARA LA CONSTRUCCIÓN DE APLICACIONES WEB

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

P R E S E N T A :

JULIO CÉSAR GARCÍA GARCÍA

Director de Tesis:

M. EN C. GUSTAVO ARTURO MÁRQUEZ FLORES

Facultad de Ciencias, UNAM

Ciudad Universitaria, Cd. Mx. Diciembre 2017

Agradecimientos

A mi madre y a mi padre por toda su confianza y por el apoyo que me han brindado a lo largo de mi vida.

Agradezco al M. en C. Gustavo Arturo Márquez Flores por todos sus consejos, enseñanzas y por haberme apoyado durante el transcurso de la maestría.

A los profesores de las asignaturas que tuve la oportunidad de cursar, por todo el conocimiento que me compartieron durante sus clases. Y a los sinodales por el tiempo que dedicaron en revisar este trabajo.

Agradezco a los amigos que me apoyaron para ingresar y a todos con los que compartí momentos en el Posgrado.

Finalmente quiero agradecer a la Universidad Nacional Autónoma de México por la formación que he recibido en los distintos niveles educativos que he cursado, en especial en la Escuela Nacional Preparatoria Plantel 3 “Justo Sierra”, en la Facultad de Ingeniería y en el Posgrado en Ciencia e Ingeniería de la Computación.

Índice general

Introducción

Contexto

En la actualidad el desarrollo Web se ha convertido en una de las principales actividades dentro del desarrollo de software, debido al creciente uso de las aplicaciones y servicios Web. Cada vez hay más personas que realizan compras o pagan servicios como el teléfono, la electricidad o el gas, mediante aplicaciones Web. Es por esto que ahora las organizaciones tienen la necesidad de crear una aplicación o servicio Web específico, para que los usuarios puedan consultar o pagar sus servicios a través de la Web.

Las aplicaciones Web actuales son sistemas de software complejos, que proporcionan servicios interactivos y personalizados, con una gran cantidad de datos, que además se pueden acceder a través de diferentes dispositivos y que facilitan la realización de transacciones [1].

Los usuarios esperan que las aplicaciones Web sean más confiables, seguras, personalizadas y fáciles de usar. A medida que la dependencia y confianza en las aplicaciones basadas en Web ha aumentado a través de los años, aspectos como el rendimiento, la seguridad, la facilidad de uso, la escalabilidad y la calidad de este tipo de aplicaciones, han adquirido una importancia primordial.

Actualmente los frameworks para aplicaciones Web o simplemente frameworks Web, son uno de los enfoques más utilizados para la construcción de aplicaciones Web, porque permiten la creación y mantenimiento de este tipo de aplicaciones, de forma rápida y eficiente [2]. Un framework Web se puede definir de manera general como un conjunto integrado de componentes que forman un diseño o una arquitectura reutilizable para aplicaciones Web.

Planteamiento del problema

La constante aparición de nuevos frameworks Web y la actualización de los existentes, así como de componentes y herramientas usadas en la construcción de aplicaciones Web, ha ocasionado que los desarrolladores tengan dificultades para elegir el framework más adecuado para la creación de una aplicación Web.

El no contar con un enfoque para el análisis o la evaluación de frameworks Web, provoca que la selección del framework sea más complicada. A pesar de estos inconvenientes los desarrolladores prefieren utilizar los frameworks para aplicaciones Web, porque estos les permiten establecer una estructura, reutilizar componentes de software y evitar muchos de los errores típicos en la construcción de una aplicación Web.

Aun cuando existen una gran cantidad de sitios Web donde se habla de las ventajas y/o desventajas que puede ofrecer un framework Web sobre otro, la mayoría de las ideas expresadas en estos sitios no suelen ser objetivas, ya que generalmente no tienen un fundamento o algún punto de comparación.

En resumen, es difícil encontrar un trabajo (libro, guía o reporte) donde se analice a los frameworks para aplicaciones Web, de tal forma que ayude en la selección de este tipo de frameworks.

Objetivo

El objetivo de este trabajo es presentar una forma para comparar frameworks Web, que ayude en el proceso de selección de un framework para la construcción de una aplicación Web.

Propuesta de solución

Para lograr el objetivo de esta tesis, fue necesario determinar criterios que permitieran la comparación de varios frameworks Web. Los criterios se identificaron a partir de las características comunes de este tipo frameworks y se agruparon en aspectos que son relevantes para una aplicación Web, como el acceso a datos, la seguridad, las pruebas, la portabilidad y la interoperabilidad. Las principales actividades que se realizaron en este trabajo son las siguientes:

1. Se eligieron algunos frameworks para aplicaciones Web, para su posterior análisis y comparación.
2. Se identificaron algunas características presentes en distintos frameworks Web, las cuales se agruparon en aspectos que son relevantes para las aplicaciones Web.

3. A partir de las características y aspectos previamente reconocidos, se identificaron algunos criterios para comparar a los frameworks para aplicaciones Web.
4. Para cada uno de los frameworks Web elegidos, se describió lo fundamental acerca de cómo son proporcionadas cada una de las características identificadas.
5. Se realizó la comparación de los frameworks para aplicaciones Web.

Contribuciones

Como se mencionó anteriormente, es difícil encontrar un trabajo en el que se especifiquen las características de los frameworks Web en general y que además presente una forma para analizar o evaluar este tipo de frameworks. Dado que la selección del framework es una parte fundamental en el desarrollo de una aplicación Web, contar con un enfoque que ayude a determinar cuál es el framework Web más adecuado para la construcción de cierta aplicación Web, se vuelve relevante.

Considerando lo anterior, las principales contribuciones de este trabajo son las siguientes:

1. La identificación de características comunes en los frameworks Web y la asociación con aspectos que son relevantes para las aplicaciones Web.
2. La identificación de criterios para la comparación de frameworks Web, que ayuden en el proceso de selección de este tipo de frameworks.

Trabajos relacionados

Los trabajos relacionados con el análisis y la comparación de frameworks para aplicaciones Web son escasos. Incluso es difícil encontrar trabajos donde se especifiquen las características de los frameworks Web en general.

En [3] se presenta un enfoque para la comparación de frameworks Web. En este trabajo se identificaron y analizaron las buenas prácticas que se pueden emplear con Lift (framework Web), para compararlo con otros frameworks Web en el uso de estas buenas prácticas. Si bien es una forma interesante para comparar a este tipo de frameworks, la evaluación realizada no es completamente objetiva, debido a que los

criterios de comparación son únicamente las buenas prácticas identificadas en Lift, por lo que este framework es el único que cumple el uso de todas las buenas prácticas.

Estructura de la tesis

Este trabajo se organiza de la siguiente manera:

- En el capítulo 1 se describen las características, una arquitectura común y los enfoques para la construcción de aplicaciones Web. De los enfoques, se revisa principalmente el de los frameworks para aplicaciones Web.
- En el capítulo 2 se eligen dos frameworks Web, para su posterior análisis y comparación mediante criterios de aspectos que son relevantes para una aplicación Web.
- Los capítulos 3 y 4 corresponden a cada uno de los frameworks Web elegidos en el capítulo 2. En estos capítulos se describe la forma en que dichos frameworks proporcionan las características comunes, que se lograron identificar en los frameworks Web.
- En el capítulo 5 se presenta la comparación de los frameworks Web, mediante los criterios identificados en el capítulo 2.

Capítulo 1

Marco Teórico

En este capítulo se presentan los conceptos fundamentales para el trabajo desarrollado en capítulos posteriores. Se describen las características y una arquitectura común de aplicaciones Web, así como los enfoques para su construcción. De los enfoques se revisa principalmente el de los frameworks Web.

1.1. Aplicaciones Web

Una definición de lo que es una aplicación Web es la siguiente [1]:

"Una aplicación Web es un sistema de software basado en tecnologías y estándares del World Wide Web Consortium (W3C) que proporciona recursos específicos de la Web, tales como contenido y servicios a través de una interface de usuario, el navegador Web."

La característica distintiva de las aplicaciones Web en comparación con otro tipo de aplicaciones de software es la forma en que utilizan la Web, las tecnologías y estándares de Internet son usados como plataforma de desarrollo y como plataforma de usuario al mismo tiempo [1].

Las aplicaciones Web tienen como base a algunas tecnologías de Internet fundamentales, en particular, HTTP, XHTML y JavaScript, pero también se componen de mecanismos para la administración de sesiones, el almacenamiento en caché, la persistencia de datos, la validación de las entradas de formularios y de interfaces de usuario enriquecidas. La seguridad es un aspecto muy importante en este tipo de aplicaciones, debido a que están expuestas a distintas formas de ataques informáticos.

1.1.1. Características de las aplicaciones Web

Las aplicaciones Web tienen distintas características, pero la gran mayoría de estas aplicaciones comparten las siguientes [4]:

- **Uso de red**. Una aplicación Web debe satisfacer las necesidades de distintos tipos de clientes, para lo cual, la aplicación Web debe residir en una red y permitir el acceso y la comunicación mediante la Internet o una Intranet.
- **Concurrencia**. Una gran cantidad de usuarios puede acceder a una aplicación Web al mismo tiempo.
- **Carga impredecible**. El número de usuarios que utilizan una aplicación Web puede variar día con día.
- **Desempeño**. Los tiempos de respuesta al utilizar una aplicación Web deben ser los mínimos posibles. Si un usuario debe esperar demasiado tiempo para obtener una respuesta a su solicitud, puede decidir dejar la aplicación Web.
- **Disponibilidad**. En la actualidad los usuarios de aplicaciones Web demandan acceso las 24 horas de los 365 días del año, aunque pensar en una aplicación Web con disponibilidad del 100% tampoco es lo más adecuado, por lo que se debe procurar tener una disponibilidad cercana a este porcentaje.
- **Orientadas a los datos**. Muchas aplicaciones Web tienen como función principal presentar texto, gráficos, audio y video al usuario final. En general las aplicaciones Web se utilizan para acceder y mostrar la información contenida en bases de datos.
- **Sensibles al contenido**. La calidad y la naturaleza del contenido es un atributo de calidad importante de una aplicación Web.
- **Orientadas a transacciones**. Las transacciones permiten más interacción con el usuario, proporcionan que el contenido se pueda actualizar de manera constante en una aplicación Web.
- **Evolución continúa**. Las aplicaciones Web evolucionan de manera continua. Por ejemplo, es común que algunas aplicaciones Web actualicen su contenido minuto a minuto.
- **Inmediatez**. La necesidad de poner rápidamente en funcionamiento una aplicación Web es una característica de muchos negocios.
- **Seguridad**. Debido que las aplicaciones Web se acceden a través de una red, deben implementarse mecanismos para proteger la información confidencial y ofrecer modos seguros de transmisión de datos.
- **Estética**. La presentación y colocación de los elementos en las páginas Web son cualidades de las aplicaciones Web. La estética tiene la misma importancia que los aspectos técnicos en aplicaciones Web que son diseñadas con el objetivo de vender productos o servicios.

1.1.2. Arquitectura de una aplicación Web

La arquitectura de una aplicación Web es el conjunto de estructuras necesarias para razonar sobre la aplicación, que comprende elementos de software, las relaciones entre dichos elementos y sus propiedades. Las estructuras forman una herramienta valiosa para el diseño de una aplicación Web [5].

Sin embargo, la arquitectura de una aplicación Web también puede ser vista como el resultado del uso de un conjunto de decisiones de diseño para satisfacer los requerimientos de la aplicación, que eventualmente deben ser efectuadas con una tecnología específica, la cual se debe seleccionar cuidadosamente.

Componentes de una arquitectura de aplicaciones Web

La comunicación entre los componentes de la arquitectura de una aplicación Web se basa en el principio de solicitud-respuesta, es decir, un componente (por ejemplo, un navegador Web) envía una solicitud a otro componente (por ejemplo, un servidor Web) y la respuesta a dicha solicitud se envía a través del mismo canal de comunicación (comunicación síncrona). En una arquitectura genérica de aplicaciones Web se pueden identificar los siguientes componentes, algunos son fundamentales y otros son opcionales [1]:

- Cliente: Generalmente es un navegador Web que es utilizado por un usuario para operar una aplicación Web. La funcionalidad del cliente se puede expandir instalando complementos.
- Cortafuegos: Es una pieza de software que regula la comunicación entre redes inseguras (por ejemplo, Internet) y redes seguras (por ejemplo, las Redes de Área Local corporativas), mediante reglas de acceso.
- Servidor Web: Es el software que soporta varios protocolos de Internet como HTTP o HTTPS, para procesar las solicitudes de clientes.
- Servidor de bases de datos: Este servidor normalmente proporciona los datos de una organización en forma estructurada, por ejemplo, en tablas.
- Servidor de aplicaciones: Tiene alguna funcionalidad requerida por varias aplicaciones, por ejemplo, flujo de trabajo o personalización.
- Aplicaciones heredadas: Una aplicación heredada es un sistema antiguo que debe ser integrado como un componente interno o externo.

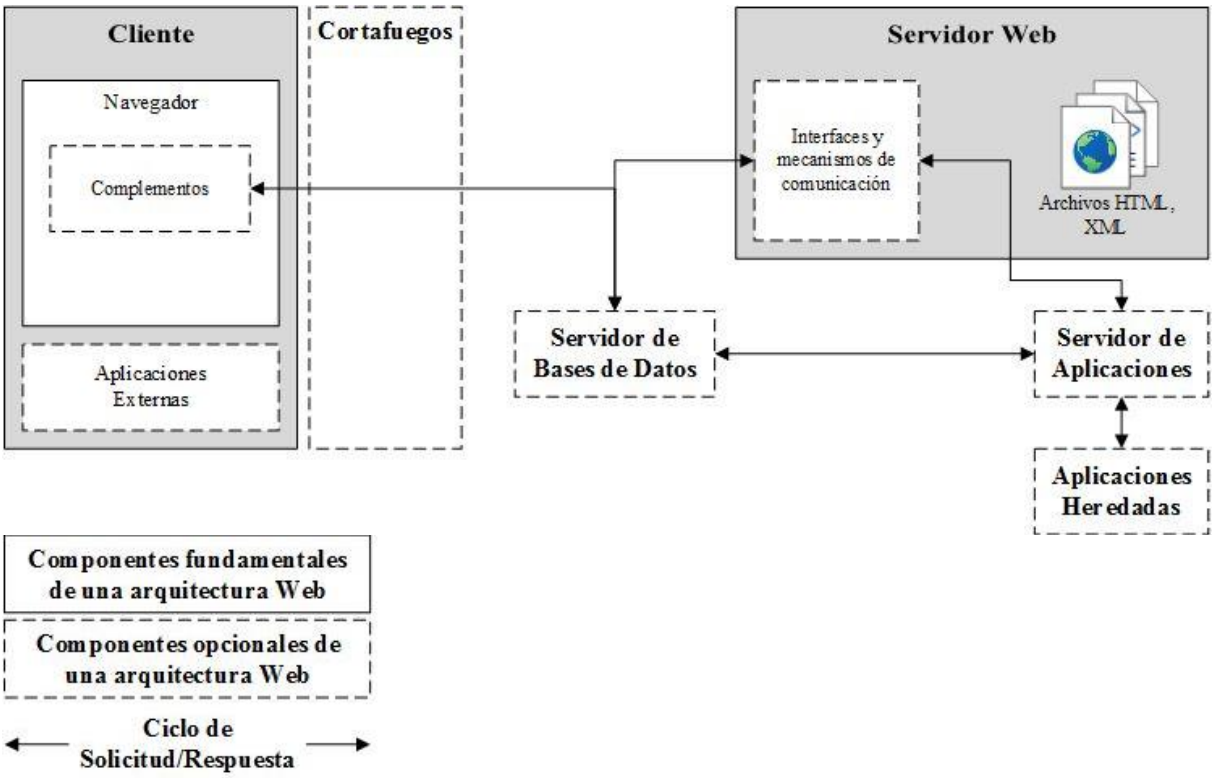

Figura 1.1: Componentes de una arquitectura genérica de aplicaciones Web

En la figura 1.1 se pueden observar los componentes fundamentales representados por rectángulos con línea continua y los componentes opcionales en rectángulos con línea no continua.

Arquitectura en capas de aplicaciones Web

Una aplicación Web es una aplicación que se puede acceder a través de un navegador Web o un agente de usuario especializado. En el navegador Web típicamente se crea una solicitud HTTP a través de un URL (Uniform Resource Locator), que frecuentemente es asociado con un recurso de un servidor Web, el servidor procesa la solicitud y comúnmente devuelve páginas HTML al cliente, las cuales son visualizadas mediante el navegador Web.

La arquitectura de una aplicación Web puede tener varias capas, aunque la arquitectura más común es en tres capas, las cuales se describen a continuación:

1. La capa de presentación normalmente incluye componentes de la interface de usuario y de la lógica de presentación;

2. La capa de negocio incluye componentes de la lógica y las entidades de negocio y opcionalmente una fachada.
3. La capa de datos generalmente incluye componentes de acceso a datos y agentes de servicio.

Además una aplicación Web normalmente tiene acceso a datos que están almacenados en una base de datos, la cual se puede encontrar en un servidor para dicho propósito. Una aplicación Web también puede establecer una comunicación para el intercambio información con otras aplicaciones, a través de servicios expuestos, como los Servicios Web. Además existen otros aspectos, tales como la seguridad, comunicación y pruebas, que afectan varias de las capas de la arquitectura de una aplicación. El ejemplo de una arquitectura común en capas de aplicaciones Web se muestra a continuación [6]:

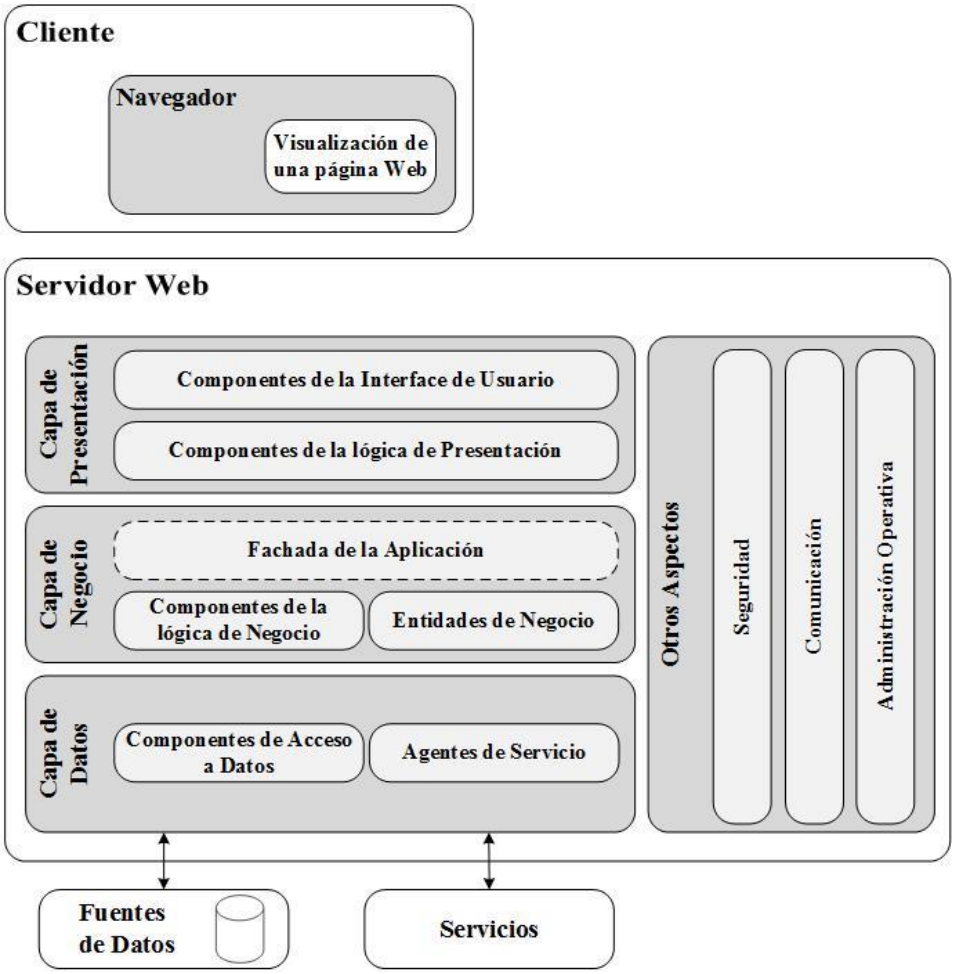

Figura 1.2: Arquitectura en capas de aplicaciones Web

1.1.3. Enfoques para la construcción de aplicaciones Web

La mayoría de los enfoques establecidos para la construcción de aplicaciones Web, se pueden dividir en cuatro categorías generales, las cuales se distinguen principalmente por el tipo de objeto que utilizan como "fuente" para generar las páginas y por el grado de soporte que proporcionan para la construcción de aplicaciones avanzadas y escalables [2].

Enfoques programáticos o programación basada en scripts (scripting)

En estos enfoques una página es generada principalmente por código escrito en un lenguaje de scripting (como Perl o Python) o en un lenguaje de programación de alto nivel, como Java. Las etiquetas HTML y las demás estructuras usadas para generar el formato de una página son producidas utilizando las instrucciones de salida del lenguaje de programación asociado, es decir, todo está incrustado dentro de la lógica del programa.

La intervención del programador se requiere para modificar prácticamente cualquier aspecto de la página generada, ya sea un cambio en la lógica del programa o un cambio en el diseño. El mayor problema con los enfoques programáticos es su naturaleza centrada en el código. Los Servlets son un ejemplo de este tipo de enfoques.

Enfoques de plantillas

Los enfoques basados en plantillas utilizan como objeto fuente una plantilla, que se compone principalmente de estructuras de formato, como etiquetas HTML con el código necesario que permita agregar potencia programática limitada, es decir, se enfocan en el formato y no en la lógica de programación.

Estos enfoques son más amigables con los diseñadores Web, cuya experiencia es en el área de diseño en lugar de la programación. Los enfoques de plantillas se centran en la estructura de la página y las etiquetas de formato y no en el código, por lo que comúnmente son llamados enfoques centrados en la página. Algunos ejemplos de enfoques basados en plantillas son Adobe Cold Fusion y Apache Velocity.

Enfoques híbridos

Los enfoques híbridos emplean estructuras de formato con código para generar las páginas Web. Tienen más poder programático que los enfoques de plantillas porque estos permiten agregar bloques de código. Combina los beneficios de la plantilla y los

enfoques programáticos. La mayoría de estos enfoques traducen objetos fuente híbridos en código y los más sofisticados emplean alguna forma de pre-compilación, de forma que el proceso de traducción y compilación no se produzca cada vez que se solicita una página Web. Ejemplos de enfoques híbridos son ASP (Active Server Pages) y JSP (Java Server Pages).

Frameworks Web

Los frameworks Web representan el siguiente nivel para la construcción de aplicaciones Web. Proporcionan una infraestructura consistente que incluye un amplio conjunto de servicios, mediante los cuales se reduce la necesidad de escribir código redundante para las funciones comunes de las aplicaciones.

Estos frameworks permiten que los módulos para generar el contenido sean distintos de los módulos utilizados para presentar el contenido en un formato particular. En lugar de combinar el diseño y la lógica en un solo módulo, los frameworks Web siguen el principio de separar lo referente al contenido de la presentación, pero siempre tratando de conseguir la mayor flexibilidad de la aplicación y la división adecuada de tareas entre los responsables de la construcción de una aplicación Web.

1.2. Frameworks para aplicaciones Web

Un framework para aplicaciones Web o framework Web se puede definir como una plataforma o un conjunto integrado de componentes que forman un diseño o una arquitectura reutilizable, el cual permite producir aplicaciones Web personalizadas.

Considerando a una aplicación Web como un conjunto estructurado de objetos, que para lograr una o más tareas permite navegar y procesar información, un framework Web puede ser visto como una definición genérica de los posibles objetos, aplicable a la forma de navegación y procesamiento de una aplicación Web. El framework Web debe definir el conjunto de posibles objetos para navegar, cómo se pueden estructurar y el comportamiento que pueden tener [7].

1.2.1. Utilidad de los frameworks para aplicaciones Web

Un framework Web puede ser visto como una plataforma abierta y basada en tecnologías y estándares comúnmente aceptados, como Java, .NET, XML, JDBC,

ADO.NET, JUnit, lo que permite que un desarrollador con experiencia pueda construir y dar soporte a una aplicación Web con una curva de aprendizaje poco pronunciada.

En la actualidad casi todos los frameworks Web implementan el patrón de diseño Modelo-Vista-Controlador (MVC). La incorporación de servicios, como búsquedas y control de versiones requiere de muy poco trabajo adicional, lo que permite un mejor aprovechamiento de estos servicios [8]. Actualmente es común ver la integración de estos frameworks con distintos Entornos de Desarrollo Integrado (IDE por sus siglas en inglés), para extender sus características y proporcionar más funcionalidades. Este enfoque de adopción e integración de tecnologías permite que los desarrolladores de aplicaciones, se concentren en la resolución de los problemas de negocio que son realmente importantes.

Los frameworks Web reducen significativamente la cantidad de tiempo, esfuerzo y recursos necesarios, además ayudan a los desarrolladores a evitar muchos de los errores comunes en la construcción y mantenimiento de aplicaciones Web. Una gran cantidad de aplicaciones Web tienen un conjunto común de requerimientos funcionales básicos, como puede ser la administración de usuarios (que incluye creación, modificación y eliminación de usuarios, recuperación de contraseña, etc.), o la implementación de los procesos de autenticación y autorización. En la actualidad los frameworks incluyen módulos para lo anterior, que han sido perfeccionados a partir de cientos de implementaciones, lo cual permite un ahorro de tiempo y que los desarrolladores puedan enfocarse en necesidades más complejas de la aplicación.

1.2.2. Tipos de frameworks para aplicaciones Web

Una de las pocas clasificaciones de framewroks Web, que se puede encontrar en la literatura de este tipo de frameworks es la siguiente [8]:

- **Frameworks basados en solicitudes**. Este tipo de frameworks se encargan de direccionar las solicitudes entrantes. Cada solicitud es manejada sin un estado, pero con sesiones de servidor se logra un cierto estado. Ejemplos de estos frameworks son: Django, Ruby on Rails, Struts y Grails.
- **Frameworks basados en componentes**. Un framework basado en componentes abstrae los aspectos internos relacionados con el manejo de una solicitud, encapsula la lógica en componentes reutilizables y maneja automáticamente el estado de una solicitud. Java Server Faces (JSF) es un ejemplo de este tipo de frameworks.

- **Frameworks Híbridos**. Los frameworks híbridos combinan características de los frameworks basados en solicitudes y de los basados en componentes, para tomar el control de todo el flujo de datos y la lógica en un modelo basado en solicitudes. Los componentes pueden ser conectados entre sí y empaquetados como grupos de componentes. Los desarrolladores tienen control total sobre los URLs, formularios, parámetros, cookies y rutas de información. Un ejemplo es RIFE, que es un framework de Java para la administración de contenido.
- **Frameworks basados en Aplicaciones de Internet Enriquecidas**. Un framework de este tipo está basado en Aplicaciones de Internet Enriquecidas (RIA por sus siglas en inglés), las cuales son aplicaciones que utilizan un navegador Web para ejecutarse, pero que cuentan con características adicionales (principalmente de aplicaciones de escritorio), que son agregadas a través de complementos o mediante una máquina virtual. Un framework basado en RIA minimiza la comunicación con el servidor, lo que significa que del lado del cliente se tiene una aplicación con un estado y un modelo de interacción con el usuario, el lado del cliente tiene mucho más que una página Web generada del lado del servidor. Adobe Flex es un framework de este tipo.
- **Metaframeworks**. Estos frameworks tienen un conjunto de interfaces esenciales para servicios comunes y una base altamente extensible para la integración de componentes y servicios de software. Su estructura es abierta y flexible para incorporar otros frameworks y componentes.

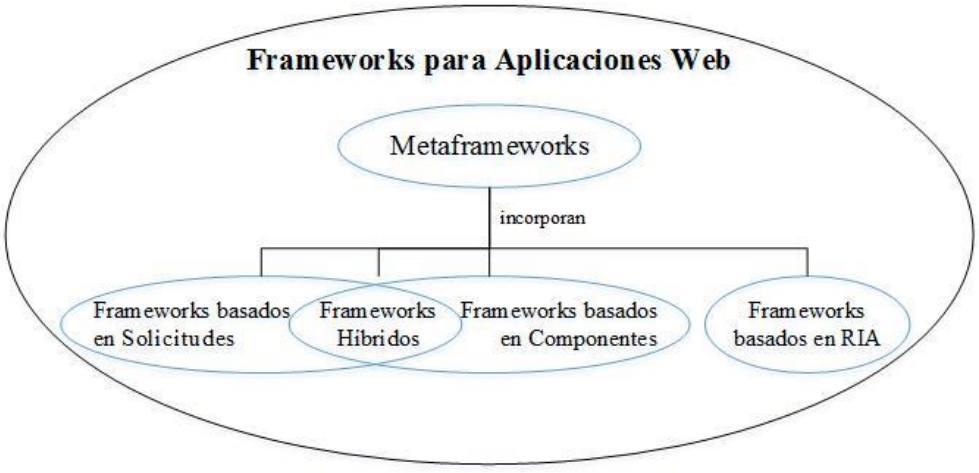

Figura 1.3: Tipos de frameworks para aplicaciones Web

Capítulo 2

Determinando los frameworks Web y los criterios para su comparación

En el presente capítulo se eligen algunos frameworks Web, que posteriormente serán analizados y comparados. Enseguida se especifican varias preguntas a partir de la abstracción de este tipo de frameworks. Estas preguntas son agrupadas en aspectos importantes para las aplicaciones Web y servirán como criterios de comparación para los frameworks Web elegidos.

2.1. Eligiendo los frameworks Web a comparar

Existen varios factores que deben ser considerados cuando se elige un framework Web, tales como el propósito de la aplicación, el uso de licencias, la curva de aprendizaje o las preferencias personales de los desarrolladores. La popularidad es otro factor importante y puede servir como primer criterio de elección. Es sabido que los frameworks populares tienen comunidades de apoyo, que aportan documentación apropiada y soporte, lo cual es de gran ayuda sobre todo para los desarrolladores con poca experiencia, además proveen extensiones y complementos a la funcionalidad principal, validaciones más rápidas y corrección de defectos, lo que en general proporciona mayor seguridad y dirige a un desarrollo de aplicaciones Web con mejor calidad.

Dos indicadores interesantes que se pueden utilizar para estimar la popularidad de los frameworks Web en la actualidad son: el porcentaje de uso y el número de libros publicados de las tecnologías de frameworks Web.

La empresa de servicios de Internet BuiltWith proporciona información sobre las tendencias de tecnologías Web y el uso de estas en los sitios Web del mundo. En la gráfica de la figura 2.1, recuperada de su sitio Web, se puede visualizar el porcentaje de uso de las principales tecnologías de frameworks Web, en los diez mil sitios Web más importantes del mundo [9].

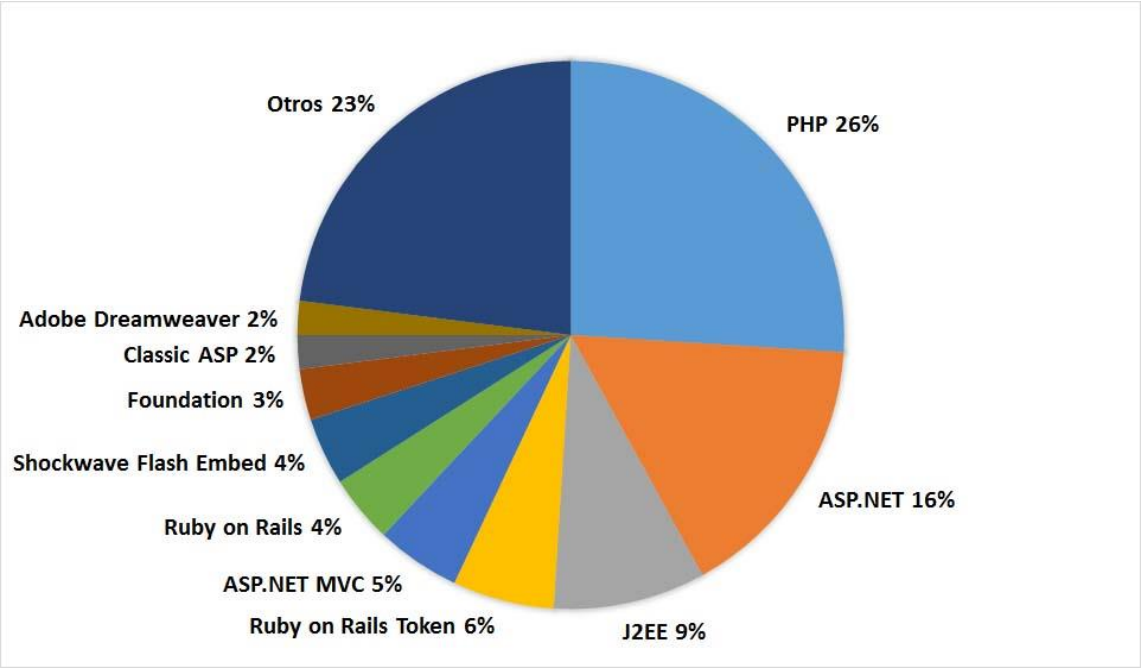

Figura 2.1: Porcentaje de uso de las principales tecnologías de frameworks Web (Extendido)

En la gráfica anterior se observa una separación entre ASP.NET, ASP.NET MVC y Classic ASP, pero en la actualidad estas son parte de la misma tecnología (ASP.NET). Además Ruby on Rails Token es solo Ruby on Rails. En la gráfica de la figura 2.2 se muestran los porcentajes reales para cada tecnología de frameworks Web.

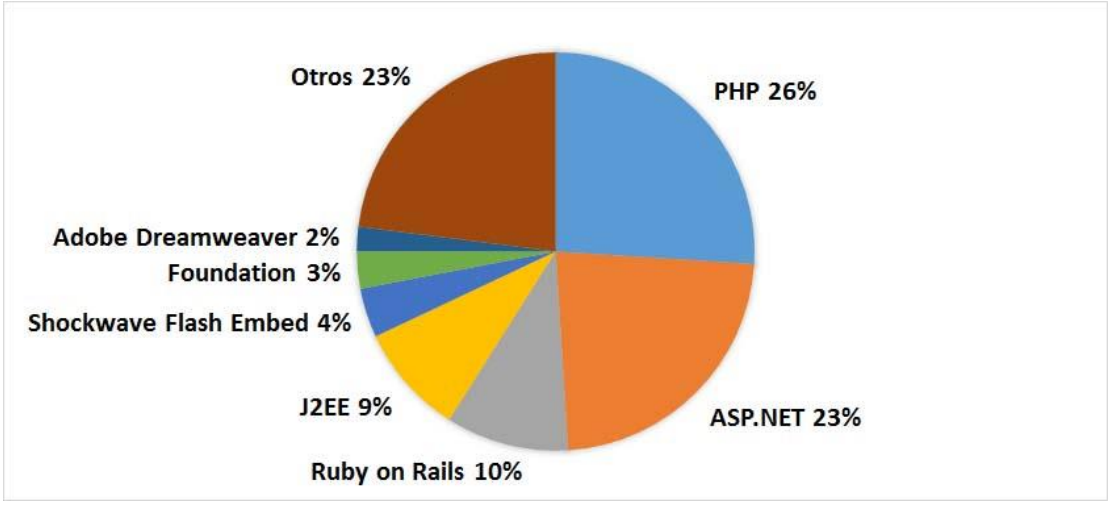

Figura 2.2: Porcentaje de uso de las principales tecnologías de frameworks Web (Reducido)

A pesar de que Shockwave Flash Embed, Foundation y Adobe Dreamweaver son consideradas por BuiltWith como tecnologías de frameworks Web, dichas tecnologías en realidad no lo son. En Shockwave Flash Embed la programación se realiza principalmente con scripts, es decir, pertenece a un enfoque programático, Foundation se ajusta más a un enfoque hibrido y Adobe Dreamweaver es un editor muy robusto, en el que a menudo se utiliza un enfoque basado en plantillas, para la creación de aplicaciones Web. Por lo que las únicas tecnologías de frameworks Web que se pueden considerar son: PHP, ASP.NET, Ruby on Rails y J2EE.

Por otra parte, Amazon comenzó como una tienda de libros en línea y en la actualidad es una de las tiendas en línea más grandes del mundo, en la que se puede encontrar prácticamente cualquier libro. La siguiente gráfica muestra el número de libros que se pueden encontrar en Amazon, relacionados con frameworks Web de las tecnologías de frameworks Web anteriores [10].

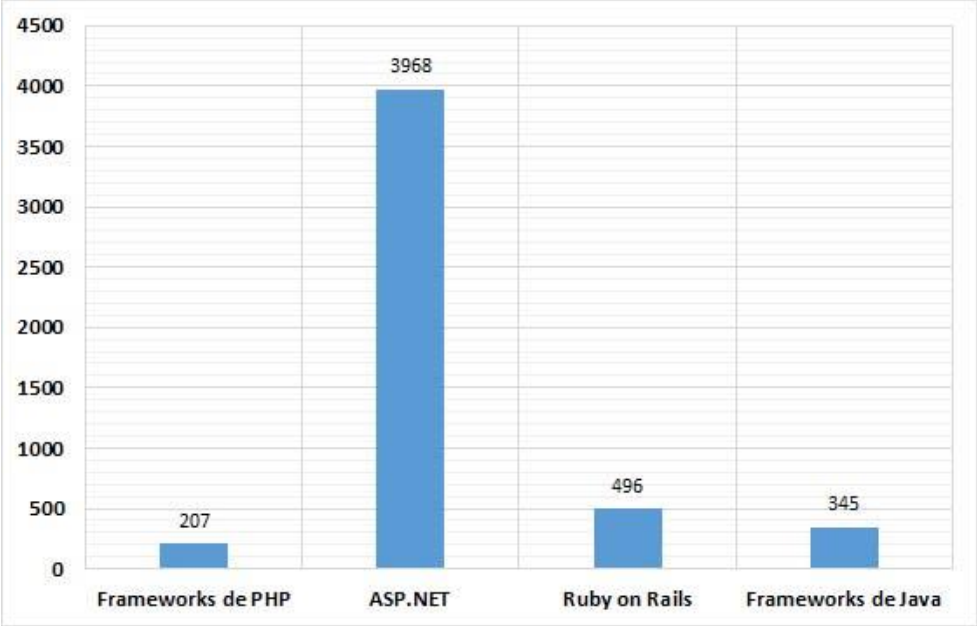

Figura 2.3: Número de libros publicados de las principales tecnologías de frameworks Web

De lo anterior se tiene que las cuatro tecnologías de frameworks Web más usadas en la actualidad son PHP, ASP.NET, Ruby on Rails y Java (J2EE), pero los frameworks que tienen más libros publicados son ASP.NET, Ruby on Rails, Frameworks de Java y Frameworks de PHP. Tomando en cuenta esto último, se puede decir que las tecnologías de frameworks más populares son ASP.NET, Ruby on Rails, PHP y Java.

Para los propósitos de este trabajo de tesis, la comparación de dos frameworks Web es suficiente y dado que para comparar dichos frameworks se necesita consultar información confiable, el número de libros publicados es más relevante que el porcentaje de uso de cada tecnología de frameworks Web, por lo que finalmente se determina que los frameworks más adecuados e interesantes para comparar son:

- ASP.NET
- Ruby on Rails

2.2. Identificando criterios para comparar a los frameworks Web

Existen aspectos que muchas veces son considerados únicamente durante el diseño de la arquitectura de una aplicación Web, pero estos aspectos también deben ser tomados en cuenta al seleccionar la tecnología con la que se construirá dicha aplicación, ya que de esta forma la elección de la tecnología contemplará las necesidades de la aplicación. Para el caso de las aplicaciones Web, uno de los enfoques más utilizados en la actualidad para su construcción es con frameworks Web, pero seleccionar el más adecuado puede ser complicado, sobre todo por la gran cantidad que existen de estos. La comparación de este tipo de frameworks sobre aspectos de la arquitectura de la aplicación Web, puede ayudar a determinar cuál o cuáles son los que mejor se ajustan a las necesidades de la aplicación.

En una arquitectura común de aplicaciones Web como la presentada en la sección 1.1.2, algunos de los aspectos principales que se pueden identificar son: el acceso a datos, la seguridad y las pruebas. Otros aspectos interesantes son: la portabilidad y la interoperabilidad.

Para comparar dos o más objetos es habitual que se tenga que abstraerlos, ya sea de manera particular o general. En la siguiente sección se describen algunas características que se encuentran presentes en diversos frameworks Web, las cuales son asociadas con uno de los aspectos previamente reconocidos.

Una forma de verificar el cumplimiento de alguna característica en un framework Web es mediante preguntas de confirmación. De la descripción de cada característica se pueden enunciar varias preguntas para este propósito. En este trabajo dichas preguntas también se utilizarán como criterios para comparar a los frameworks Web.

2.2.1. Portabilidad

En ciertos casos es necesario trasladar una aplicación Web de un hardware, software o un entorno operacional a otro. La portabilidad de una aplicación depende de las opciones relacionadas con la tecnología utilizada para implementar la aplicación, así como de las características de las plataformas en las que se necesita ejecutar dicha aplicación [11].

Desarrollo multiplataforma

La construcción de una aplicación Web generalmente es realizada por un equipo de desarrollo, cuyos integrantes pueden tener la necesidad o disposición de trabajar en distintos sistemas operativos. Cuando un framework Web solo se puede utilizar en un determinado sistema operativo, los integrantes del equipo de desarrollo se ven obligados a trabajar sobre dicho sistema, lo que además limita el software requerido por la aplicación Web, al que es soportado por el sistema operativo en cuestión. En muchos casos tampoco se puede observar el rendimiento de la aplicación Web con otro tipo de software, como navegadores o servidores Web.

Es importante que tanto el framework Web como el software fundamental (IDE, servidor Web, manejador de bases de datos, etc.) para la construcción de una aplicación Web, se puedan utilizar en distintos sistemas operativos. Además el framework Web debe comportarse de manera similar en todos esos sistemas y ser compatible con software de distintos proveedores, que pueda complementar sus características.

En general si con un framework Web es posible crear aplicaciones Web en distintos sistemas operativos, se puede decir que permite un desarrollo multiplataforma.

Algunos criterios de comparación que se pueden obtener de esta característica son los siguientes:

- ¿Los IDE con soporte para el framework están disponibles para distintos sistemas operativos?
- ¿El framework admite el uso de distintos servidores Web (comerciales y no comerciales)?
- ¿Los servidores Web que permite el framework se pueden usar en diversos sistemas operativos?

- ¿El framework ofrece soporte para distintos manejadores de bases de datos (comerciales y no comerciales)?
- ¿Los manejadores de bases de datos que soporta el framework se pueden utilizar en distintos sistemas operativos?
- ¿El software que complementa o extiende la funcionalidad del framework está disponible para diversos sistemas operativos?

2.2.2. Interoperabilidad

En ocasiones se requiere que dos o más aplicaciones Web puedan establecer una comunicación exitosa e intercambiar información. Una aplicación interoperable facilita el intercambio y la reutilización de información de forma interna y externa. Los protocolos de comunicación, interfaces y formatos de datos son las consideraciones clave para la interoperabilidad. Los estándares son otro aspecto importante que se debe tener en cuenta al diseñar una aplicación interoperable [6].

Servicios Web

De acuerdo con el W3C, un Servicio Web es [12]:

"Un sistema de software que permite la interacción máquina a máquina en una red. Tiene una interface descrita en un formato procesable por máquina, específicamente WSDL (Web Service Description Language). Otros sistemas solo pueden interactuar con el servicio Web a través de la forma establecida en su descripción, utilizando mensajes envueltos en SOAP (Simple Object Access Protocol) y normalmente transportados mediante HTTP."

Aunque en la definición anterior se indica tanto el formato para describir un Servicio Web, como el lenguaje y los protocolos de comunicación y transporte para el intercambio de mensajes, esto solo hace referencia a los Servicios Web basados en SOAP. En la actualidad se pueden identificar dos clases de Servicios Web: los Servicios Web basados en SOAP y los Servicios Web basados en REST.

Los Servicios Web basados en REST son llamados de esta forma porque se apoyan en el estilo arquitectónico REST (REpresentational State Transfer). La principal característica del estilo REST es el conjunto de restricciones de diseño que tiene. Estas restricciones se describen a continuación [13]:

- Sin estado. Cada solicitud que el cliente envía al servidor debe contener toda la información necesaria para atender la solicitud.
- Caché. Cada respuesta del servidor debe estar implícita o explícitamente etiquetada como almacenable o no en caché.
- Interface uniforme. La interacción entre el cliente y el servidor se debe ajustar a una interface uniforme. En el contexto del HTTP esto se realiza a través de sus métodos POST, GET, PUT, DELETE, OPTIONS y HEAD.
- Sistema en capas. La arquitectura puede estar compuesta de múltiples capas y cada capa no puede ver más allá de la siguiente capa con la que se comunica.
- Código a solicitud. Con esto se permite que la funcionalidad de un cliente se pueda ampliar mediante la descarga de código adicional. Con respecto a la Web, esto se refiere a tecnologías como Flash, Applets o JavaScript.

Por estas restricciones es que se tiene la necesidad en los Servicios Web REST, de llevar la información de ida y vuelta a través de recursos, que pueden ser descripciones de información que el programador necesita conocer. Los métodos de HTTP que más se utilizan para las solicitudes son: POST, GET, PUT y DELETE. Estos métodos comúnmente son asociados a las operaciones básicas de Creación, Lectura, Actualización y Borrado, también conocidas como operaciones CRUD (Create, Read, Update y Delete), que se pueden aplicar a los recursos de una aplicación.

Actualmente los servicios Web basados en SOAP son un estándar para la interoperabilidad, debido a que han sido ampliamente adoptados por las empresas para apoyar sus procesos internos y proporcionar interacción de negocios, mientras que los los servicios Web REST se ofrecen como una alternativa. Los frameworks Web generalmente proporcionan algún framework especializados, API o enfoque para la creación de estos tipos de Servicios Web.

Los criterios de comparación que se pueden conseguir de esta característica son los siguientes:

- ¿El framework permite describir los datos de un Servicio Web mediante el formato WSDL?
- ¿El framework tiene soporte para el envío de mensajes SOAP?
- ¿El framework proporciona algún framework o API para la creación de Servicios Web basados en SOAP?
- ¿El framework permite crear Servicios Web REST?
- ¿El framework proporciona algún framework o API para la creación de Servicios Web REST?

2.2.3. Acceso a Datos

En la selección de la tecnología adecuada para el acceso a datos en una aplicación Web, se debe considerar el tipo de los datos con los que se está trabajando y la manera en que dichos datos se deben manipular dentro de la aplicación. La tecnología debe proporcionar formas para abstraer los datos y se debe adaptar a las necesidades de la aplicación Web [6].

Soporte para múltiples fuentes de datos

Las aplicaciones Web normalmente utilizan datos que se encuentran almacenados en alguna fuente, que comúnmente es una base de datos. La mayoría de los frameworks Web permiten configurar una base de datos como fuente de datos principal para una aplicación Web, mediante componentes de software de bibliotecas o frameworks especializados para el acceso a datos.

Al ser las bases de datos el tipo de fuente de datos más común para una aplicación Web, se vuelve fundamental que los frameworks Web soporten manejadores de bases de datos de distintos proveedores (tanto comerciales como libres), así como la posibilidad de usar múltiples bases de datos en una misma aplicación. Además los frameworks deben tener soporte para otro tipo de fuentes de datos, tales como archivos planos o XML y proporcionar los mecanismos para acceder a los datos de estas fuentes.

Los criterios de comparación que se pueden apreciar en esta característica son los siguientes:

- ¿El framework permite utilizar bases de datos relacionales como fuente de datos?
- ¿El framework permite utilizar archivos XML como fuente de datos?
- ¿El framework ofrece soporte para otro tipo de fuente, además de bases de datos relacionales y archivos XML?

Soporte de Mapeo Objeto Relacional

Un alto nivel de abstracción de los módulos de una aplicación Web se puede conseguir, si durante su codificación se emplean objetos y atributos en lugar de las filas y campos, de las tablas de una base de datos, además de definir métodos simples en las clases de la aplicación Web, para asociarlos con las sentencias SQL de manipulación de datos, tales como select, insert, update y delete.

Los frameworks Web comúnmente ofrecen herramientas de Mapeo Objeto Relacional (ORM por sus siglas en inglés) para realizar tareas como las que se mencionaron previamente. El Mapeo Objeto Relacional consiste en traducir los sistemas de tipos de una base de datos relacional a los tipos de un lenguaje de programación orientado a objetos, para simplificar las tareas básicas de acceso a los datos [14].

El soporte de ORM debe incluir enfoques para la generación del esquema de una base de datos, a partir de un conjunto de clases con atributos que tienen un tipo de dato específico y viceversa. También se debe proporcionar una definición básica de métodos para las consultas de recuperación, modificación y borrado de datos.

El Mapeo Objeto Relacional permite que el tiempo destinado a la generación de las clases y el esquema de la base de datos, pueda ser empleado en otras necesidades de una aplicación Web. Además posibilita la construcción de un prototipo funcional de una aplicación en poco tiempo y ayuda a que la conversión entre el tipo de dato de los atributos de una clase y el de los campos de una tabla se realice correctamente.

Algunos criterios de comparación que se pueden extraer de esta característica son los siguientes:

- ¿El framework permite generar el esquema de una base de datos a partir de clases?
- ¿El framework permite generar las clases a partir del esquema de una base de datos?

Migraciones para esquemas de bases de datos

Las migraciones son una forma de control de versiones para el esquema de una base de datos. Permiten la modificación del esquema de una base de datos a partir del código de una aplicación Web, lo cual ayuda a mantener la integridad de datos a lo largo de las diferentes versiones de los modelos, que pudieran producirse durante la construcción de una aplicación Web basada en el patrón MVC [15]. Por ejemplo, con las migraciones se puede administrar la forma en que se debe cambiar el tipo de dato de un campo, para una versión posterior del modelo de una aplicación Web y que esto se pueda reflejar en el esquema de la base de datos.

Los criterios de comparación que se pueden conseguir de esta característica son los siguientes:

- ¿El framework permite administrar el esquema de la base de datos desde el código de una aplicación?
- ¿El framework permite automatizar el proceso de una migración?
- ¿El framework proporciona buenas prácticas para el uso de migraciones sobre el esquema de una base de datos?

2.2.4. Seguridad

Las aplicaciones Web deben tener la capacidad de minimizar el impacto de acciones maliciosas o accidentales que afecten su funcionamiento, así como evitar la divulgación o pérdida de información [6]. A nivel arquitectónico la seguridad se puede reducir a entender los requerimientos de seguridad específicos de la aplicación y a crear los mecanismos que puedan soportarlos.

Cuando se reduce la probabilidad de que un ataque tenga éxito y no perjudique el funcionamiento de una aplicación Web, la fiabilidad de dicha aplicación aumenta. En el aseguramiento de una aplicación se debe proteger a sus activos (datos y programas), para evitar que estos no puedan ser leídos o modificados por personas o aplicaciones no autorizadas. Algunos mecanismos utilizados para asegurar las aplicaciones Web son la autenticación, la autorización y el cifrado de los datos [11].

Implementación del proceso de autenticación

En muchas aplicaciones Web se requiere identificar a los usuarios mediante credenciales, para garantizar que los usuarios son quienes dicen ser. En este proceso generalmente se utiliza un nombre de usuario y una contraseña, los cuales son cotejados contra una lista conocida de los mismos [11]. Diseñar una estrategia de autenticación efectiva es importante para la seguridad y confiabilidad de una aplicación. Una autenticación poco adecuada o débil puede provocar que la aplicación sea vulnerable a los ataques de suplantación de identidad.

Actualmente la mayoría de los frameworks Web ofrecen algunas formas de autenticación predefinidas, que pueden ser implementadas en poco tiempo dentro de una aplicación Web. Las implementaciones proporcionadas por los frameworks Web deben ser fáciles de configurar y personalizar, para que el desarrollador pueda tener el mayor control posible sobre el proceso de autenticación.

Algunos criterios de comparación que se pueden obtener de esta característica son los siguientes:

- ¿El framework proporciona implementaciones predefinidas del proceso de autenticación?
- ¿Las implementaciones predefinidas de autenticación se pueden personalizar?

Implementación del proceso de autorización

En las aplicaciones Web a veces es necesario restringir el acceso a ciertos recursos de gran importancia, es decir, que estos recursos solo puedan ser utilizados por aquellos usuarios a los que se les ha concedido autorización para ello, esto generalmente se realiza a través de una lista de control de acceso [11]. Diseñar una buena estrategia de autorización es importante para la seguridad y confiabilidad de una aplicación. Una estrategia de autorización poco adecuada puede causar la divulgación de información, la modificación de datos y que los usuarios consigan privilegios que no les corresponden.

La mayoría de los frameworks Web proporcionan métodos y ejemplos de cómo implementar el proceso de autorización en una aplicación Web. Las formas provistas por un framework Web para implementar el proceso de autorización en una aplicación Web, deben ser fáciles de implementar, modificar y extender.

Los criterios de comparación que se pueden conseguir de esta característica son los siguientes:

- ¿El framework proporciona implementaciones predefinidas del proceso de autorización?
- ¿Las implementaciones predefinidas de autorización se pueden personalizar?

Mecanismos para evitar ataques informáticos comunes

Las aplicaciones Web son atacadas con mucha frecuencia, por lo cual es necesario contar con mecanismos que las protejan de ataques informáticos comunes, tales como ataques de inyección SQL, de inyección de scripts en sitios (XSS) o de falsificación de solicitudes en sitios (CSRF). Estos ataques son descritos brevemente a continuación [16]:

- Ataques de inyección de SQL: Los ataques de inyección SQL modifican las consultas que se realizan sobre una base de datos, manipulando los

parámetros de una aplicación Web. El objetivo popular de los ataques de inyección de SQL es eludir el proceso autorización para leer y manipular los datos de una aplicación Web.

- Ataques de inyección de scripts en sitios (XSS): Estos ataques se aprovechan de una vulnerabilidad de seguridad que permite a un atacante colocar secuencias de comandos (normalmente JavaScript) en páginas Web. Cuando los usuarios cargan las páginas afectadas, los scripts del atacante se ejecutan. Comúnmente estos scripts se utilizan para robar cookies y tokens de sesión, cambiar el contenido de la página Web o para redirigir el navegador a otra página. Las vulnerabilidades aprovechadas por los ataques de inyección XSS ocurren generalmente cuando una aplicación toma los datos de entrada de un usuario y los utiliza para generar una página sin validarla, codificarla o escaparla.
- Ataques de falsificación de solicitudes en sitios (CSRF): En estos ataques un sitio web malicioso puede afectar la interacción entre el navegador de un cliente y una aplicación Web que confía en ese navegador. Estos ataques son posibles porque los navegadores Web envían algunos tipos de tokens de autenticación automáticamente con cada solicitud a una aplicación Web. Esta forma de ataque también se conoce como ataques de un clic, porque el ataque aprovecha la sesión activa de un usuario autenticado, para ejecutar solicitudes en el servidor Web que solo un usuario autenticado podría hacer.

Si bien no existe una forma clara para resolver estos problemas y por lo general se recomienda el uso de buenas prácticas, es necesario que un framework Web proporcione medidas de protección y guías de prevención que puedan ser tomadas en cuenta durante la construcción de una aplicación Web. Estas medidas y guías deben ser claras para poder implementarlas y personalizarlas de acuerdo a las necesidades que se tenga, con el propósito de conseguir un alto nivel de seguridad en una aplicación Web.

Los criterios de comparación que se pueden obtener de esta característica son los siguientes:

- ¿El framework proporciona mecanismos predefinidos para evitar ataques de inyección SQL?
- ¿El framework proporciona mecanismos predefinidos para evitar ataques de inyección de scripts en sitios (XSS)?
- ¿El framework proporciona mecanismos predefinidos para evitar ataques de falsificación de solicitudes en sitios (CSRF)?
- ¿El framework proporciona recomendaciones generales de seguridad?

2.2.5. Capacidad de Pruebas

Las partes de una aplicación Web siempre deben ser probadas y para esto se necesita establecer criterios de prueba y ejecutar las acciones con las que se pueda determinar el cumplimiento de dicho criterios [6].

Las pruebas permiten aislar las fallas de una aplicación de manera oportuna y efectiva. Las decisiones iniciales en el diseño de una aplicación afectan en gran medida la cantidad de casos de prueba requeridos, entre más complejo sea el diseño, será más difícil probarlo a fondo. Generalmente las pruebas son más fáciles de realizar cuando se tiene un diseño sencillo, pero completo. La reutilización de componentes que ya han sido probados reduce el esfuerzo en las pruebas [11]. Estas pruebas al final son efectuadas mediante herramientas, que son proporcionadas por la tecnología con la cual se crea la aplicación. Hoy en día los desarrolladores Web se enfocan principalmente en dos tipos de pruebas: pruebas unitarias y pruebas de integración.

Entornos de ejecución para aplicaciones Web

Es común que los desarrolladores ocupen un único entorno de ejecución para aplicaciones Web sencillas, pero para el caso de aplicaciones más complejas, casi siempre es necesario contar con diferentes entornos de ejecución, por ejemplo, un entorno de desarrollo (donde se hagan las modificaciones), un entorno de pruebas (donde se realicen las pruebas) y un entorno de producción (el que utilizan los usuarios finales).

El uso de diferentes entornos de trabajo para cada fase del proceso de desarrollo de una aplicación Web, facilita la gestión de datos y en general la administración de dicha aplicación. En la actualidad la mayoría de los frameworks Web ofrecen los entornos de ejecución mencionados anteriormente: de desarrollo, pruebas y producción.

Algunos criterios de comparación que se pueden apreciar en esta característica son los siguientes:

- ¿El framework permite la configuración de un entorno de desarrollo?
- ¿El framework permite la configuración de un entorno para realizar pruebas?
- ¿El framework permite la configuración de un entorno de producción?
- ¿El framework permite la configuración de un entorno personalizado?

Soporte para pruebas unitarias

Las pruebas unitarias son las pruebas aplicadas a los componentes de una aplicación de forma aislada, de acuerdo a un criterio previamente establecido que permita a los desarrolladores saber cuándo dejar de hacer dichas pruebas. Los componentes que generalmente se prueban en una aplicación Web son las clases y los métodos [17].

En aplicaciones Web con una arquitectura basada en el patrón MVC, los modelos de dominio son probados para asegurar el funcionamiento adecuado de los objetos de dominio. Existen varias herramientas para realizar pruebas unitarias, pero su uso depende de la tecnología utilizada para la construcción de la aplicación Web.

Antes de escribir una prueba unitaria es necesario identificar un componente lo suficientemente pequeño y razonable para probar, que normalmente no es más grande que una clase y en la mayoría de los casos es un método. Todas las pruebas producidas para las piezas de código identificadas, generan una gran cantidad de pruebas unitarias, las cuales deben ser ejecutadas con frecuencia a medida que la construcción de la aplicación avanza. Si este proceso no es automatizado, puede resultar en una gran pérdida de productividad del desarrollador. Existen frameworks con librerías para pruebas unitarias que permiten automatizar este proceso, con estos frameworks el desarrollador puede escribir las pruebas en un lenguaje de programación dentro de un entorno de desarrollo y enseguida crear un conjunto de reglas de aprobación/falla, con las que el framework determina si la prueba tuvo éxito o no.

Algunos criterios de comparación que se pueden encontrar en esta característica son los siguientes:

- ¿El framework tiene soporte para pruebas unitarias?
- ¿El framework permite automatizar las pruebas unitarias?
- ¿El framework proporciona alguna guía para realizar pruebas unitarias?

Soporte para pruebas de integración

Una vez que las pruebas unitarias se han realizado satisfactoriamente, lo siguiente es probar los componentes de software que interactúan entre sí, agrupándolos en componentes cada vez más grandes y mediante pruebas definidas en un plan de integración. La idea es probar combinaciones de las partes de una aplicación y después expandir este proceso a sus módulos, hasta que eventualmente todos los módulos de la aplicación hayan sido probados juntos. Existen herramientas de terceros que permiten

llevar a cabo este tipo de pruebas, mientras que los frameworks Web comúnmente proporcionan recomendaciones para realizarlas.

Los criterios de comparación que resultan de esta característica son los siguientes:

- ¿El framework tiene soporte para pruebas de integración?
- ¿El framework proporciona alguna guía para realizar pruebas integración?

Depuración de aplicaciones Web

Es importante considerar que el código de una aplicación Web puede contener errores de sintaxis o lógicos, los cuales pueden ser difíciles de diagnosticar. Muchas veces estos errores no son evidentes y no provocan notificaciones o mensajes de error, que puedan servir para buscarlos y corregirlos, por lo que contar con herramientas que ayuden a depurar el código de una aplicación, se vuelve relevante.

La forma en como depurar una aplicación Web depende en gran medida de la tecnología usada para construirla. Generalmente se utiliza un IDE que proporcione o permita el uso de herramientas de depuración para buscar errores en el código, que puedan provocar que la aplicación funcione de forma irregular. Los frameworks Web frecuentemente incluyen mecanismos para inspeccionar variables u objetos a través del mismo lenguaje de programación, utilizado para codificar la aplicación.

Los criterios de comparación que se pueden identificar en esta característica son los siguientes:

- ¿El framework proporciona algún modo para la depuración del código de la aplicación Web?
- ¿El framework permite usar herramientas de terceros para la depuración del código de la aplicación Web?
- ¿El framework proporciona instrucciones para la depuración del código de la aplicación Web?

Capítulo 3

El framework ASP.NET

En este capítulo se da una descripción general de los conceptos, estrategias o tecnologías, que utiliza el framework ASP.NET para proporcionar las características especificadas en la sección 2.2. La información de este capítulo permitirá comparar al framework posteriormente.

3.1. Información general de ASP.NET

ASP.NET (versión 4.5) es un framework Web que forma parte de la plataforma .NET. Ha sido desarrollado y comercializado por Microsoft, y se usa en gran medida para construir sitios, aplicaciones y servicios Web grandes, principalmente con HTML, CSS y JavaScript. El framework Web ASP.NET está compilado en la plataforma .NET, por lo que todas las características de .NET se encuentran disponibles para las aplicaciones ASP.NET [18].

La forma más común de utilizar ASP.NET es a través del IDE Visual Studio. Las aplicaciones ASP.NET se pueden codificar en cualquier lenguaje compatible con el entorno CLR (Common Language Runtime), por ejemplo, Visual Basic o F#, aunque el más usado es C# [19]. Algunas características que se destacan en la documentación de ASP.NET son las siguientes [20]:

- Tres formas de crear aplicaciones ASP.NET: utilizando Formularios Web, basadas en el patrón de diseño MVC y con Páginas Web.
- El compilador ASP.NET.
- Una infraestructura para seguridad.
- Características para el monitoreo del estado y rendimiento de las aplicaciones.
- Entorno ampliable para el alojamiento y administración del ciclo de vida de una aplicación.
- Integración con entornos visuales para desarrolladores y diseñadores como Visual Web Developer o Visual Studio.

- Un framework para la generación de páginas Web con datos dinámicos.

3.2. Portabilidad

3.2.1. Desarrollo multiplataforma

De acuerdo con la documentación del framework, para desarrollar aplicaciones Web con ASP.NET es necesario contar con lo siguiente [21]:

- El framework .NET: Es indispensable tener instalado el framework .NET, debido a que ASP.NET es parte de .NET.
- Un entorno para la construcción de aplicaciones: Si bien es posible crear páginas y clases ASP.NET con algún editor de texto, se recomienda utilizar un IDE que proporcione características como plantillas, autocompletado, compilación en tiempo de diseño, entre otras. La construcción de aplicaciones ASP.NET generalmente requiere del IDE Microsoft Visual Studio, porque este complementa y agrega nuevas características al framework ASP.NET.
- Un servidor Web: Contar con un servidor Web es esencial porque es el software que permite el despliegue de una aplicación Web y proporciona las herramientas necesarias para dar respuesta a las solicitudes realizadas desde un navegador Web. Para aplicaciones ASP.NET se recomienda el servidor Microsoft IIS (Internet Information Services).

Otro tipo de software que frecuentemente es requerido en el desarrollo de una aplicación ASP.NET es:

- Un manejador de bases de datos: Las aplicaciones Web que necesitan almacenar datos, requieren de permisos adecuados para la lectura y escritura de datos en una fuente de datos, que generalmente es una base de datos. Microsoft SQL Server es el manejador de bases de datos recomendado para aplicaciones ASP.NET.
- Un servidor de correo electrónico: Cuando se requiere enviar correos electrónicos desde una aplicación Web es necesario contar con un servidor SMTP (Simple Mail Transfer Protocol).

De lo anterior se puede concluir que para aprovechar ASP.NET es necesario utilizar software de Microsoft, como el IDE Visual Studio y el servidor Web IIS, e incluso SQL Server como manejador de bases de datos. Además este software solo está disponible para sistemas operativos Windows, lo que finalmente condiciona el uso de ASP.NET a estos sistemas y por lo cual se puede decir que ASP.NET no permite un desarrollo multiplataforma.

Actualmente existe una variante de ASP.NET conocida como ASP.NET Core, que se diseñó para permitir un desarrollo multiplataforma, pero aún se encuentra en desarrollo y no ha sido adoptada de buena forma por la comunidad. El proyecto denominado Mono contiene una implementación de ASP.NET y permite usar el framework en sistemas Linux, OpenBSD y Solaris, pero no tiene soporte de Microsoft dado que su desarrollo y mantenimiento es realizado por un tercero, además su configuración es complicada y varias características son afectadas por los problemas de compatibilidad con estos sistemas.

3.3. Interoperabilidad

3.3.1. Servicios Web

Windows Communication Foundation (WCF) es el framework recomendado para la construcción de Servicios Web en aplicaciones ASP.NET. WCF permite la creación de aplicaciones orientadas a servicios y forma parte de la plataforma .NET. Con WCF los datos se pueden enviar como mensajes asíncronos desde el punto final de un servicio a otro. Un punto final puede ser un cliente de un servicio, que solicite datos desde el punto final del servicio. El intercambio de datos se puede realizar de forma local en una sola computadora, dentro de una red corporativa, o en el Internet. Los mensajes pueden ser tan simples como un solo carácter o una palabra y enviarse como XML, o tan complejos como un flujo de datos binarios. Las características proporcionadas por WCF son las siguientes [23]:

- **Orientación al servicio**. Una consecuencia del uso de estándares de Servicios Web es que WCF permite crear aplicaciones orientadas a servicios. Los servicios tienen la ventaja general de estar débilmente acoplados en lugar de ser codificados en cada aplicación. Una relación débilmente acoplada implica que cualquier cliente creado en alguna plataforma, puede conectarse a cualquier servicio siempre y cuando se cumplan los contratos esenciales.
- **Múltiples patrones de mensajes**. El patrón más común para el intercambio de mensajes es el de solicitud/respuesta, en donde un punto final solicita

datos a un segundo punto final, que finalmente responde. Otro patrón es el de mensaje unidireccional, en donde un punto final único envía un mensaje sin ninguna expectativa de respuesta.

- **Metadatos de servicio**. WCF permite la publicación de los metadatos de un servicio mediante formatos especificados en estándares de la industria, como WSDL, XML Schema y WS-Policy. Los metadatos se pueden emplear para generar y configurar los clientes que puedan acceder a los servicios Web. Estos metadatos se pueden publicar a través de HTTP y HTTPS.
- **Contratos de datos**. Dado que WCF está construido sobre .NET, se pueden utilizar métodos de un lenguaje de programación para suministrar los contratos que se requieran. Al codificar un Servicio Web mediante C# o Visual Basic, la forma más fácil de manejar los datos es creando clases que representen a las entidades de datos. WCF permite crear estas clases y generar automáticamente los metadatos que permitan a los clientes cumplir con los tipos de datos que se han especificado para el Servicio Web.
- **Seguridad**. Los mensajes intercambiados pueden ser cifrados y configurados para pedir que los usuarios se autentiquen antes de que puedan recibir los mensajes. La seguridad se puede implementar utilizando estándares conocidos, como SSL o WS-SecureConversation.
- **Múltiples codificaciones y transportes**. Los mensajes son enviados a través de un protocolo de transporte y codificación. WCF permite enviar mensajes codificados en XML utilizando SOAP sobre HTTP, para su uso en la Web. Como alternativa ofrece el envío de mensajes en cola a través de TCP, que también es conocido como Message Queuing (MSMQ).
- **Mensajes confiables y en cola**. WCF utiliza sesiones implementadas a través de la especificación WS-Reliable Messaging y la implementación MSMQ para lograr que el intercambio de mensajes sea confiable.
- **Mensajes duraderos**. Un mensaje duradero es aquel que no se pierde a pesar de una interrupción en la comunicación. Los mensajes intercambiados mediante un patrón de mensaje duradero siempre son guardados en una base de datos. Si se produce una interrupción, el intercambio de mensajes se puede reanudar en cuanto la conexión se restaure.
- **Soporte REST**. En WCF se pueden construir servicios Web basados en el estilo arquitectural REST, así como procesar datos XML sin formato que no tengan que ser envueltos sobre SOAP. WCF soporta formatos XML específicos, como ATOM (un estándar RSS muy popular) y también formatos no XML, como JSON (JavaScript Object Notation).

La publicación de servicios mediante estándares como WSDL, XML-Schema y WS-Policy y el intercambio de mensajes a través de SOAP sobre HTTP, hacen que WCF sea ideal para la construcción de Servicios Web basados en SOAP. A pesar de que también es posible crear Servicios Web REST, esta no es la mejor opción, debido a la cantidad y la complejidad de los archivos que deben configurarse para adaptarlo.

3.4. Acceso a Datos

3.4.1. Soporte para múltiples fuentes de datos

ASP.NET incluye varios controles para la fuente de datos de una aplicación Web, a través de los cuales se pueden usar diferentes tipos de fuentes de datos, tales como una base de datos, un archivo XML o un objeto de negocio de capa media. Estos controles se utilizan para establecer la conexión con la fuente de datos y para recuperar los datos de dicha fuente, con el propósito de que estén disponibles para otros controles con los cuales tengan comunicación. En la siguiente tabla se describen los controles para los tipos de fuentes de datos soportadas en ASP.NET [24]:

Tabla 3.1: Controles para los tipos de fuentes de datos soportadas en ASP.NET

Tipo de fuente de datos soportada	Control de fuente de datos	Descripción
Base de datos	**SqlDataSource**	Permite trabajar con bases de datos SQL Server, OLE DB (Object Linking and Embedding for Databases), ODBC (Open DataBase Connectivity) y Oracle. Cuando se utiliza con SQL Server, proporciona capacidades avanzadas de almacenamiento en caché.
	EntityDataSource	Permite vincular los datos asociados a un Modelo de Datos de Entidad (EDM por sus siglas en inglés).
	AccessDataSource	Admite el uso de una base de datos de Microsoft Access.
Base de datos o archivo XML	**LinqDataSource**	Permite utilizar LINQ (Language-Integrated Query) en páginas Web ASP.NET para recuperar y modificar datos de un objeto.

Archivo XML	**XmlDataSource**	Facilita el trabajo con un archivo XML, especialmente para mostrar datos en forma jerárquica. Se pueden utilizar expresiones XPath (XML Path Language) para realizar consultar y aplicar una transformación XSL (Extensible Stylesheet Language) a los datos.
Objeto de negocio de capa media	**ObjectDataSource**	Admite el uso de un objeto de negocio o de otra clase, para crear aplicaciones Web que dependan de objetos de capa media para administrar los datos.

En lo que se refiere a bases de datos, ADO.NET es la tecnología que se utiliza en el código de una aplicación ASP.NET para realizar la conexión con una base de datos. ADO.NET logra la comunicación con el sistema manejador de bases de datos (DBMS) mediante un software proveedor de datos. Los proveedores de datos de Microsoft permiten establecer una conexión con bases de datos de los siguientes manejadores [25]:

- SQL Server en todas sus versiones, incluyendo SQL Server Express, LocalDB, SQL Server Compact.
- Cualquier base de datos de un manejador que admita ODBC u OLEDB.

También es posible utilizar proveedores de datos de terceros, como MySQL, SQLite, Oracle y DB2, pero con estos no se tienen características de Mapeo Objeto Relacional.

3.4.2. Soporte de Mapeo Objeto Relacional

La lectura o actualización de datos en una aplicación ASP.NET se puede realizar directamente con ADO.NET, escribiendo y ejecutando directamente las consultas SQL, así como el código necesario para convertir los datos del formato de la base datos a objetos, propiedades y colecciones con las que se puede trabajar en el código de la aplicación. Dado que esto no es la mejor opción, ASP.NET ofrece como alternativa el framework Entity (EF por sus siglas en inglés), que brinda soporte para el Mapeo Objeto Relacional, el cual permite que el manejo del código de bajo nivel que interactúa con el proveedor de datos ADO.NET, sea transparente al codificar la aplicación Web [25].

El framework Entity es un conjunto de tecnologías que soportan el desarrollo de aplicaciones Web orientadas a datos y que se puede usar a través del IDE Visual Studio. Para usar el framework Entity en una aplicación Web se requiere de un modelo de datos de entidad, que se compone por un modelo conceptual, un modelo de almacenamiento y una especificación de mapeo entre estos modelos. A continuación se describen las características y componentes del framework Entity [26]:

- El Modelo de Datos de Entidad (EDM por sus siglas en inglés) es la pieza central del framework Entity. Contiene el modelo conceptual que define las entidades y las relaciones, específica el modelo de almacenamiento donde se definen las estructuras de almacenamiento para la persistencia de datos e incluye una especificación de mapeo que conecta el modelo conceptual con el modelo de almacenamiento.
- El componente Servicios de Objeto es un punto de entrada para acceder a los datos de una fuente de datos y devolverlos. Este componente es responsable de la materialización, que es el proceso de convertir los datos devueltos por un proveedor EntityClient en una estructura de objeto de entidad.
- LINQ to Entities permite usar LINQ, que es un lenguaje utilizado para escribir consultas sobre el modelo de objetos. Devuelve entidades definidas en el modelo conceptual. Las consultas son escritas en alguno de los lenguajes de programación del Framework .NET, como Visual Basic o C#.
- El proveedor de datos EntityClient extiende el modelo del proveedor de ADO.NET para acceder a los datos de las entidades y relaciones conceptuales. Convierte una consulta Entity SQL en una que sea entendible para la base de datos.
- Entity SQL es un lenguaje de consulta independiente del almacenamiento, es similar a SQL y permite que EntityClient pueda comunicarse con la base de datos.
- El componente de metadatos de ADO.NET administra los metadatos asociados con los modelos y el mapeo. El mecanismo de almacenamiento utiliza un archivo con tres dialectos XML: el Lenguaje de Definición del Modelo Conceptual (CSDL por sus siglas en inglés), el Lenguaje de Definición del Modelo de Almacenamiento (SSDL por sus siglas en inglés) y el Lenguaje de Especificación de Mapeo (MSL por sus siglas en inglés).

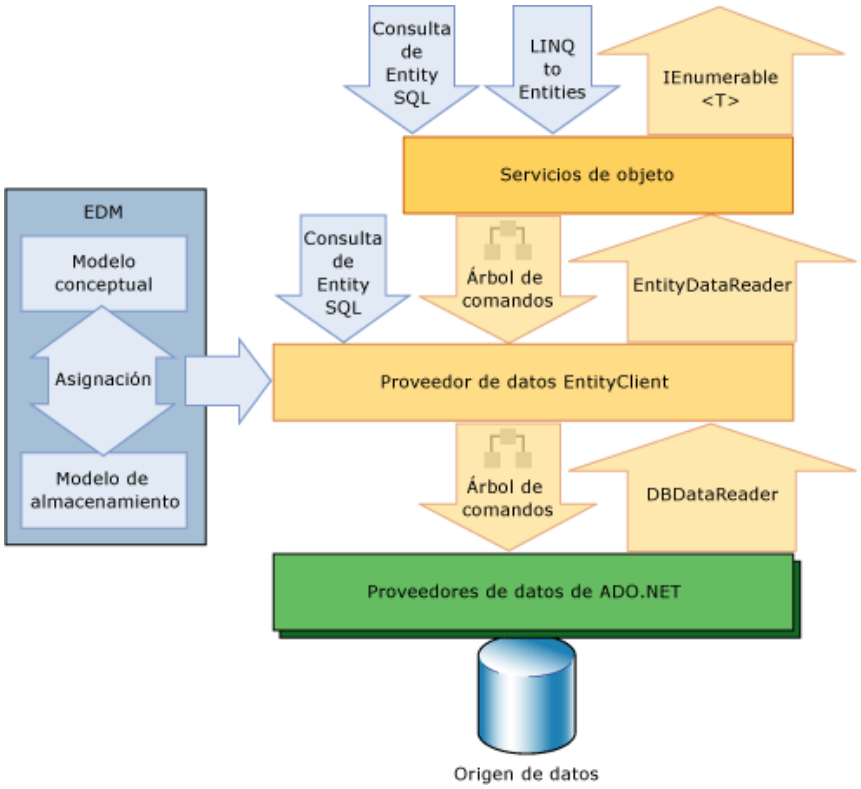

Figura 3.1: Arquitectura del framework Entity

La figura 3.1 muestra cómo se relacionan las interfaces de programación del framework Entity. Una flecha descendente indica una consulta sobre la fuente de datos, y una flecha ascendente indica los datos devueltos. El componente Servicios de Objeto genera un árbol de comandos canónico que representa una consulta LINQ to Entities o Entity SQL sobre el modelo conceptual. El proveedor EntityClient transforma el árbol de comandos basado en el EDM, en un nuevo árbol de comandos canónico, que no es más que una consulta SQL sobre la fuente de datos.

Al usar el framework Entity es posible trabajar los modelos de datos y las bases de datos bajo tres enfoques [26]:

- **Database First**. En este enfoque el modelo de datos se genera primero a partir de una base de datos existente, es decir, se aplica un proceso de ingeniería inversa para crear el modelo. El modelo de datos obtenido es almacenado en un archivo con extensión edmx. Este modelo se componen de entidades y propiedades, que corresponden a las tablas y columnas de la base de datos que se utilizó.

- **Model First**. En este enfoque los modelos de datos se crean primero mediante un entorno gráfico (diseñador del framework Entity) de Visual Studio. Cuando los modelos son terminados, se pueden generar las sentencias SQL del lenguaje de definición de datos (DDL por sus siglas en inglés), para crear la base de datos. El modelo de datos es almacenado en un archivo con extensión edmx.
- **Code First**. Ya sea que exista una base de datos o no, es posible emplear el framework Entity sin la necesidad de usar el diseñador para crear un archivo edmx. En el caso de que la base de datos no exista, se pueden codificar las clases con sus respectivos atributos, y a partir de estas crear las correspondientes tablas y columnas de la base de datos de la aplicación. Si la base de datos existe, las herramientas del framework Entity pueden generar las clases y atributos que correspondan a las tablas y columnas de la base existente.

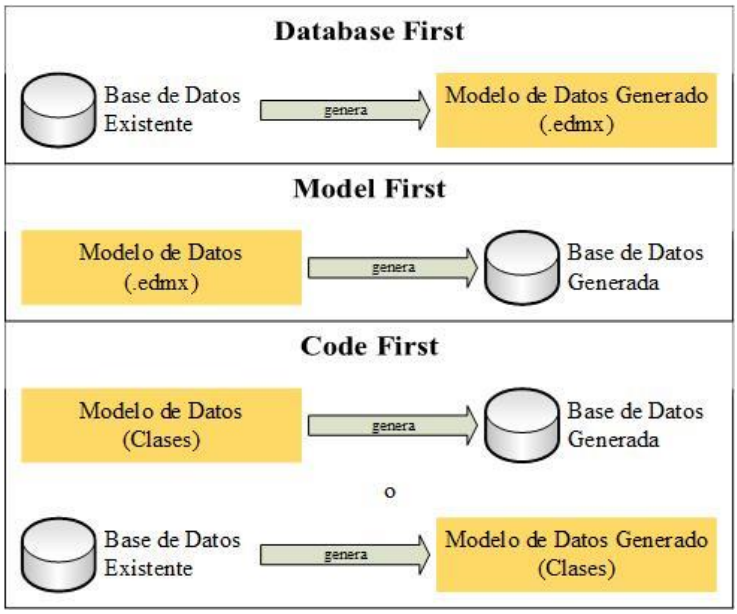

Figura 3.2: Enfoques para trabajar los Modelos de Datos y las Bases de Datos con el framework Entity

La diferencia entre los tres enfoques es que al utilizar Database First o Model First, los modelos de datos se crean en un entorno gráfico, mientras que en el enfoque Code First, las clases de los modelos son codificados con un lenguaje de programación, que generalmente es C#. Aunque Database First o Model First agilizan la construcción de una aplicación, es importante considerar como se desea administrar los cambios en los modelos de datos, después de crear la base de datos.

3.4.3. Migraciones para esquemas de bases de datos

Durante la construcción de una aplicación ASP.NET, las migraciones pueden usarse a través del enfoque Code First del framework Entity. Si la base de datos para una aplicación Web es creada mediante Code First, se puede usar Code First Migrations para automatizar el proceso de implementación de dicha base de datos, en un entorno de producción [25].

Las migraciones permiten que los cambios realizados en un modelo de datos se puedan aplicar en el esquema de la base de datos en producción. Por ejemplo, cuando se agrega una nueva propiedad a un modelo de datos, casi siempre es necesario crear su correspondiente campo en la tabla de la base de datos. Esto se puede realizar con migraciones de la siguiente forma [27]:

1. Lo primero es habilitar las migraciones dentro del contexto del proyecto de Visual Studio de la aplicación Web, que contiene el modelo de datos modificado. Para habilitar las migraciones es necesario ejecutar el comando `enable-migrations` en la consola de administración de paquetes del IDE, lo cual creara una carpeta llamada Migrations con dos archivos:

    a. Una clase de configuración, que permite configurar cómo se deben comportan las migraciones en el contexto del proyecto de la aplicación Web.

    b. Una migración llamada InitialCreate, con la cual se crea automáticamente una base de datos con las tablas correspondientes a los modelos definidos de forma inicial para la aplicación Web.

2. Y para que la modificación en el modelo de datos se pueda reflejar en la correspondiente tabla de la base de datos, se necesita hacer lo siguiente:

    a. Ejecutar el comando `Add-Migration` seguido de un nombre descriptivo para el archivo de migración que se creara en la carpeta Migrations, en dicho archivo se especifica la forma en que el cambio será efectuado. Por ejemplo, si se agrega una propiedad Url a un modelo llamado Blog, el comando podría verse de la siguiente forma:

    ```
    Add-Migration AgregaUrlABlog
    ```

    b. Y ejecutar el comando `Update-Database` para efectuar los cambios especificados en el archivo de migración.

3.5. Seguridad

3.5.1. Implementación del proceso de autenticación

La autenticación para aplicaciones ASP.NET es implementada a través de proveedores de autenticación, que son módulos que contienen el código necesario para autenticar las credenciales de un solicitante [28]. Para necesidades de autenticación simples de una aplicación Web existen dos modos de autenticación predefinidos: la autenticación de Windows y la autenticación mediante Formularios, aunque también es posible indicar que la aplicación Web no utilizará un modo de autenticación. En la siguiente tabla se describen los modos de autenticación predefinidos [29]:

Tabla 3.2: Modos de autenticación predefinidos para aplicaciones ASP.NET

Modo de autenticación	Descripción
Windows	La configuración que debe realizarse para utilizar el modo Windows en una aplicación Web en general es muy poca, debido a que las credenciales (nombres y contraseñas de usuario) que se puedan usar son las del sistema operativo Windows que se utiliza como servidor, mientras que la responsabilidad del proceso de autenticación se delega al servidor Web IIS, por lo que su uso es indispensable. Este modo es una opción conveniente para aplicaciones de Intranet. Este modo se establece en el archivo de configuración (web.config) de la aplicación Web y además se debe habilitar la característica de autenticación mediante Windows en el servidor IIS.
Forms	Para utilizar el modo de autenticación mediante formularios en una aplicación Web, primero se debe crear la página de inicio de sesión con un formulario, para recopilar las credenciales de autenticación del usuario (por ejemplo, su nombre y contraseña) y después se debe escribir el código para el proceso de autenticación. La autenticación con formularios en su forma más simple se establece en el archivo de configuración (web.config) de la aplicación Web, indicando el modo de autenticación Forms y la página que contiene el formulario de inicio de sesión.

	Una forma conveniente de utilizar la autenticación mediante formularios es junto a ASP.NET membership, ya que con este proveedor se pueden almacenar los datos de los usuarios (nombres, contraseñas y direcciones de correo electrónico) en una base de datos de SQL Server, además proporciona métodos predefinidos para validar las credenciales y para administrar los datos de los usuarios.

En resumen, el modo Windows no es posible adaptarlo para cubrir otro tipo de necesidades de autenticación, debido a que el servidos IIS es el encargado de realizar el proceso, utilizando para este propósito, los tipos de credenciales (nombre y contraseña de un usuario o certificados) soportados por un sistema operativo Windows. En cambio con el modo Forms existe la posibilidad de usar otro tipo de credenciales para la autenticación en una aplicación Web, ya que el desarrollador es el responsable de implementar este proceso.

Aunque lo recomendable es usar los modos de autenticación Windows y Forms para aplicaciones de Intranet e Internet, respectivamente, es importante saber que el desarrollador puede crear nuevos mecanismos de autenticación, que le permitan tener un mayor control sobre este proceso y con el que se puedan cubrir mejor las necesidades de la aplicación Web.

3.5.2. Implementación del proceso de autorización

Después de que un usuario se ha autenticado dentro una aplicación Web, lo siguiente es determinar que recursos puede acceder dicho usuario. En ASP.NET existen dos formas predefinidas para autorizar el acceso a un recurso determinado: la autorización mediante archivo y la autorización mediante el URL. En la tabla 3.3 se describen las formas de autorización disponibles para aplicaciones ASP.NET [30]:

Tabla 3.3: Formas de autorización predefinidas para aplicaciones ASP.NET

Forma de autorización	Descripción
Autorización mediante archivo	La autorización mediante archivo se realiza a través de un módulo (`FileAuthorizationModule`), que proporciona servicios de autorización contra listas de control de acceso (ACL, por sus siglas en inglés) del sistema de archivos de un sistema operativo Windows, por lo cual esta forma de autorización solo se puede utilizar cuando el modo de autenticación es el de Windows. En esta forma de

	autorización, las listas de control de acceso son revisadas para determinar si un usuario autenticado puede acceder a un recurso determinado.
Autorización mediante el URL	La autorización mediante el URL se realiza con un módulo (`UrlAuthorizationModule`), que permite mapear usuarios y roles con determinados URL dentro de aplicaciones ASP.NET. Este módulo se utiliza para permitir o negar selectivamente el acceso a partes arbitrarias de una aplicación Web (normalmente directorios) para usuarios o roles específicos. Con la autorización mediante el URL, se concede o se niega explícitamente el acceso a un directorio, de acuerdo al nombre de usuario o una lista de roles de la que pueda ser miembro el usuario. Los permisos de acceso a un directorio determinado se especifican en su archivo de configuración. En este archivo es posible definir una sección llamada `authorization` con los subelementos `allow` y `deny`, para permitir o negar el acceso a ciertos usuarios o roles. Los subelementos `allow` y `deny` son interpretados de acuerdo al orden en que aparecen en la configuración. Los permisos especificados para un directorio también se aplican a sus subdirectorios, a menos que los permisos del archivo de configuración de un subdirectorio los sobrescriba.

3.5.3. Mecanismos para evitar ataques informáticos comunes

ASP.NET proporciona algunas recomendaciones y mecanismos para la prevención de los siguientes ataques en contra de aplicaciones Web:

- Ataques de inyección SQL.
- Ataques de inyección de scripts en sitios (XSS).
- Ataques de falsificación de solicitudes en sitios (CSRF).

Prevención de ataques XSS

El ataque XSS en un nivel básico engaña a una aplicación Web para poder insertar una etiqueta `<script>` en una de las páginas que se puedan generar o bien agregar un evento `On*` en un elemento del HTML. Con el fin de prevenir ataques XSS en una aplicación Web, en la documentación de ASP.NET se recomienda seguir los siguientes pasos durante su construcción [31]:

1. Nunca colocar datos no confiables en los elementos del HTML, a menos que se realicen el resto de los pasos descritos. Los datos no confiables son aquellos que pueden ser controlados por un atacante, como los campos de entrada de los formularios del HTML, cadenas de consultas, cabeceras HTTP, incluso los datos procedentes de una base de datos pueden ser utilizados por un atacante para alterar la base de datos de la aplicación.
2. Codificar en HTML los datos no confiables, que sean colocados dentro de un elemento o un atributo de un elemento del HTML. Por ejemplo, el carácter < debe ser escrito como `<` para que el elemento del HTML que lo contenga sea generado correctamente.
3. Si es necesario colocar datos no confiables en el código JavaScript, se recomienda colocarlos a través de un elemento del HTML cuyo contenido se obtenga en tiempo de ejecución. Si esto no es posible, es necesario que los datos estén codificados con JavaScript. La codificación JavaScript toma caracteres peligrosos y los reemplaza con su código hexadecimal, por ejemplo, el carácter < se codifica como `\u003C`.
4. Una cadena de consulta URL que contenga datos no confiables debe ser codificada como un URL.

Estos pasos pueden ser implementados en aplicaciones ASP.NET MVC a través del motor Razor, que proporciona herramientas para codificar automáticamente todos los datos procedentes de variables, que sean colocadas dentro de un bloque de código Razor. Un bloque de código Razor comienza con un `@` y está encerrado entre `{}`. Estos bloques no se pueden utilizar para datos de variables del lenguaje JavaScript (es decir, un contexto JavaScript), solo se pueden usar en un contexto del HTML, debido a que a los datos de las variables que son colocadas dentro de un bloque de código Razor, se les aplica las reglas de codificación de atributos del HTML.

Prevención de ataques CSRF

La forma de prevenir ataques CSRF en aplicaciones ASP.NET MVC es mediante tokens anti-falsificación, también llamados tokens para la verificación de solicitudes. Este mecanismo de prevención funciona de la siguiente manera [32]:

1. El cliente solicita una página HTML que contiene un formulario.

2. El servidor incluye dos tokens en la respuesta. Un token se envía como una cookie y el otro se coloca en un campo oculto del formulario. Los tokens se generan aleatoriamente para que un atacante no pueda adivinar estos valores.
3. Cuando el cliente envía el formulario, también debe enviar ambos tokens de nuevo al servidor. El cliente envía la cookie que contiene uno de los tokens y el otro como parte de los datos del formulario.
4. Si una solicitud no incluye estos tokens, el servidor simplemente rechaza dicha solicitud.

3.6. Capacidad de Pruebas

3.6.1. Entornos de ejecución para aplicaciones Web

Las aplicaciones ASP.NET pueden ser ejecutadas bajo los entornos de Desarrollo, Preproducción y Producción. En la siguiente tabla se describen estos entornos de ejecución [22]:

Tabla 3.4: Entornos de ejecución predefinidos para aplicaciones ASP.NET

Entorno de ejecución	Descripción
Desarrollo	Permite definir un entorno típico para el desarrollo una aplicación Web. Se suele emplear para usar funciones no deseables en un entorno de producción, como la página donde se muestran las excepciones generadas por la aplicación.
Preproducción	Permite definir un entorno de preproducción, utilizado para las pruebas finales antes de la implementación en la producción. Idealmente, sus características deben reflejar las de un entorno de producción, para que cualquier problema que pueda surgir en producción ocurra primero en el entorno de preproducción y además no tenga impacto en los usuarios.
Producción	Permite configurar el entorno para que una aplicación Web sea utilizada por los usuarios finales. Este entorno debe configurarse para maximizar la seguridad, el rendimiento y la robustez de la aplicación Web. Algunas recomendaciones para este entorno son las siguientes:

	• Activar el almacenamiento en caché. • Desactivar las páginas de error de diagnóstico. • Activar las páginas de error amigables. • Habilitar el registro y el monitoreo de producción.

3.6.2. Soporte para pruebas unitarias

En ASP.NET el soporte para pruebas unitarias se ofrece principalmente para las aplicaciones Web basadas en el patrón MVC, debido a la separación de componentes que se logra con una arquitectura basada en este patrón.

La herramienta más utilizada para crear pruebas unitarias en aplicaciones ASP.NET MVC es el asistente gráfico de Visual Studio, con el cual se pueden generar clases de pruebas unitarias, tanto para los métodos y acciones de un modelo como de un controlador determinado. El asistente forma parte del framework para pruebas unitarias denominado Visual Studio Unit Test, que está incluido en prácticamente todas las ediciones de Visual Studio. A pesar de que este framework es la herramienta predeterminada para administrar las pruebas unitarias, estas pruebas también se pueden crear con frameworks de terceros, como NUnit, MbUnit o XUnit, y con bibliotecas, como Rhino Mocks, Type Mocks o NMock.

En las pruebas unitarias comúnmente se invoca de forma directa a los métodos de las clases de los controladores de una aplicación. En un contexto de aplicaciones ASP.NET MVC al invocar dentro de una prueba unitaria a una acción de un controlador, se recomienda validar que la vista presentada sea la correcta (aunque esto no significa que el código HTML también es verificado) y que se devuelvan los datos que la vista requiere. También se recomienda probar que la redirección a otro controlador o vista se haga correctamente [33].

3.6.3. Soporte para pruebas de integración

El enfoque utilizado para las pruebas de integración en aplicaciones ASP.NET MVC, es mediante la automatización de un navegador Web para reproducir las acciones que un usuario haría, como pulsar botones, enlaces o enviar formularios.

Dos opciones de código abierto para automatizar a un navegador Web, que pueden usar los desarrolladores ASP.NET son [34]:

- Selenium RC (Remote Control): es una herramienta robusta y madura, que se compone de una aplicación Java, con la que se pueden enviar comandos para automatizar las solicitudes en navegadores Web como Internet Explorer, Mozilla Firefox y Safari, además de clientes para .NET, Python y Ruby, lo que permite crear scripts de prueba en los lenguajes soportados por estas plataformas.
- WatiN: es una biblioteca de .NET que puede enviar comandos para automatizar los navegadores Web Internet Explorer y Mozilla Firefox. Su API no es tan robusta como Selenium, pero maneja adecuadamente los escenarios comunes y es fácil de configurar, solo necesita agregar la referencia a la DLL en la aplicación ASP.NET.

En la literatura de ASP.NET prácticamente no hay información sobre pruebas de integración, debido a que el framework fue diseñado para crear y ejecutar pruebas unitarias de forma sencilla en aplicaciones ASP.NET MVC.

3.6.4. Depuración de aplicaciones Web

La depuración de aplicaciones ASP.NET se apoya fundamentalmente en las herramientas que proporciona Visual Studio, entre las que se encuentran un depurador para examinar aplicaciones Web mientras se están ejecutando. Algunas de las funcionalidades que tiene este depurador son las siguientes [35]:

- Puntos de interrupción: con estos puntos se indican los lugares en el código donde el depurador detendrá la aplicación, para ver el estado de datos actual de la aplicación y, a continuación, pasar por cada línea de código.
- Revisión a pasos (Stepping): después de que una aplicación se ha detenido en un punto de interrupción, el código se puede ejecutar línea por línea, lo cual se conoce como revisión a pasos del código.
- Visualización de datos: el depurador ofrece varias opciones para visualizar y rastrear los datos mientras una aplicación se está ejecutando. El depurador permite modificar los datos mientras una aplicación se encuentra en modo de interrupción, y que al continuar su ejecución pueda utilizar estos datos modificados.

Algunos consejos para la depuración de aplicaciones ASP.NET son:

- Nunca colocar el valor `true` a la variable `debug` en el archivo de configuración de una aplicación Web, para un entorno de producción.
- No dejar vacío el bloque `Catch` (del lenguaje C#) con el fin de evitar las excepciones que pueda producir un código envuelto en un bloque `Try` dentro de una aplicación. Aunque esto evita que las excepciones aparezcan en las páginas de la aplicación, también provoca que la depuración sea más complicada, debido a que ya no se tiene conocimiento de que ha ocurrido un problema.
- Evitar capturar excepciones genéricas y ser lo más explícito posible con los tipos de excepción en la sentencia `Catch` de un bloque `Try`/`Catch`.

Capítulo 4

El framework Ruby on Rails

En el presente capítulo se ofrece una descripción general de los conceptos, estrategias o tecnologías, que utiliza el framework Ruby on Rails para proporcionar las características especificadas en la sección 2.2. La información de este capítulo servirá para comparar al framework posteriormente.

4.1. Información general de Ruby on Rails

Ruby on Rails o simplemente Rails (versión 5.0) es un framework Web de código abierto, que fue diseñado principalmente por David Heinemeier Hansson y que tiene como objetivo fundamental, el desarrollo ágil de aplicaciones Web.

La construcción de aplicaciones Web con Ruby on Rails se basa en dos principios fundamentales: convención sobre configuración y no repetirse a sí mismo. El primero de estos principios se refiere a permitir que el desarrollador se ocupe lo menos posible de aspectos de configuración generales para que pueda enfocarse en las verdaderas necesidades de la aplicación, mientras que el segundo alude a que la repetición de código debe ser evitada. Rails fue escrito en Ruby, debido a que este lenguaje de programación permite el logro de la mayor parte de sus objetivos gracias a su sintaxis abreviada y su naturaleza dinámica [36].

La arquitectura de una aplicación Rails está basada en el patrón de diseño MVC, es decir, está compuesta por tres capas, que tienen una responsabilidad específica [37].

La capa Vista se compone de plantillas, que son las encargadas de representar apropiadamente los recursos de una aplicación. Las plantillas pueden tener distintos formatos, pero la mayoría de estas plantillas son código HTML con código Ruby incrustado (archivos .erb).

La capa Modelo representa el modelo de dominio, que encapsula la lógica de negocio de la aplicación. Aunque la mayoría de los modelos Rails representan a una de

las tablas de una base de datos, los modelos también pueden ser clases ordinarias Ruby con implementaciones de interfaces.

La capa Controlador es la responsable de administrar las solicitudes HTTP y dar una respuesta adecuada, lo que generalmente significa responder código HTML, aunque los controladores de Rails también pueden generar XML, JSON, PDF, vistas específicas para dispositivos móviles y otros formatos. Los controladores manipulan los modelos y procesan las plantillas para generar una respuesta apropiada.

Ruby on Rails ofrece un conjunto de funcionalidades como la persistencia de base de datos independiente del proveedor, la generación automática de código para operaciones como la inserción, lectura, actualización y borrado de la información.

4.2. Portabilidad

4.2.1. Desarrollo multiplataforma

En la documentación del framework Ruby on Rails se indica que para comenzar con la construcción de una aplicación Web es necesario contar con lo siguiente [38]:

- El lenguaje Ruby: Este lenguaje debe estar disponible en el sistema operativo para que el código de las aplicaciones Web pueda ser interpretado. La versión del framework Ruby on Rails utilizada o que se pretenda utilizar, determinara cual es la versión apropiada del lenguaje que se necesita tener instalada.
- El gestor de paquetes RubyGems: Generalmente se incluye con el lenguaje Ruby.
- El framework Ruby on Rails: El framework puede ser instalado como una gema, utilizando el gestor de paquetes RubyGems. Cuando el desarrollo se realiza en sistemas operativos Windows es necesario identificar la versión más adecuada para estos sistemas.
- Un servidor Web: Para poder desplegar las aplicaciones Web es indispensable tener un servidor Web. Actualmente el servidor Puma es el que se distribuye por defecto con el framework Web, pero tiene soporte para otros, como Phusion Passenger.
- Una instalación funcional del manejador de bases de datos SQLite3: Es necesario contar con este manejador, debido a que está configurado por

defecto en los proyectos generados con Ruby on Rails, aunque las aplicaciones Rails también pueden utilizar otros manejadores (ver sección 4.4.1).

- Software para creación de código: El código de una aplicación Web debe ser escrito mediante un software para dicho propósito, como un editor de texto o un IDE. Actualmente la mayoría de los editores de texto reconocen el código Ruby y entre los IDE con soporte para Rails, ya sea de manera parcial o completa están Ruby Mine, Aptana y NetBeans.

Dado que el lenguaje Ruby, el servidor Web (Puma) y el manejador de bases de datos (SQLite) recomendados se pueden instalar y configurar en distintos sistemas operativos, Ruby on Rails sí permite un desarrollo multiplataforma, así como por el hecho de que tiene soporte para varios servidores Web y manejadores de bases de datos, y que no depende significativamente de un editor de texto o IDE específico.

4.3. Interoperabilidad

4.3.1. Servicios Web

Ruby on Rails no proporciona alguna API, ni framework especializado para la creación de Servicios Web basados en SOAP.

La alternativa a los Servicios Web basados en SOAP son los basados en REST. Las aplicaciones Rails se pueden estructurar de acuerdo a una arquitectura REST, ya que este estilo arquitectural permite determinar los controladores y acciones que se necesitan codificar para una aplicación Web.

En un contexto de aplicaciones de Rails, REST significa que la mayoría de los componentes (por ejemplo, los usuarios) de una aplicación son modelados como recursos que se pueden crear, leer, actualizar y eliminar, acciones que corresponden a las operaciones básicas (CRUD) que se realizan sobre las bases de datos relacionales y que en las aplicaciones Rails son asociadas con los métodos POST, GET, PUT y DELETE de una solicitud HTTP [41].

4.4. Acceso a Datos

4.4.1. Soporte para múltiples fuentes de datos

Las bases de datos son el único tipo de fuente de datos soportado por aplicaciones Rails. Si bien es posible utilizar los datos de archivos XML o cadenas JSON, los mecanismos para la carga y extracción de los datos se deben crear en su totalidad, pero las capacidades de Mapeo Objeto Relacional y migraciones no estarán disponibles.

Las aplicaciones Rails utilizan de forma predeterminada bases de datos SQLite3 para cada uno de los entornos de ejecución ofrecidos por Ruby on Rails. Sin embargo las aplicaciones también se pueden configurar con los siguientes manejadores de bases de datos [42]:

- MySQL
- PostgreSQL
- Oracle
- SQL Server

4.4.2. Soporte de Mapeo Objeto Relacional

Ruby on Rails proporciona soporte de Mapeo Objeto Relacional para aplicaciones Web, mediante la biblioteca Active Record. Esta biblioteca contiene una implementación del patrón Active Record, que ofrece un enfoque simple y eficiente para el acceso y la persistencia de objetos a bases de datos dentro de una aplicación Web [43]. En este patrón los datos y el comportamiento están encapsulados en los objetos de los modelos de dominio (ver figura 4.1). Active Record incorpora la lógica de acceso a datos en los objetos de dominio para que las operaciones comunes (por ejemplo, lectura y escritura en una base de datos) se puedan realizar de forma directa, ya que generalmente los datos de estos objetos deben leerse o almacenarse en una base de datos [44].

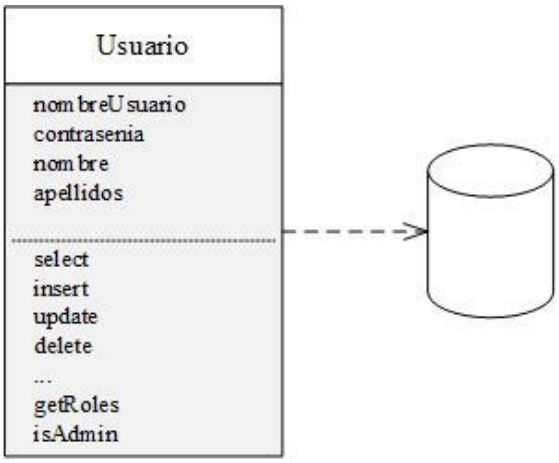

Figura 4.1: Representación del patrón Active Record

En la implementación del patrón Active Record de Ruby on Rails, un modelo de dominio (clase Ruby) es asignado por defecto a una tabla de base de datos y una instancia (objeto) de dicho modelo a una de las filas de la tabla.

La biblioteca Active Record de Ruby on Rails incluye mecanismos para la representación de los modelos y sus relaciones, operaciones CRUD, búsquedas complejas, validación, entre otros. Se apoya en gran medida en el principio de "convención sobre configuración", por lo que es más fácil de usar cuando se está creando un nuevo esquema de base de datos que puede seguir esas convenciones. También puede ser adaptado para que funcione bien con esquemas de bases de datos heredadas que no cumplen necesariamente con las convenciones de Rails [43].

Active Record incluye varios mecanismos, pero lo que más se destaca es su capacidad de [42]:

- Representar modelos y sus datos.
- Representar las asociaciones entre los modelos.
- Representar jerarquías de herencia a través de modelos relacionados.
- Validar modelos antes de los datos se almacenen en la base de datos.
- Realizar operaciones de base de datos con un lenguaje orientado a objetos.

El uso de las convenciones adoptadas por Rails para crear los modelos y las tablas es recomendado ampliamente. Active Record utiliza de forma predeterminada

una convención para los nombres de los modelos y las tablas de una base de datos. De esta forma se puede establecer una asociación entre los modelos y las tablas. Para que Active Record pueda inferir el nombre de una tabla a partir del nombre de un modelo, se deben utilizar las siguientes reglas para los nombres de modelos y tablas [45]:

- El nombre de la tabla de una base de datos debe ser el plural del nombre del modelo definido en la aplicación y que está asociado con dicha tabla.
- El nombre de la tabla debe estar en minúsculas.
- Si el nombre de un modelo se compone de dos o más palabras, cada una de estas debe comenzar con mayúscula. El nombre de la tabla asociada a un modelo con este tipo de nombre, se conformara con las mismas palabras separadas por guiones bajos y tomando en cuenta los puntos anteriores.

Active Record además asume la existencia de ciertos campos en las tablas de una base de datos. Para que los modelos de una aplicación se puedan asociar de forma predeterminada con las tablas correspondientes, es necesario considerar algunas reglas para los nombres de los siguientes tipos de campo de una tabla:

- Claves primarias: de forma predeterminada, Active Record utiliza un campo de tipo entero y que se llame id, como clave primaria de la tabla.
- Claves foráneas: estos campos se nombran de acuerdo a la siguiente convención nombre_singular_de_la_tabla_id, por ejemplo, articulo_id, pedido_id, etc.

Estas convenciones comúnmente se usan cuando se tiene que crear la base de datos para una aplicación Web. Esto agiliza la construcción de una aplicación, debido a que reduce significativamente la cantidad de código que se tiene que escribir.

4.4.3. Migraciones para esquemas de bases de datos

Rails proporciona un lenguaje de dominio específico para administrar el esquema de una base de datos, el cual se denomina migraciones. El propósito de usar este lenguaje es evitar el manejo de los cambios a través de sentencias SQL y permitir que el esquema y los cambios sean independientes del manejador de bases de datos utilizado.

Una migración puede verse como una versión nueva de una base de datos. Las migraciones se utilizan desde la creación del esquema de una base de datos. El esquema en un principio no contiene nada, y cada migración lo modifica para agregar o quitar tablas, columnas u otros objetos de bases de datos. Active Record proporciona las instrucciones y comandos para actualizar un esquema, pero también facilita lo necesario para deshacerlas.

Es importante considerar que las migraciones únicamente son envueltas en una transacción en aquellos manejadores de bases de datos que admiten transacciones con sentencias que pueden modificar el esquema. Si el manejador no admite lo anterior y una migración falla durante su ejecución, las partes de la migración que han tenido éxito no se revertirán, por lo que esto se tendrá que realizar manualmente.

Las migraciones se almacenan como archivos dentro de un directorio determinado de la aplicación y que se ejecutan sobre cualquier manejador de bases de datos que Active Record soporte. El nombre de estos archivos es de la forma `YYYYMMDDHHMMSS_create_usuarios.rb`, es decir, se conforma de una marca de tiempo seguida de un guion bajo y un nombre representativo de los cambios que se están realizando. Rails utiliza la marca de tiempo para determinar qué migración se debe ejecutar y en qué orden, por lo que sí se copia una migración desde otra aplicación o se genera nuevamente, se debe tener en cuenta el orden en que se ejecutará [46].

4.5. Seguridad

4.5.1. Implementación del proceso de autenticación

Existen varios complementos de autenticación de terceros, que se encuentran disponibles para aplicaciones Rails. Los más populares son devise y authlogic, que tienen como característica principal el almacenamiento de las contraseñas de forma cifrada, es decir, no se pueden configurar para que las contraseñas se almacenen en texto plano [47]. El complemento devise se compone de los siguientes módulos [48]:

- El módulo encargado de cifrar y almacenar la contraseña de un usuario en la base de datos.
- Un módulo que permite agregar soporte OmniAuth, para utilizar distintos sistemas de autenticación en una aplicación Web. Por ejemplo, OmniAuth permite que los usuarios se puedan autenticar mediante sus cuentas de Facebook o Twitter.

- El módulo encargado de enviar correos electrónicos con instrucciones para confirmar el registro de un usuario.
- Un módulo que permite restablecer la contraseña de un usuario.
- El módulo que permite el registro de un usuario, así como la modificación y la eliminación de su cuenta.
- Un módulo para llevar un seguimiento de las fechas y direcciones IP de los inicios de sesión de un usuario.
- El módulo que permite expirar las sesiones que no han estado activas durante un período de tiempo especificado en la aplicación.
- Un módulo para realizar validaciones de correo electrónico y contraseña.
- El módulo que bloquea la cuenta de un usuario después de un número especificado de intentos de inicio de sesión fallidos en una aplicación.

Aunque la forma más común de implementar el proceso de autenticación dentro de una aplicación Rails es mediante alguno de los complementos anteriores, Ruby on Rails permite utilizar el método `has_secure_password`, con el que se pueden tener características similares a devise y authlogic, pero se necesita escribir más código, lo cual aumenta el tiempo y esfuerzo empleado para implementar la autenticación. Tanto los complementos de terceros, como el método de autenticación proporcionado por Rails, pueden personalizarse de acuerdo a las necesidades de autenticación de la aplicación Web.

4.5.2. Implementación del proceso de autorización

Una vez que el usuario se ha autenticado en una aplicación Web, mediante el proceso de autorización se puede limitar su acceso a ciertas acciones, por ejemplo, para actualizar un registro. Una forma común de restringir el acceso a los recursos de una aplicación es a través de los roles de usuario, con los cuales también se puede condicionar el contenido de las vistas de una aplicación [43]. La implementación de una autorización simple basada en roles en una aplicación Rails, comúnmente requiere lo siguiente [48]:

- Un atributo para los roles de los usuarios.
- Métodos auxiliares en los controladores de la aplicación, para definir sentencias condicionales con las cuales se pueda restringir el acceso a determinadas acciones.

- Métodos que permitan verificar los roles de un usuario, para mostrar el contenido adecuado de las vistas de la aplicación.

Lo anterior generalmente provoca que el código de los controladores crezca y se vuelvan difíciles de mantener. Es por esto que los desarrolladores prefieren el uso de bibliotecas para implementar el proceso de autorización en una aplicación Rails. Pundit y CanCan son las bibliotecas más utilizadas para esto, debido a que permiten organizar y centralizar las reglas de acceso en clases para este propósito, que se asocian a los modelos de la aplicación.

Pundit se apoya principalmente en el concepto de clases de políticas. Una política es una clase que tiene el mismo nombre que la clase de un modelo, pero con el sufijo Policy. En la clase de políticas se utiliza un usuario y una instancia del modelo para establecer los permisos que el usuario tiene sobre determinadas acciones del modelo.

Los métodos de Rails para restringir acciones y verificar los roles de un usuario en general son suficientes para aplicaciones con requerimientos de acceso sencillos, pero para aplicaciones más complejas se recomienda utilizar alguna biblioteca, sobre todo porque permiten administrar el proceso de autorización de mejor manera, además de que los controladores de la aplicación se mantienen simples.

4.5.3. Mecanismos para evitar ataques informáticos comunes

En la documentación de Ruby on Rails se puede encontrar una guía de seguridad, la cual contiene algunas medidas para prevenir los siguientes ataques en contra de aplicaciones Rails [47]:

- Ataques de inyección SQL.
- Ataques de inyección de scripts en sitios (XSS).
- Ataques de inyección CSS.
- Ataques de inyección AJAX.
- Ataques de inyección de línea de comandos.
- Ataques de inyección de cabeceras.
- Ataques de falsificación de solicitudes en sitios (CSRF).

Medidas para prevenir ataques de inyección SQL

A nivel del código de una aplicación Rails, los ataques de inyección SQL se pueden prevenir si durante la codificación de la lógica de la aplicación se utilizan los métodos que Active Record hereda a los modelos. La mayoría de estos métodos (por ejemplo, `Model.find()`) de Active Record, contienen un filtro que escapa de forma automática los caracteres especiales SQL, tales como las comillas simples y dobles, el carácter nulo y los saltos de línea.

Medidas para prevenir ataques XSS

En las aplicaciones Web es muy importante filtrar las cadenas que se pueden ingresar a través de sus elementos de entrada, pero también es fundamental escapar la salida, es decir, el contenido que se va a mostrar en una determinada página.

Para prevenir los ataques XSS en aplicaciones Rails, se recomienda usar en general un enfoque basado en listas blancas, es decir, listas con las que se pueda indicar cuáles son los valores permitidos tanto en el contenido de entrada, como en el de salida. En la codificación de la lógica de una aplicación, se pueden utilizar algunos métodos (por ejemplo, `sanitize()` y `strip_tags()`), que se apoyan en listas blancas predefinidas para eliminar etiquetas y atributos del HTML, que no son válidos dentro de la cadena de una entrada del HTML.

Medidas para prevenir ataques CSRF

Como primera medida de prevención para este tipo de ataques, se recomienda utilizar adecuadamente los métodos GET y POST del HTTP. En el sitio Web del W3C se puede encontrar una lista que describe las situaciones en las que conviene usar GET y POST:

Utilizar GET si:

- La interacción parece más una pregunta, es decir, si es una operación segura como una consulta, una búsqueda o una operación de lectura.

Utilizar POST si:

- La interacción es para ordenar elementos.

- La interacción cambia el estado del recurso de forma, por ejemplo, el registro de un usuario a un determinado servicio.
- Los datos que el usuario envía determinan los resultados de la interacción.

Además de lo anterior, se aconseja incluir un token de seguridad en las solicitudes que no se realicen con el método GET y verificarlo en el servidor Web. Los formularios de una aplicación Rails incluyen de forma predeterminada un token de seguridad, el cual es cotejado en el servidor Web cuando los valores del formulario son enviados. Si el token no coincide con lo esperado, se lanza una excepción.

4.6. Capacidad de Pruebas

4.6.1. Entornos de ejecución para aplicaciones Web

Para las aplicaciones Rails existen tres entornos o modos de ejecución predefinidos en los que se puede configurar el servidor Web. En la siguiente tabla se describen dichos entornos de ejecución [39]:

Tabla 4.1: Entornos de ejecución predefinidos para aplicaciones Rails

Entorno de ejecución	Descripción
Desarrollo	Es recomendable usarlo durante el desarrollo de una aplicación Web. En cada solicitud al servidor Web cargara todo el código fuente de la aplicación, lo que provocara un menor rendimiento del servidor, pero reducirá considerablemente la necesidad de reiniciar el mismo.
Pruebas	Se utiliza para realizar pruebas en una aplicación. Cada vez que el servidor Web es reiniciado los datos almacenados en la base de datos son destruidos, por lo que se debe tener cuidado de no utilizar la misma base de datos para los tres entornos que se pueden configurar en una aplicación Rails. Para el nombre de las bases de datos de cada entorno se recomienda usar la siguiente convención: nombre_de_la_base_de_datos<_entorno de ejecución>
Producción	Este entorno se debe configurar para las aplicaciones Web terminadas, es decir, las que ya pueden ser usadas por los usuarios finales. Bajo este entorno no se muestran las excepciones que pueda provocar el código de la aplicación. El código ya no es cargado totalmente en cada

	solicitud al servidor Web, por lo que el rendimiento mejora considerablemente.

Aunque los entornos anteriores son suficientes para la mayoría de los casos, existen circunstancias en las que se necesita otro tipo de entorno. Por ejemplo, a veces se tiene la necesidad de trabajar en un "entorno de ensayo", es decir, un ambiente que refleje el entorno de producción, pero que además sirva para realizar pruebas. Es posible definir un entorno como este para una aplicación Rails, mediante un archivo llamado "staging.rb" con los parámetros requeridos e iniciar el servidor en este nuevo modo [40].

4.6.2. Soporte para pruebas unitarias

El soporte proporcionado para pruebas unitarias en aplicaciones Rails se enfoca principalmente en probar los métodos y acciones de los modelos y controladores. Este tipo de pruebas se codifican en clases de pruebas unitarias. El esqueleto de una clase de pruebas unitarias es generado de forma automática cuando se crea un modelo o controlador con el generador de Rails, aunque también es posible generar la clase de forma individual [49]. Algunos casos de pruebas unitarias comunes que se pueden identificar en aplicaciones Rails son los siguientes [50]:

- Casos especiales de la lógica de negocio, como lo que sucede si los datos son nulos o tienen un valor inesperado.
- Casos de error que puedan producir una experiencia de usuario única.
- Casos de implementación, como valores devueltos por los métodos.

Las pruebas unitarias en aplicaciones Rails usan una biblioteca de pruebas llamada MiniTest, la cual proporciona un conjunto de herramientas para pruebas Ruby, así como soporte para un desarrollo dirigido por pruebas y por comportamiento. Algunas características de MiniTest son [49]:

- Proporciona un amplio conjunto de aserciones para la creación de pruebas legibles.
- Provee un motor de pruebas de aserciones sobre expectativas especificadas.
- Proporciona una forma de evaluar el rendimiento de los algoritmos de una aplicación.

- Incluye herramientas para pruebas de simulación (mock) y verificación (stub).

4.6.3. Soporte para pruebas de integración

Al igual que para las pruebas unitarias, Rails incluye un generador que permite crear el esqueleto de clases para pruebas de integración. Los archivos que contienen las clases de estas pruebas son almacenados en un directorio de la aplicación [49].

En la documentación de Ruby on Rails se recomienda que las pruebas de integración se utilicen para probar flujos de trabajo primordiales dentro de una aplicación, por lo cual es importante identificar las partes de la aplicación donde se requiere realizar pruebas de este tipo. En un contexto de aplicaciones Rails, algunas partes que se deben considerar al realizar pruebas de integración son las siguientes:

- En la interacción entre un controlador y el modelo u otros objetos que proporcionan datos.
- En la interacción de múltiples acciones de controlador(es), que comprendan un flujo de trabajo común.
- En ciertos problemas de seguridad que impliquen la interacción entre el estado de un usuario y la acción de un controlador particular.

Ruby on Rails tiene pocas funciones para realizar pruebas de integración, pero existen herramientas de terceros que permiten cubrir de mejor forma este tipo de pruebas. Algunas opciones interesantes son las siguientes [50]:

- Capybara: es una biblioteca basada en Web que permite automatizar pruebas para simular la interacción entre usuarios y una aplicación Web. Ofrece un lenguaje de dominio específico para describir las acciones que son ejecutadas sobre una aplicación.
- Cucumber: es una herramienta que fue diseñada para escribir pruebas de integración sencillas y claras en un lenguaje llamado Gherkin.

4.6.4. Depuración de aplicaciones Web

Cuando una aplicación se comporta de manera inesperada, generalmente se imprime el contenido de las variables en el registro (log) o la consola para tratar de diagnosticar el problema. Sin embargo muchas veces este tipo de seguimiento de errores no es efectivo para encontrar la causa del problema. Contar con depurador que permita visualizar la traza de ejecución de una aplicación se vuelve indispensable en estos casos.

Rails proporciona un depurador llamado byebug, que permite establecer puntos de interrupción y revisar paso a paso el código de una aplicación Rails. Para usar este depurador es necesario establecer los puntos de interrupción e invocarlo (con un método Ruby) desde la parte de la aplicación Rails que se quiera depurar.

Cuando se alcanza un punto de interrupción o un evento inesperado, el depurador crea un contexto con información sobre el programa suspendido, para que el depurador pueda evaluar variables y proporcionar información sobre el lugar donde se detuvo el programa que se está depurando.

También existen complementos de terceros que ayudan en la búsqueda de errores y la depuración de aplicaciones Rails. A continuación se mencionan algunos de estos complementos [51]:

- Footnotes: presenta información de la solicitud realizada, en el pie de cada página de una aplicación Rails.
- Query Trace: registra el seguimiento de consultas en el registro de la aplicación.
- Query Reviewer: presenta un resumen de advertencias para cada consulta que analizó.
- Better Errors: reemplaza la página de error estándar en una aplicación Rails con otra que contiene más información del contexto, como el código fuente y la inspección de variables.

Capítulo 5

Comparación de los frameworks Web

5.1. Comparación de ASP.NET y Ruby on Rails

En esta sección se muestra la comparación de los frameworks Web ASP.NET y Ruby on Rails, para los aspectos de portabilidad, interoperabilidad, acceso a datos, seguridad y capacidad de pruebas. La comparación es presentada en tabla 5.1, mediante las preguntas obtenidas en la sección 2.2, además se toma en cuenta la información de los capítulos 3 y 4.

Tabla 5.1: Comparación de los frameworks Web ASP.NET y Ruby on Rails

Aspecto	Criterio	ASP.NET	Ruby on Rails
Portabilidad	P1: ¿Los IDE con soporte están disponibles para distintos sistemas operativos?	No	Sí, aunque el soporte es parcial en la mayoría de los IDE
	P2: ¿Admite el uso de distintos servidores Web (comerciales y no comerciales)?	No	Sí
	P3: ¿Los servidores Web que permite se pueden usar en diversos sistemas operativos?	No	Sí
	P4: ¿Ofrece soporte para distintos manejadores de bases de datos (comerciales y no comerciales)?	Sí	Sí
	P5: ¿Los manejadores de bases de datos que soporta se pueden utilizar en distintos sistemas operativos?	Sí	Sí
	P6: ¿El software que complementa o extiende la funcionalidad está disponible para diversos sistemas operativos?	No	Sí

Aspecto	Criterio	ASP.NET	Ruby on Rails
Interoperabilidad	I1: ¿Permite describir los datos de un Servicio Web mediante el formato WSDL?	Sí	No
	I2: ¿Tiene soporte para el envío de mensajes SOAP?	Sí	No
	I3: ¿Proporciona algún framework o API para la creación de Servicios Web basados en SOAP?	Sí	No
	I4: ¿Permite crear Servicios Web REST?	Sí	Sí, aunque hay que considerar que no fue diseñado con ese propósito.
	I5: ¿Proporciona algún framework o API para la creación de Servicios Web REST?	Sí	No
Acceso a Datos	AD1: ¿Permite utilizar bases de datos relacionales como fuente de datos?	Sí	Sí
	AD2: ¿Permite utilizar archivos XML como fuente de datos?	Sí	No
	AD3: ¿Ofrece soporte para otro tipo de fuente, además de bases de datos relacionales y archivos XML?	Sí	No
	AD4: ¿Permite generar el esquema de la base de datos a partir de clases existentes?	Sí	No
	AD5: ¿Permite generar las clases a partir del esquema de la base de datos existente?	Sí	No
	AD6: ¿Permite administrar el esquema de la base de datos desde el código de la aplicación?	Sí	Sí
	AD7: ¿Permite automatizar la creación de una migración?	Sí	Sí
	AD8: ¿Proporciona buenas prácticas para el uso de migraciones sobre el esquema de una base de datos?	No	No

Aspecto	Criterio	ASP.NET	Ruby on Rails
Seguridad	S1: ¿Proporciona implementaciones predefinidas del proceso de autenticación?	Sí	Sí
	S2: ¿Las implementaciones predefinidas de autenticación se pueden personalizar?	Sí	Sí
	S3: ¿Proporciona implementaciones predefinidas del proceso de autorización?	Sí	Sí
	S4: ¿Las implementaciones predefinidas de autorización se pueden personalizar?	Sí	Sí
	S5: ¿Proporciona mecanismos predefinidos para evitar ataques de inyección SQL?	Sí	Sí
	S6: ¿Proporciona mecanismos predefinidos para evitar ataques de inyección de scripts en sitios (XSS)?	Sí	Sí
	S7: ¿Proporciona mecanismos predefinidos para evitar ataques de falsificación de solicitudes en sitios (CSRF)?	Sí	Sí
	S8: ¿Proporciona una guía de seguridad?	No	Sí
Capacidad de Pruebas	CP1: ¿Permite configurar un entorno de desarrollo?	Sí	Sí
	CP2: ¿Permite configurar un entorno para realizar pruebas?	Sí	Sí
	CP3: ¿Permite configurar un entorno de producción?	Sí	Sí
	CP4: ¿Permite configurar un entorno personalizado?	No	Sí
	CP5: ¿Tiene soporte para pruebas unitarias?	Sí	Sí
	CP6: ¿Permite automatizar las pruebas unitarias?	Sí, aunque solo crea la plantilla	Sí, aunque solo crea el esqueleto

Aspecto	Criterio	ASP.NET	Ruby on Rails
Capacidad de Pruebas	CP7: ¿Proporciona alguna guía para realizar pruebas unitarias?	Sí	Sí
	CP8: ¿Tiene soporte para pruebas de integración?	Sí	Sí
	CP9: ¿Proporciona alguna guía para realizar pruebas integración?	No	No
	CP10: ¿Proporciona algún entorno para la depuración del código de la aplicación Web?	Sí	Sí
	CP11: ¿Permite usar herramientas de terceros para la depuración del código de la aplicación Web?	No	Sí
	CP12: ¿Proporciona instrucciones para la depuración del código de la aplicación Web?	Sí	Sí

5.2. Interpretación de la comparación

A partir del número total de criterios de cada aspecto entre el número de criterios cumplidos por cada framework en cada aspecto, se puede obtener el porcentaje de criterios cumplidos por cada framework en cada aspecto (ver tabla 5.2 y figura 5.1). Este porcentaje puede ser interpretado como el grado en que pueden lograrse cada uno de los aspectos en una aplicación Web, al utilizar estos frameworks.

Tabla 5.2: Porcentaje de criterios cumplidos por ASP.NET y Ruby on Rails

	Porcentaje de criterios cumplidos por:	
	ASP.NET	Ruby on Rails
Portabilidad	33.3%	100.0%
Interoperabilidad	100.0%	20.0%
Acceso a Datos	87.5%	37.5%
Seguridad	87.5%	100.0%
Capacidad de Pruebas	75.0%	91.7%

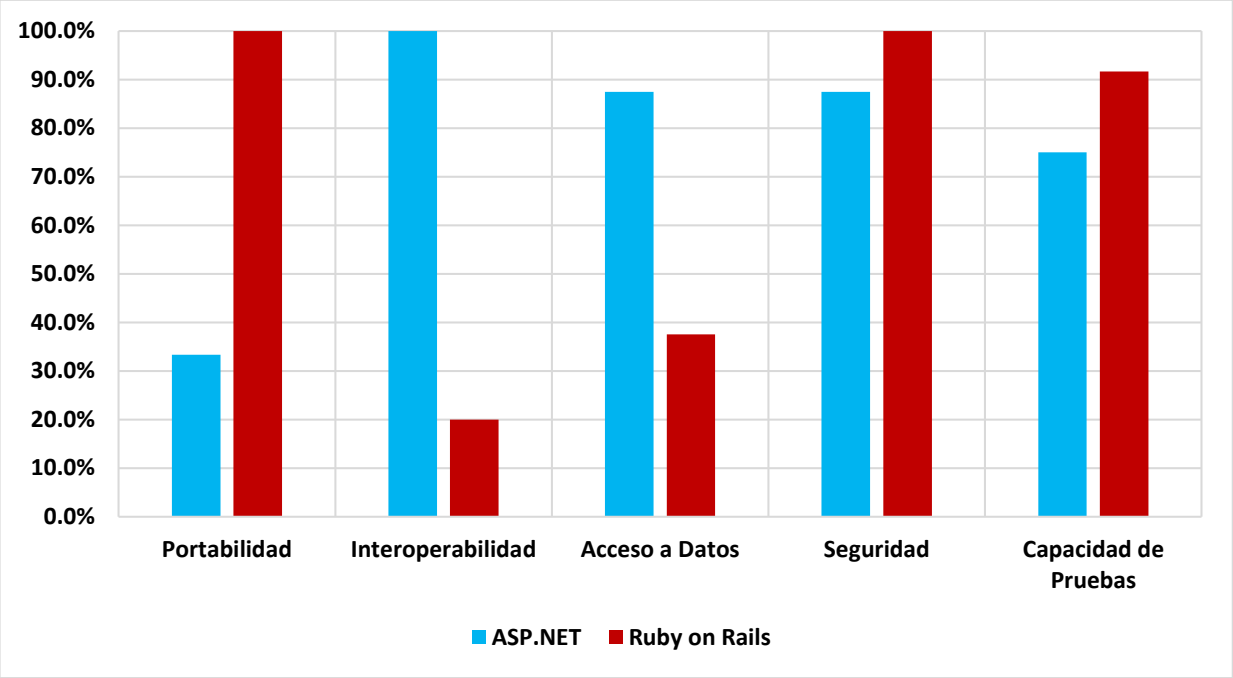

Figura 5.1: Comparación del porcentaje de criterios cumplidos por los frameworks Web, en cada aspecto

La comparación ayuda a determinar cuáles son los frameworks Web que poseen las características, que permiten lograr cada uno de los aspectos considerados en una aplicación Web. Lo que finalmente se puede apreciar es lo siguiente:

- Ruby on Rails tiene características para el desarrollo de aplicaciones Web portables, como la integración con distintos Entornos de Desarrollo Integrado, el soporte de distintos servidores Web y manejadores de bases de datos, los cuales pueden ser utilizados en varios sistemas operativos. En contraste ASP.NET no permite construir aplicaciones Web portables, debido al poco o nulo soporte de otro software distinto al de Microsoft.
- Con ASP.NET se pueden construir aplicaciones Web interoperables, mediante de la creación de Servicios Web, que utilizan estándares de la industria como WSDL y SOAP, e incluso basados en una arquitectura REST. A pesar de que las aplicaciones Rails se pueden estructurar de acuerdo al estilo REST y a partir de esto crear Servicios Web basados en REST, es importante tomar en cuenta que Ruby on Rails no fue diseñado para construir Servicios Web.
- ASP.NET tiene características más completas para el acceso a datos, tales como la generación del esquema de la base de datos a partir de las clases de una aplicación y viceversa, las cuales ayudan a agilizar la construcción de

una aplicación Web. Al disponer de menos características para el acceso a datos en aplicaciones Rails, se requiere de más codificación, aunque en la mayoría de los casos las características proporcionadas son suficientes para el acceso a datos en aplicaciones Web.

- ASP.NET y Ruby on Rails poseen características similares, con las que se puede lograr un buen nivel de seguridad para las aplicaciones Web, ya que permiten implementar los procesos de autenticación y autorización y proporcionan medidas contra ataques informáticos comunes.
- Tanto ASP.NET como Ruby on Rails proporcionan mecanismos para realizar pruebas comunes, por ejemplo, pruebas unitarias y de integración, aunque el soporte se enfoca a las primeras. Estos frameworks también inlcuyen herramientas para la depuración del código de una aplicación Web, principalmente mediante de depuradores.

Conclusiones

Una parte importante en el proceso de desarrollo de una aplicación Web, es la selección de la tecnología con la que se construirá dicha aplicación. En este trabajo se presentó una forma para comparar frameworks Web, que puede ser utilizada en la selección de este tipo de frameworks. Como se pudo observar, la comparación con criterios obtenidos a partir de características comunes de los mismos frameworks y agrupados en aspectos de la arquitectura de la aplicación Web, permite determinar cuál o cuáles de los frameworks proporcionan las herramientas adecuadas para cubrir las necesidades de la aplicación Web.

Es conveniente realizar una abstracción de los frameworks Web porque esto permite identificar sus características y a partir de éstas, conseguir criterios más relevantes con los que se pueda comparar a varios frameworks Web. Sin embargo, también es importante considerar que se requiere de cierto conocimiento o dominio de este tipo de software, para poder abstraerlos.

En cuanto a los frameworks Web que fueron comparados en este trabajo, se puede concluir que ASP.NET es más adecuado para aplicaciones Web con necesidades imperantes de interoperabilidad, acceso a datos, seguridad y pruebas, en cambio Ruby on Rails es mejor considerarlo para aplicaciones con requisitos de portabilidad, seguridad y pruebas.

Trabajo a futuro

En este trabajo se tomaron en cuenta cinco aspectos de una arquitectura común para aplicaciones Web, pero es importante mencionar que existen otros aspectos que también deben ser considerados, por ejemplo, la facilidad de uso y la madurez. Y de la misma forma que con los aspectos de este trabajo, se pueden determinar criterios que permitan la comparación de los frameworks Web sobre estos nuevos aspectos.

Los criterios identificados en este trabajo fueron utilizados para comparar determinados aspectos en algunos frameworks Web, sin embargo dichos criterios también se pueden usar para evaluar de forma distinta a este tipo de frameworks, por ejemplo, dependiendo del impacto de cada criterio para el tipo de aplicación Web que se pretenda desarrollar, se le puede asignar un peso a cada uno de los criterios, para que de este modo la importancia de los criterios sea considerada en la evaluación.

Finalmente, las características de los frameworks Web presentadas en este trabajo, se pueden considerar en el diseño de un nuevo framework Web, para abarcar la mayoría o sólo determinados aspectos que sean relevantes para cierto tipo de aplicaciones Web, dependiendo del alcance que se quiera ofrecer o lograr.

Referencias

[1] Kappel, G., Pröll, B., Reich, S., Retschitzegger, W. (2006). *Web Engineering: The Discipline of Systematic Development of Web Applications*. John Wiley & Sons.

[2] Shklar, L., Rosen, R. (2009). *Web Application Architecture: Principles, Protocols, and Practices*. John Wiley & Sons.

[3] Salas-Zarate et al. (2015). Analyzing best practices on Web development frameworks: The lift approach. *Science of Computer Programming*. 102, págs. 1-19.

[4] Pressman, R., Lowe, D. (2008). *Web engineering: A practitioner's approach*. McGraw-Hill.

[5] Bass, L., Clements, P., Kazman, R. (2012). *Software Architecture in Practice*. Addison-Wesley.

[6] Microsoft Patterns & Practices Team. (2009). *Microsoft Application Architecture Guide (Patterns & Practices)*. Microsoft Press.

[7] Murugesan, S., Deshpande, Y. (2001). *Web Engineering: Managing Diversity and Complexity of Web Application Development*. Springer.

[8] Freire, M., Pereira, M. (2007). *Encyclopedia of Internet Technologies and Applications*. Information Science Publishing.

[9] BuiltWith. *Framework Usage Statistics. Statistics for websites using Framework technologies*. Disponible en https://trends.builtwith.com/framework [Consultado en Enero del 2017]

[10] Amazon. *Búsqueda de libros relacionados con Frameworks de PHP, ASP.NET, Ruby on Rails y Frameworks de Java*. Disponible en https://www.amazon.com [Consultado en Febrero del 2017]

[11] Gorton, I. (2011). *Essential Software Architecture*. Springer.

[12] World Wide Web Consortium (W3C). *Web Services Architecture*. Disponible en https://www.w3.org/TR/ws-arch/#whatis [Consultado en Febrero del 2017]

[13] Dix, P. (2010). *Service-Oriented Design with Ruby and Rails*. Addison-Wesley.

[14] Barcia, R., Hambrick, G., Brown, K. (2008). *Persistence in the Enterprise: A Guide to Persistence Technologies*. IBM Press.

[15] Laravel. *Database: Migrations*. Disponible en https://laravel.com/docs/5.4/migrations [Consultado en Febrero del 2017]

[16] The Open Web Application Security Project (OWASP). *Category:Attack*. Disponible en https://www.owasp.org/index.php/Category:Attack [Consultado en Marzo del 2017]

[17] Braude, E., Bernstein, M. (2016). *Software Engineering: Modern Approaches* Waveland Press.

[18] Microsoft. *ASP.NET Documentation – ASP.NET – Introduction to ASP.NET - ASP.NET Overview*. Disponible en https://docs.microsoft.com/en-us/aspnet/overview [Consultado en Marzo del 2017]

[19] Microsoft. *Documentation – APIs and Reference – ASP.NET reference - ASP.NET y Visual Studio para Web*. Disponible en https://msdn.microsoft.com/en-us/library/dd566231.aspx [Consultado en Marzo del 2017]

[20] Microsoft. *Documentation – APIs and Reference – ASP.NET reference – ASP.NET Overview*. Disponible en https://msdn.microsoft.com/en-us/library/4w3ex9c2(v=vs.120).aspx [Consultado en Marzo del 2017]

[21] Microsoft. *Documentation – APIs and Reference – ASP.NET Development Requirements*. Disponible en https://msdn.microsoft.com/en-us/library/ms228041(v=vs.85).aspx [Consultado en Marzo del 2017]

[22] Microsoft. *Docs – ASP.NET – ASP.NET Core – Working with multiple environments*. Disponible en https://docs.microsoft.com/en-us/aspnet/core/fundamentals/environments [Consultado en Abril del 2017]

[23] Microsoft. *Docs – .NET – .NET Framework – Windows Communication Foundation – What Is Windows Communication Foundation*. Disponible en https://docs.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf [Consultado en Abril del 2017]

[24] Microsoft. *Docs – APIs and Reference – Data Source Controls Overview*. Disponible en https://msdn.microsoft.com/en-us/library/ms227679(v=vs.100).aspx [Consultado en Abril del 2017]

[25] Microsoft. *Docs – APIs and Reference – ASP.NET Data Access Option*. Disponible en https://msdn.microsoft.com/en-us/library/ms178359(v=vs.110).aspx [Consultado en Abril del 2017]

[26] Microsoft. *Documentation – APIs and Reference – Entity Framework Features*. Disponible en https://msdn.microsoft.com/en-us/library/bb896338(v=vs.90).aspx [Consultado en Mayo del 2017]

[27] Microsoft. *Documentation – APIs and Reference – Entity Framework Code First Migrations*. Disponible en https://msdn.microsoft.com/en-us/library/jj591621(v=vs.113).aspx [Consultado en Mayo del 2017]

[28] Microsoft. *Documentation – APIs and Reference – ASP.NET Authentication*. Disponible en https://msdn.microsoft.com/en-us/library/eeyk640h.aspx [Consultado en Junio del 2017]

[29] Dorrans, B. (2010). *Beginning ASP.NET Security*. Wrox.

[30] Microsoft. *Documentation – APIs and Reference – ASP.NET Authorization*. Disponible en https://msdn.microsoft.com/en-us/library/wce3kxhd.aspx [Consultado en Junio del 2017]

[31] Microsoft. *Docs – ASP.NET – ASP.NET Core – Preventing Cross-Site Scripting*. Disponible en https://docs.microsoft.com/en-us/aspnet/core/security/cross-site-scripting [Consultado en Agosto del 2017]

[32] Microsoft. *Docs – ASP.NET – ASP.NET Web API – Preventing Cross-Site Request Forgery (CSRF) Attacks in ASP.NET Web API*. Disponible en https://docs.microsoft.com/en-us/aspnet/web-api/overview/security/preventing-cross-site-request-forgery-csrf-attacks [Consultado en Agosto del 2017]

[33] Microsoft. *Documentation – APIs and Reference – Unit Testing Web Applications*. Disponible en https://msdn.microsoft.com/en-us/library/ff936235(v=vs.100).aspx [Consultado en Septiembre del 2017]

[34] Freeman, A., Sanderson, S. (2012). *Pro ASP.NET MVC 4*. Apress.

[35] Spaanjaars, I. (2014). *Beginning ASP.NET 4.5.1 in C# and VB*. Wrox.

[36] Rails. *The Rails Doctrine*. Disponible en http://rubyonrails.org/doctrine [Consultado en Marzo del 2017]

[37] Ruby on Rails API. *The API Documentation*. Disponible en http://api.rubyonrails.org [Consultado en Marzo del 2017]

[38] Rails Guides. *Getting Started with Rails*. Disponible en http://guides.rubyonrails.org/getting_started.html [Consultado en Agosto del 2017]

[39] Ponce, S. (2013). *Ruby on Rails. Desarrollo práctico de aplicaciones Web*. Alfaomega Grupo Editor.

[40] Rails Guides. *Configuring Rails Applications*. Disponible http://guides.rubyonrails.org/configuring.html [Consultado en Agosto del 2017]

[41] Puglisi, S. (2015). *RESTful Rails Development: Building Open Applications and Services*. O'Reilly Media.

[42] Rails Guides. *Active Record Basics*. Disponible en http://guides.rubyonrails.org/active_record_basics.html [Consultado en Septiembre del 2017]

[43] Fernandez, O. (2014). *The Rails 4 Way*. Addison-Wesley.

[44] Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley.

[45] Marshall, K., Pytel, C., Yurek, J. (2007). *Pro Active Record: Databases with Ruby and Rails*. Apress.

[46] Rails Guides. *Active Record Migrations*. Disponible en http://guides.rubyonrails.org/active_record_migrations.html [Consultado en Septiembre del 2017]

[47] Rails Guides. *Ruby on Rails Security Guide*. Disponible en http://guides.rubyonrails.org/security.html [Consultado en Septiembre del 2017]

[48] Barie, H., Mutiara, N., Sakti, G. (2013). *Learning Devise for Rails*. Packt Publishing.

[49] Rails Guides. *A Guide to Testing Rails Applications*. Disponible en http://guides.rubyonrails.org/v4.2/testing.html [Consultado en Septiembre del 2017]

[50] Rappin, N. (2014). *Rails 4 Test Prescriptions: Build a Healthy Codebase*. Pragmatic Bookshelf.

[51] Rails Guides. *Debugging Rails Applications*. Disponible en http://guides.rubyonrails.org/v4.2/debugging_rails_applications.html [Consultado en Septiembre del 2017]