CAPÍTULO 1

Marco teórico

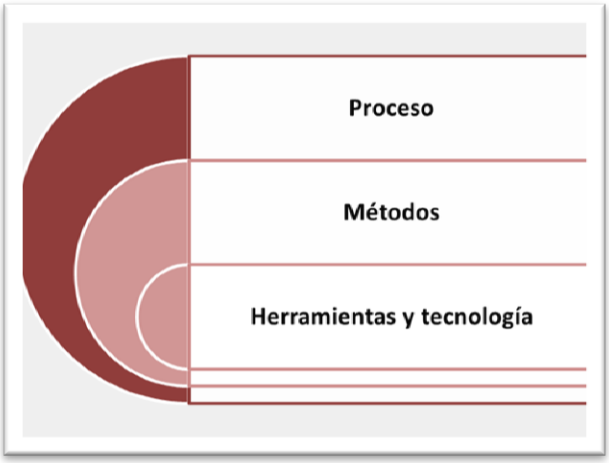
1.1 Ingeniería Web (IWeb)

Con el desarrollo de Internet, la mayoría de los proyectos y sistemas están enfocados para las aplicaciones basadas en la Web (WebApps). Por lo que ahora se habla de la ingeniería Web (IWeb).

La IWeb no es más que la ingeniería de software, pero con un enfoque a las WebApps. La ingeniería de software es una tecnología estratificada (figura 1.1); cualquier enfoque de la ingeniería debe estar sustentado en un compromiso con la calidad, fomentando la mejora continua de sus procesos.

1.1 Estratos de la ingeniería de software

En la figura 1.1 se ilustran los estratos de la ingeniería de software, pero si se está hablando de las WebApps, los estratos de la ingeniería de software se pueden representar también tal como se ilustra en la figura 1.2.



1.1 Estratos de la ingeniería WebApp

Para la figura 1.2, se observan los estratos denominados Proceso, Métodos y Herramientas y tecnología. El estrato Proceso forma la base para el control de la gestión de los proyectos de software y establece los productos de trabajo como son: los modelos, documentos, datos, formatos, etc., asegurando la calidad de software, donde se encuentre el rendimiento, la confiabilidad y la escalabilidad. En cuanto al estrato Métodos, va a proporcionar las formas para poder construir el software, donde se incluyen los requisitos y el diseño de la WebApp. El sistema se construye con el estrato Herramientas y tecnología especializada asociada con la Web. Finalmente, el producto se entrega a los usuarios y se evalúa mediante criterios, tanto técnicos como empresariales.

Dado que la Web evoluciona continuamente, se deben establecer mecanismos para el control de configuraciones, el aseguramiento de la calidad en todos sus estratos, junto con un soporte continuo. Es por estas razones que cuando se habla de la ingeniería Web, ya no se visualiza el estrato Un enfoque de calidad, como se observa en la figura 1.1, este estrato ya viene explícito en todos y cada uno de sus niveles, asegurando no correr el riesgo de producir una Web enmarañada o mal desarrollada con altas prioridades de fracaso conforme ésta vaya creciendo.

En la figura 1.3 se mencionan algunas de las características de los estratos de la ingeniería Web para las WebApps.

Proceso

Los modelos de procesos deben adoptar un desarrollo ágil que incorpore rápidos ciclos de desarrollo. Reducir el ciclo de desarrollo es ahora una de las misiones más importantes de la ingeniería del software.

Actividades que se deben definir dentro de un proceso:

1) Adopte el cambio
2) Adopte la creatividad y la independencia del equipo de desarrollo
3) Construya sistemas que utilicen pequeños equipos de desarrollo
4) Subraye el desarrollo evolutivo o incremental mediante el uso de cortos ciclos de desarrollo

Métodos

El panorama de los métodos de la ingeniería Web abarca un conjunto de labores técnicas que permiten a los desarrolladores Web comprender, caracterizar y construir una WebApp de alta calidad. Se pueden categorizar de la siguiente manera:

Métodos de comunicación

Define la comunicación entre los desarrolladores Web y los demás participantes de la WebApp. Las técnicas de comunicación son importantes durante la recolección de requisitos.

Métodos de análisis de requisitos

Proporcionan una base para comprender el contenido de una WebApp, la función que proporcionará al usuario final y los modos de interacción que requerirá mientras ocurra la navegación por medio de la WebApp.

Métodos de diseño

Abarca una serie de técnicas de diseño que abordan el contenido, la aplicación y la arquitectura de información, así como el diseño de interfaz y la estructura de navegación de la WebApp.

Métodos de prueba

Incorpora revisiones, tanto del contenido y el modelo de diseño, abarcando los conflictos existentes en la arquitectura y sus componentes. Además de incorporar pruebas de navegación, factibilidad de uso, pruebas de seguridad y pruebas de configuración.

Herramientas y Tecnología

La evolución tanto de la tecnología como de las herramientas orientadas a las WebApp, han propiciado que éstas sean más complejas y extendidas.

Dicha tecnología abarca un amplio conjunto de lenguajes de modelación para la generación de contenidos, como el HTML, XML, lenguajes de programación para el desarrollo de componentes, como JAVA, .NET, PHP, CORBA, entre otros. Mientras que en la parte de herramientas están las de multimedia, de conectividad de base de datos, herramientas de seguridad y utilidades de servidor, herramientas de administración y más.

Figura 1.3 Características de los estratos de la ingeniería Web[1]

[1] Pressman, Roger S. *Ingeniería del software, un enfoque práctico*, McGraw-Hill, Sexta edición, Madrid España, 2006. Páginas 507-508.

1.1.1 Atributos de los sistemas y aplicaciones basados en Web

Cabe mencionar que cada WebApp posee ciertos tipos de categorías que apliquen según sean sus necesidades, como por ejemplo si es que está orientada a transacciones, acceso a una base de datos, orientada a servicios, de descarga, si sólo es informativa, personalizable, entrada del usuario con base a formularios, ya sea para almacén de datos o para extraer información.

Los atributos que se pueden encontrar en la mayoría de las WebApps, se describen en la tabla 1.1. Atributos de la WebApp.

Tabla 1.1 Atributos de la WebApp

Atributo	Descripción
Intensidad de red	Las aplicaciones Web residen en una red y debe satisfacer las necesidades de una variada comunidad de clientes, por lo que puede permitir una comunicación mundial.
Concurrencia	Un gran número de usuarios puede tener acceso a la WebApp al mismo tiempo.
Carga impredecible	El número de usuarios que acceden a la WebApp puede variar, un día puede haber más número de visitantes en comparación con otros.
Desempeño	Si un usuario espera demasiado para que se despliegue la información que requiere, propiciará a que se vaya a cualquier otra parte.
Disponibilidad	Algunos usuarios demandan disponibilidad para accesos a la WebApp
Gobernada por los datos	Algunas de las funciones primordiales de la WebApp es usar hipermedia para presentar contenidos de texto, gráficos, audio, video al usuario final.
Sensibilidad al contenido	La calidad estética del contenido sigue siendo un importante determinante de la calidad de una WebApp.
Evolución continua	Las aplicaciones Web evolucionan de manera continua. Algunas están diseñadas para que se vayan actualizando en un tiempo específico o bien el contenido sea calculado de manera independiente para cada solicitud. Por lo que su crecimiento debe de realizarse de forma controlada y consistente.
Inmediatez	Los desarrolladores Web deben aplicar métodos de planeación, análisis, diseño, implementación y puesta a prueba que han sido adaptados a los apretados tiempos requeridos para el desarrollo de la WebApp.
Seguridad	Como las aplicaciones Web están disponibles mediante la red, es difícil limitar la población de usuarios finales que pueden tener acceso a la aplicación. Con la finalidad de proteger el contenido confidencial y la transmisión de datos, se deben de implementar fuertes medidas de seguridad a lo largo de la infraestructura que sustenta y una WebApp.
Estética	Una parte muy importante de todo sitio Web es indudablemente su presentación y disponibilidad de sus elementos. Cuando una aplicación se diseña para comercializar o vender, la estética puede tener tanto que ver con el éxito como de su diseño técnico.

1.1.2 El proceso de la IWeb

El proceso que se debe de elegir para el desarrollo de un sitio Web depende mucho de qué tipo de proyecto se desea implementar o también depende de los acuerdos con los que se lleguen a plantear con el cliente a quien se le va a desarrollar el sitio Web. Se puede elegir desde un proceso ágil que produzca liberaciones de aplicaciones Web a un ritmo acelerado, o elegir un modelo incremental para un proceso más lento y más elaborado, que necesiten ser más detallados y analizados con detenimiento.

Las actividades del marco de trabajo para la ingeniería Web se aplican empleando un flujo de proceso incremental, como se representa en la figura 1.4.

Figura 1.4 El proceso de la ingeniería Web[2]

[2] Pressman, Roger S. *Ingeniería del software, un enfoque práctico*, McGraw-Hill, Sexta edición, Madrid España, 2006. Pág. 511.

a) **Comunicación con el cliente:** se caracteriza por el análisis del negocio y la formulación. En el análisis del negocio se define el contexto empresarial-organizativo para la WebApp, se identifican los participantes, se define la integración entre la WebApp y otras aplicaciones de negocio, base de datos y funciones. Mientras que en la formulación es una actividad de recopilación de requisitos, describe el problema que la WebApp habrá de resolver junto con los requisitos básicos para la WebApp.
b) **Planeación:** se crea un plan de proyecto para el incremento de la WebApp.
c) **Modelado:** las labores convencionales de análisis y diseño de ingeniería de software se adaptan al desarrollo de la WebApp, se mezclan y luego se funden en la actividad de modelado IWeb.
d) **Construcción:** las herramientas y la tecnología IWeb se aplican para construir la WebApp que se ha modelado. Una vez que se construye, se dirige a una serie de pruebas rápidas para asegurar que se descubran los errores en el diseño (es decir, contenido, arquitectura, interfaz, navegación).
e) **Entrega y retroalimentación:** la WebApp se configura para su ambiente operativo, se entrega a usuarios finales y luego comienza un periodo de evaluación. La retroalimentación acerca de la evaluación se presenta al equipo de IWeb y el incremento se modifica conforme se requiera.

1.1.3 ¿Qué metodología usar para el desarrollo de un proyecto de ingeniería Web?

Las metodologías más recomendadas para el desarrollo de proyectos Web son los modelos de procesos ágiles, ya que las WebApps suelen tener actualizaciones constantemente, por lo que el proceso debe tener ciclos de desarrollo cortos. Y antes de definir un marco de trabajo para el proceso de las WebApps, se tienen que considerar los siguientes tres puntos:

1. **Las WebApps con frecuencia se entregan de manera incremental:** las actividades del marco de trabajo ocurrirán de manera repetida conforme cada incremento se someta a ingeniería y se entregue.
2. **Los cambios ocurrirán frecuentemente:** los cambios pueden ocurrir como resultado de la evaluación de un incremento entregado o como consecuencia de cambiar las condiciones de los negocios.
3. **Los plazos cortos:** al tener plazos cortos aminora la creación y revisión de voluminosa documentación de ingeniería, pero no excluye la simple realidad de que el análisis crítico, el diseño y la prueba deban registrarse en alguna forma.

Existen varias metodologías que emplean procesos ágiles, pero en este capítulo sólo se abarcarán dos de ellas, el Proceso Unificado Racional (RUP) y el proceso de *extreme programming* (XP), presentadas a continuación:

El Proceso Unificado Racional (RUP)

El Proceso Unificado (también llamado Proceso Unificado Racional, después de que lo respalda la *Rational Corporation*, que es un contribuyente importante en el desarrollo y refinamiento de proceso y un conductor de herramientas y tecnología), es un intento encaminado a reunir las mejores características de los modelos de procesos de software e incluso llega a implementar mucho de los mejores principios del desarrollo ágil de software.

En la actualidad, el RUP se emplea de forma amplia en proyectos orientados a objetos de todos los tipos. Además de reconocer la importancia de la comunicación con el cliente y enfatizar el importante papel de la arquitectura de software.

En un proyecto de RUP organiza el trabajo y las iteraciones en cuatro fases fundamentales descritas en la tabla 1.2:

Tabla 1.2 Características de las fases de RUP

Fase	Características
Inicial	Abarca la comunicación con el cliente y las actividades de planeación. Se identifican los requisitos de negocios, se propone una arquitectura aproximada para el sistema y se desarrolla el plan de desarrollo de software para las subsiguientes fases.
Elaboración	Abarca la comunicación con el cliente y las actividades del modelado genérico del proceso. Proporciona los casos de uso preliminares. Se define la arquitectura básica y se planifica el proyecto considerando recursos disponibles.
Construcción	Se concentra en la elaboración de un producto totalmente operativo y eficiente, además de generar el manual de usuario. Aquí en esta fase el producto se desarrolla a través de iteraciones donde cada iteración involucra tarea de análisis, diseño e implementación. También se reafirma la arquitectura básica (proporcionada en las fases anteriores) conforme se vaya construyendo, por lo cual se permiten cambios en la arquitectura. Gran parte del trabajo es programación y pruebas. Se documentan, tanto el sistema construido, como el manejo del mismo.
Transición	Abarca las últimas etapas de la actividad de construcción. El software se entrega a los usuarios finales para realizar las últimas pruebas, con el fin de reportar defectos como cambios necesarios. Se generan los manuales de usuario, procedimientos de instalación, entre otros para la liberación del proyecto.

El RUP describe actividades de trabajo como escribir casos de uso en disciplinas. Donde una disciplina es un conjunto de actividades y artefactos relacionados en un área determinada, como por ejemplo, las actividades realizadas en el análisis de requisitos.

En la tabla 1.3 se describen las disciplinas involucradas para RUP:

Tabla 1.3 Características de las disciplinas de RUP

Disciplina	Características
Modelado empresarial	Tiene como objetivo comprender la estructura y la dinámica de la organización, comprender problemas actuales e identificar posibles mejoras. Se utiliza el modelo de casos de usos para describir los procesos del negocio y los clientes, también se puede hacer uso de los diagramas de actividad y de clases.
Requisitos	Tiene como objetivo de especificar o establecer lo que el sistema debe hacer, también define los límites del sistema, y una interfaz de usuario, realiza una estimación del costo y tiempo de desarrollo. Utiliza el modelo de caso de usos para modelar el sistema, actores y relaciones.
Análisis y diseño	Define la arquitectura del sistema y tiene como objetivo trasladar los requisitos en implementación. Como por ejemplo, para el análisis, se transforman los casos de uso en clases, y en el diseño se refina el análisis para poder implementar los diagramas de clases, los diagramas de colaboración, diagramas de secuencia y el modelo de despliegue de la arquitectura.
Implementación	Su objetivo es implementar las clases de diseño como componentes (como por ejemplo, fichero fuente), asignar los componentes a los nodos, probar los componentes individualmente, integrar los componentes en un sistema ejecutable (enfoque incremental). Utiliza el modelo de implementación.
Prueba	Verifica la integración de los componentes (prueba de integración), además de verificar que todos los requisitos han sido implementados (pruebas del sistema) y asegura que los defectos detectados han sido resueltos antes de la distribución.
Despliegue	Asegura que el producto está preparado para el cliente y procede a su entrega. Se realizan las actividades de empaquetar, distribuir e instalar el producto, así como la tarea de enseñar al usuario.
Gestión de cambios y configuración	Es esencial para controlar el número de artefactos producidos por la cantidad de personal que trabaja en un proyecto conjuntamente. Los controles sobre los cambios son de mucha ayuda ya que evitan confusiones costosas como la compostura de algo que ya se había arreglado, y aseguran que los resultados de los artefactos no entren en conflicto.
Gestión de proyectos	Con la Gestión de proyectos se logra una mejoría en el manejo de una entrega exitosa de software. En resumen, su propósito consiste en proveer pautas para: - Administrar proyectos de software intensivos - Planear, dirigir personal, ejecutar acciones y supervisar proyectos - Administrar el riesgo
Entorno	Se enfoca sobre las actividades necesarias para configurar el proceso que engloba el desarrollo de un proyecto. Su propósito es proveer a la organización que desarrollará el software, un ambiente en el cual basarse, el cual provee procesos y herramientas para poder desarrollar el software.

En la figura 1.5 se muestra cómo es que las disciplinas están presentes en las fases del ciclo de vida de RUP.

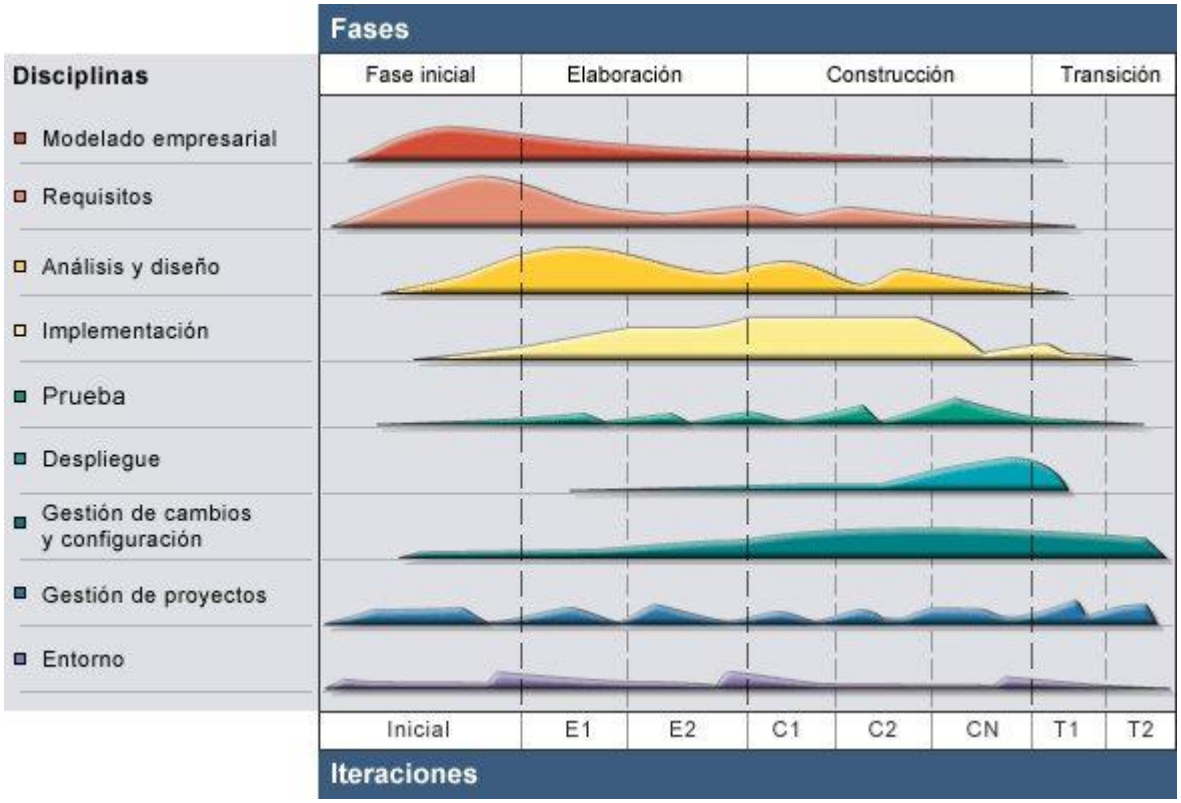


Figura 1.5 Ciclo de vida RUP[3]

Algunos de los artefactos o productos de trabajo que se pueden generar para cada fase del RUP se muestran en la figura 1.6.

[3] http://www.rational.com

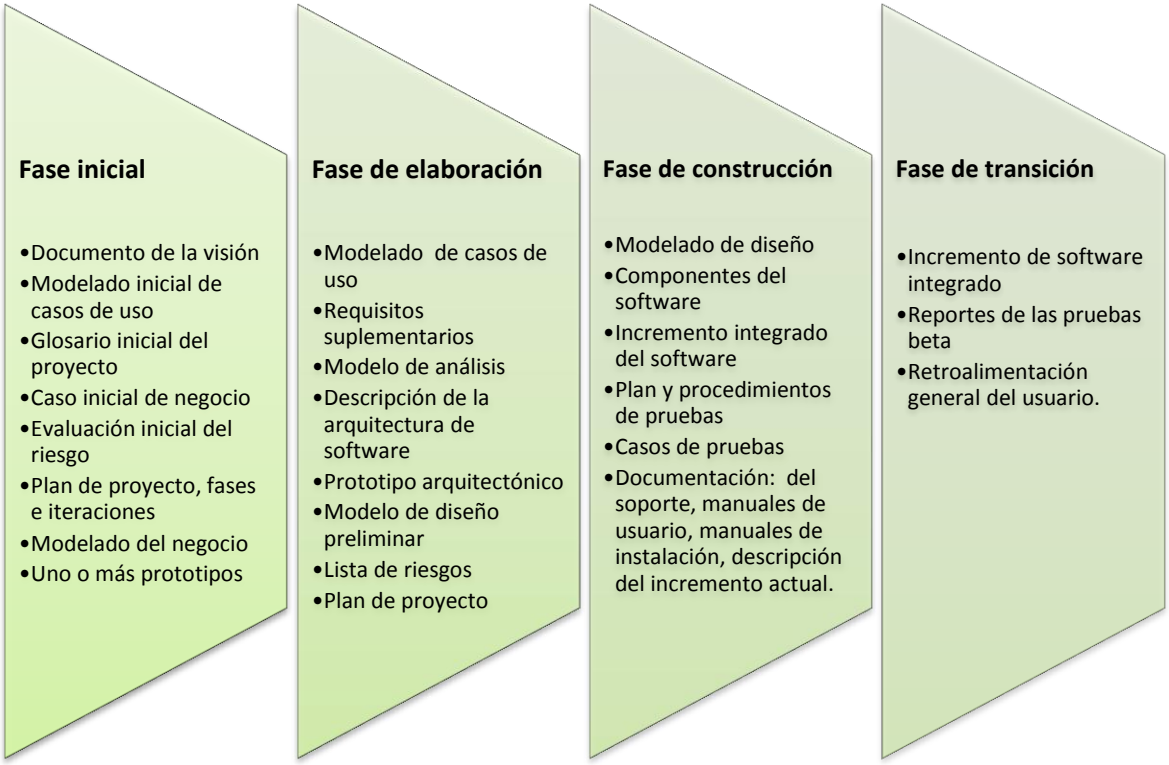


Figura 1.6 Algunos productos de trabajo sugeridos para cada clase del RUP[4]

En cada ciclo de iteración en RUP se hace exigente el uso de artefactos, es decir, un artefacto es un término general para cualquier producto de trabajo: código, gráficas, esquemas de datos, documento de texto, diagramas, modelos, etc. Cabe mencionar que todas las actividades y artefactos son opcionales. En un proyecto donde se está utilizando RUP se debe de seleccionar un subconjunto de artefactos que sirva para tratar necesidades particulares, según sea el proyecto IWeb a desarrollar.

Una de las ventajas que nos proporcionan las prácticas de RUP es que nos da una estructura organizada, mientras que entre sus desventajas es que tiende a ser pesado en documentación y procesos, siendo letal para proyectos pequeños.

Siendo que el proceso unificado es un intento por obtener los mejores rasgos y características de los modelos tradicionales del proceso de software, motivo por el cual se ha elegido esta metodología como guía para implementar la aplicación Web a desarrollar, además de que el proceso unificado reconoce la importancia de la comunicación con el cliente y los métodos directos para describir su punto de vista respecto de un sistema que son los casos de uso, permite cambios futuros y la opción

[4] Pressman, Roger S. *Ingeniería del software, un enfoque práctico*, McGraw-Hill, Sexta edición, Madrid España, 2006. Pág. 72

de la reutilización de artefactos, sugiere un flujo del proceso iterativo e incremental, lo que da la sensación evolutiva que resulta esencial en el desarrollo moderno del software.

El proceso programación extrema (*XP)*

Además de RUP, también es recomendable el modelo de proceso ágil como la programación extrema (o simplemente XP por sus siglas en inglés *extreme* programming), donde su combinación de programar probando primero y desarrollo iterativo son compatibles o idénticas a las prácticas del RUP.

El método XP es una de las metodologías de desarrollo de software más exitosa en la actualidad para proyectos de corto plazo. Es un modelo ágil que busca la satisfacción del cliente y la entrega temprana de software incremental con una simplicidad general del desarrollo. Considera la codificación como la actividad principal en todo el proyecto de software. Esta metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, para llegar al éxito del proyecto.

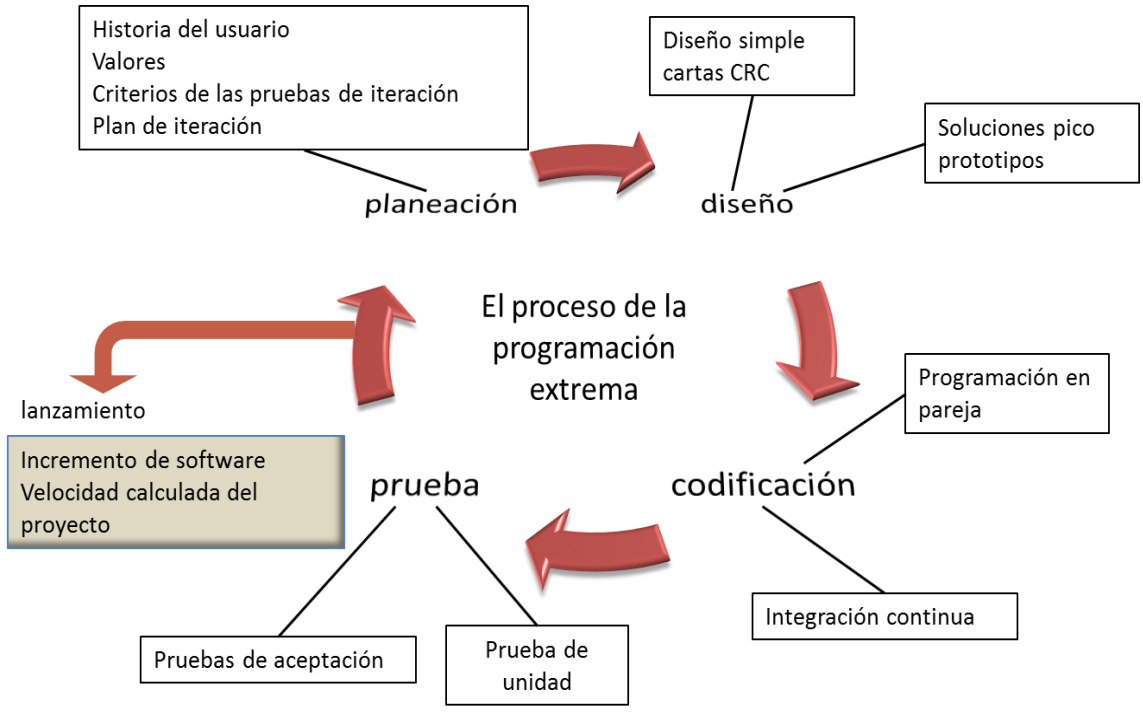


Figura 1.7 El proceso de la Programación Extrema (XP)[5]

[5] Pressman, Roger S. *Ingeniería del software, un enfoque práctico*, McGraw-Hill, Sexta edición, Madrid España, 2006. Pág. 85

Algunas de las características de XP son las siguientes:

- **Pruebas unitarias:** se basa en las pruebas realizadas a los principales procesos, de tal manera que se adelanta en algo hacia el futuro, para obtener posibles errores.
- **Re-fabricación:** se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- **Programación en pares:** una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto, mientras uno conduce, el otro consulta el mapa.

1.2 UML (*Unified Modeling Language*)

Unified Modeling Language UML, es una notación patrocinada por el *Object Management Group* (OMG), que se ha convertido en un estándar para definir, organizar, construir, documentar y visualizar los elementos que forman la arquitectura de un sistema como son: requisitos, arquitectura, construir, código fuente, planes de proyecto, pruebas, prototipos, versiones[6].

Sin embargo, hay que recalcar que UML es una notación y no un proceso o método, es decir, una herramienta útil para representar los modelos del sistema en desarrollo, más no ofrece ningún tipo de guía o criterios acerca de cómo obtener esos modelos.

[6] Larman, Craig. *UML y patrones, una introducción al análisis y diseño orientado a objetos y al proceso unificado.* Pearson Prentice Hall, Segunda edición, Madrid España, 2003. Pág. 10.

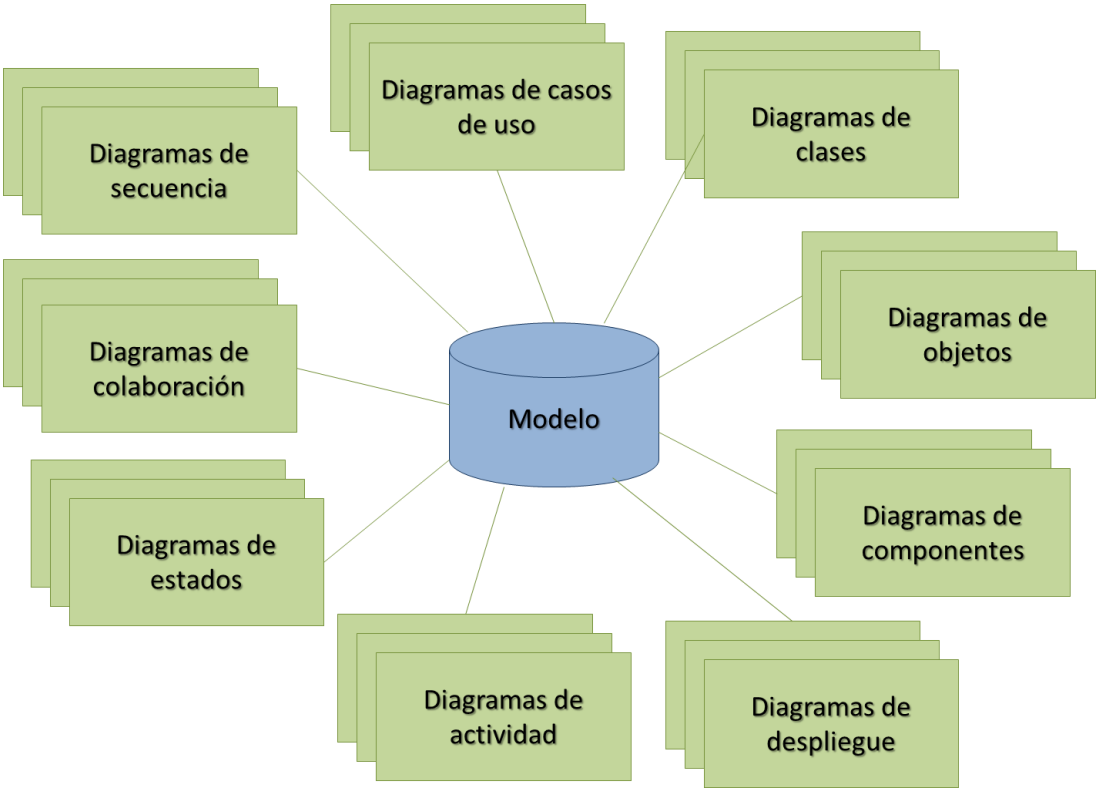


Figura 1.8 Diagramas del Modelo UML

UML se puede usar para modelar distintos tipos de sistemas como son los: sistemas de software, sistemas de hardware y organizaciones del mundo real.

En la figura 1.8, UML presenta los nueve diagramas que se pueden usar para modelar:

a) **Diagramas de secuencia:** para modelar el paso de mensajes entre objetos. Se usa para modelar iteraciones entre objetos de un sistema. Un diagrama de secuencia se modela para cada caso de uso.
b) **Diagramas de caso de uso:** sirven para describir las interacciones del sistema con su entorno, identificando los actores que representan los diferentes roles desempeñados por los usuarios del sistema, y los casos de uso, que corresponden a la funcionalidad que el sistema ofrece a sus usuarios, explicada desde el punto de vista de éstos.
c) **Diagramas de objetos:** sirven para modelar la estructura estática de los objetos en el sistema.
d) **Diagramas de clases**: son una colección de elementos de un modelo estático declarativo, tales como clases, interfaz, y sus relaciones, conectados como un grafo entre sí y con sus contenidos.
e) **Diagramas de componentes:** ilustra los componentes de software que se usarán para construir el sistema.

f) **Diagramas de despliegue:** sirven para modelar la distribución del sistema.
g) **Diagramas de actividad:** son, en esencia, diagramas de flujo, con algunos elementos adicionales que les permiten expresar conceptos como la concurrencia y la división del trabajo.
h) **Diagramas de estados**: sirven para modelar el comportamiento de los objetos en el sistema.
i) **Diagramas de colaboración:** muestran no sólo los mensajes a través de los cuales se produce la interacción entre los objetos, como en los diagramas de secuencia, sino también los enlaces entre los objetos; se trata de una mezcla de diagramas de objetos y diagramas de secuencia.

1.3 Modelado de diseño para las aplicaciones Web

Toda persona que haya navegado en la Web tiene una opinión acerca de lo que hace un buen sitio Web. Los puntos de vista varían enormemente. Algunos usuarios disfrutan los gráficos, otros quieren textos simples o desean una presentación abreviada. Algunos les gustan las herramientas analíticas sofisticadas o los accesos a bases de datos, a otros del gustan las cosas simples. De hecho, la persuasión del usuario de lo que es “bueno”, puede ser más importante que cualquier discusión técnica de la calidad de la una página Web o de un sitio Web.

Por lo que en la figura 1.9 se presenta un árbol con los cinco principales atributos de calidad de una aplicación Web.

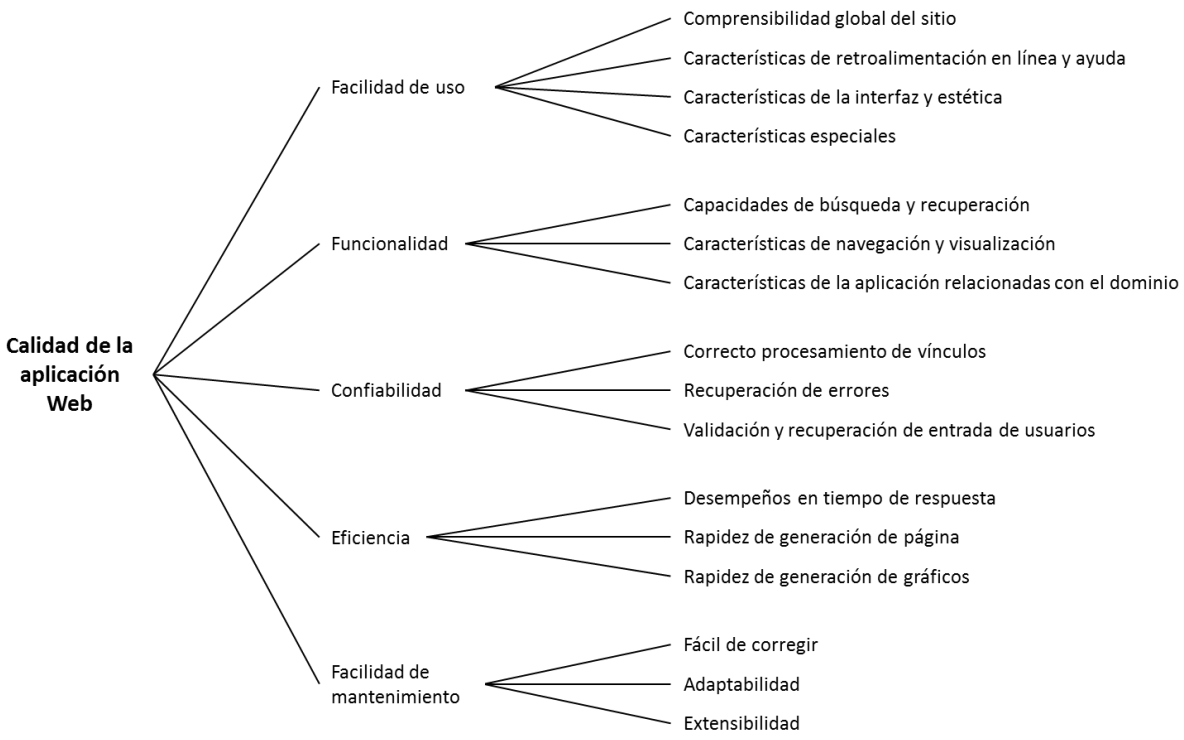


Figura 1.9 Diagrama de calidad de la aplicación Web[7]

También, a los atributos de calidad mencionados anteriormente se le pueden añadir los siguientes atributos mostrados en la tabla 1.4.

Tabla 1.4 Otros atributos de calidad para las WebApps

Atributo	Descripción
Seguridad	La medida clave de la seguridad es la habilidad de las WebApps y su ambiente de servidor de rechazar el acceso no autorizado.
Disponibilidad	Se refiere a tener la característica de estar disponible no sólo en un navegador o a una plataforma.
Escabilidad	Se refiere al ambiente del servidor puede escalarse para mantener a un número considerable de usuarios.
Tiempo en mercado	Es una medida de calidad desde un punto de vista de los negocios.

[7] Pressman, Roger S. *Ingeniería del software, un enfoque práctico*, McGraw-Hill. Sexta edición, Madrid España, 2006. Pág. 568.

1.3.1 Metas de diseño

El modelo de diseño tiene metas establecidas aplicables a todas las WebApps, sin importar el dominio, tamaño o complejidad de la aplicación como son:

- **Simplicidad:** se aplica a los términos de moderación. Los diseñadores no deben proporcionar a los usuarios finales contenidos con demasiada animación, evitar las enormes o extensas páginas.
- **Consistencia:** el contenido se debe de construir de manera consistente, por ejemplo: los documentos de textos y los estilos de fuentes deben de ser los mismos a lo largo de todos los documentos de texto, los gráficos deben de tener una apariencia consistente, tanto color y estilo. El diseño de interfaz debe definir modos consistentes de interacción, navegación y despliegue.
- **Identidad:** la estética, la interfaz y el diseño de navegación del sitio Web deben ser consistentes con el dominio de la aplicación para la cual se va a construir. Es decir, el diseño de cada WebApp debe de estar organizada para lograr diferentes objetivos.
- **Robustez:** los usuarios esperan contenidos y funciones robustas o que sean relevantes para sus necesidades.
- **Navegabilidad:** la navegación debe ser simple y consistente. Debe de estar diseñada de modo que sea intuitiva y predecible. El usuario final debe de entender cómo moverse por el sitio Web sin la necesidad de buscar vínculos.
- **Apariencia visual:** de todas las categorías del software, las aplicaciones Web son las más visuales, más dinámicas y más estéticas. Muchas de las características de diseño están en el ojo de los usuarios, por lo que la forma de la navegación, el equilibrio del texto, la coordinación del color, los gráficos, entre otros, contribuyen al aspecto visual.
- **Compatibilidad:** las exigencias de las aplicaciones Web es la necesidad de que sean compatibles con diferentes tipos equipos, de sistemas operativos, navegadores.

1.3.2 Productos del diseño para aplicaciones Web

Las actividades que se pueden realizar durante el modelado de diseño se pueden representar en una pirámide, como se muestra en la figura 1.10.

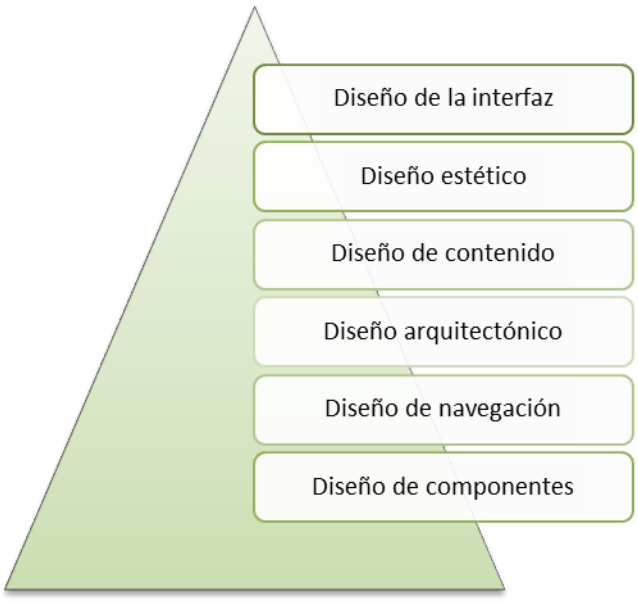


Figura 1.10 Pirámide del diseño para aplicaciones Web[8]

- **Diseño de la interfaz:** describe la estructura y organización de la interfaz del usuario. Incluye una representación de la plantilla de la pantalla.
- **Diseño estético:** es el diseño gráfico, describiendo la apariencia del sitio Web, incluye esquemas de color, platilla geométrica, tamaño de fuente, uso de gráficos y decisiones estéticas relacionadas.
- **Diseño de contenido:** define la plantilla, la estructura y el bosquejo de todo el contenido que se representa como parte de la aplicación Web.
- **Diseño arquitectónico:** identifica la estructura global para el sitio Web.
- **Diseño de navegación:** representa el flujo de navegación entre los objetos de contenido y para todas las funciones de sitio Web.
- **Diseño de componentes:** desarrolla la lógica de procesamiento detallado que se requiere para implementar componentes funcionales.

1.3.3 Arquitectura MVC

La arquitectura modelo, vista y controlador (MVC), es un patrón de arquitectura de software que separa la parte lógica de una aplicación de su presentación. Es muy utilizada dentro de las WebApps,

[8] Pressman, Roger S. *Ingeniería del software, un enfoque práctico*, McGraw-Hill. Sexta edición, Madrid España, 2006. Pág. 573

ya que organiza el código de una aplicación separando el lenguaje de programación del HTML, la interfaz de usuario y la lógica de control en tres componentes:

- **Modelo:** información almacenada en una base de datos o en XML. Junto con las reglas de negocio que transforman esa información, teniendo en cuenta las acciones de los usuarios.
- **Vista:** define la interfaz de usuario, HTML, CSS, etc., enviados en el navegador.
- **Controlador:** responde a eventos y modifica la vista y el modelo.

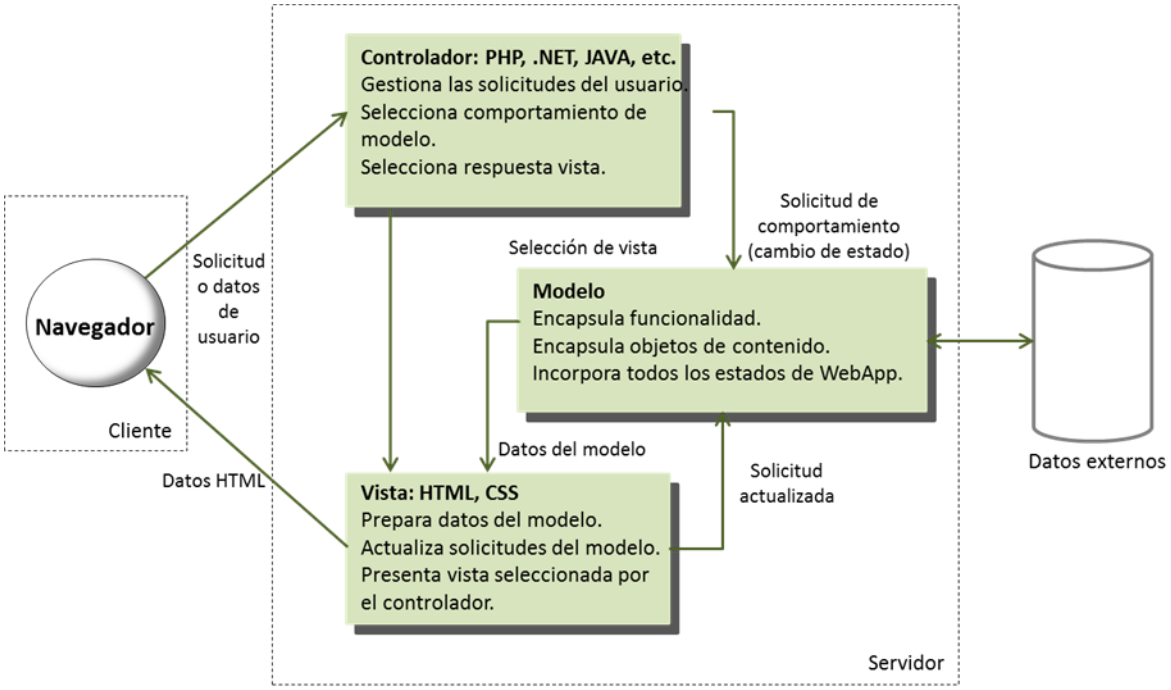


Figura 1.11 Arquitectura MCV[9]

Una de las principales ventajas de esta arquitectura es que se pueden realizar cambios a los componentes de una aplicación independientemente, es decir cuando se realiza un cambio de base de datos, programación o interfaz de usuario sólo tocamos uno de los componentes sin tener la necesidad de conocer cómo funcionan los otros.

[9] Pressman, Roger S. *Ingeniería del software, un enfoque práctico*, McGraw-Hill. Sexta edición, Madrid España, 2006. Páginas 588-590

1.4 Mantenimiento del software

El mantenimiento del software se realiza una vez que se entrega un proyecto de software, su objetivo primordial es el mejorar y optimizar el producto de software, además de que también ayuda a detectar y corregir algunos defectos.

Algunos de los tipos de mantenimiento que se pueden proporcionar son:

- **Perfectivo:** se realizan actividades para mejorar o añadir nuevas funcionalidades requeridas por el usuario.

- **Adaptativo:** se realizan actividades para adaptar el sistema a los cambios (hardware o software, diferente sistema operativo, otro sistema gestor de bases de datos, etc.) en su entorno tecnológico.

- **Correctivo:** se realizan actividades dedicadas a corregir defectos en el hardware o en el software detectados por los usuarios durante la explotación del sistema.

- **Preventivo:** realiza actividades para facilitar el mantenimiento futuro del sistema, como son las revisiones periódicas del producto de software. Este tipo de mantenimiento ayuda a reducir los tiempos que pueden generarse por mantenimiento correctivo.

1.5 Lenguajes de programación para las WebApps

Actualmente existen diferentes lenguajes de programación para el desarrollo de las WebApps. Desde los inicios de Internet, se dieron soluciones mediante lenguajes estáticos. A medida que pasó el tiempo, las tecnologías fueron desarrollándose y surgieron nuevos problemas para dar solución. Esto dio lugar a desarrollar lenguajes de programación para las Web dinámicas, que permitieran interactuar con los usuarios y utilizar sistemas de base de datos.

Algunos de los lenguajes que actualmente se utilizan para el desarrollo de las aplicaciones Web se presentan a continuación:

PHP

Características:	• Es un lenguaje de programación interpretado de alto nivel para el desarrollo de sitios Web • Su sintaxis es muy similar al lenguaje C, Java, Perl • No necesita ser compilado, por lo que cada vez que debe ejecutar un programa, lo interpreta, verificando su sintaxis
Ventajas:	• Muy fácil de aprender • Se caracteriza por ser un lenguaje muy rápido • Soporta la orientación a objetos, clases y herencias • Es un lenguaje multiplataforma: Linux, Windows, entre otros • Capacidad para conectarse a diferentes base de datos: Mysql, PostgreSQL, Oracle, MS SQL Server • Es un programa de código abierto, el cual está disponible para cualquier persona
Desventajas:	• Se necesita un servidor Web • Todo el trabajo lo realiza el servidor y no delega al cliente. Por lo tanto, puede ser más ineficiente a medida que las solicitudes aumenten de número • La legibilidad del código puede verse afectada al mezclar sentencias HTML y PHP

ASP.NET

Características:	• Es un lenguaje comercializado por Microsoft y usado por programadores para desarrollar sitios Web • Es el sucesor de la tecnología ASP • Está creado para desarrollar Web sencillas o grandes aplicaciones • Para sus desarrollos, puede utilizar C# o VB.NET • Los archivos cuentan con la extensión aspx • Para su funcionamiento se necesita tener instalado IIS con el Framework .Net en cual Microsoft Windows 2003 ya lo incluye
Ventajas:	• Completamente orientado a objetos • Controles de usuarios y personalizados • División entre la capa de aplicación o diseño y el código • Facilita el mantenimiento de grandes aplicaciones • Incremento de velocidad de respuesta del servidor • Mayor velocidad • Mayor seguridad

Desventajas:	• Mayor consumo de recursos • No es un programa de código abierto • Es una herramienta de Microsoft, por lo que hay que pagarla para poderla adquirir • Su mantenimiento suele ser más complejo • ASP sólo funciona sobre la plataforma Microsoft

JSP

Características:	• Es un lenguaje para la creación de sitios Web dinámicos, acrónimo de Java Server Pages • Está orientado a desarrollar páginas Web en Java • Es un lenguaje multiplataforma. Creado para ejecutarse del lado del servidor • Creado por Sun Microsystems • Comparte ventajas similares a las de ASP.NET, desarrollado para la creación de aplicaciones Web potentes • Posee un motor de páginas basado en los servlets de Java • Para su funcionamiento se necesita tener instalado un servidor de aplicaciones
Ventajas:	• Permite separar la parte dinámica de la estática en las páginas Web • Es un lenguaje estructurado • Es más fácil de construir y dar mantenimientos a grandes aplicaciones modulares • Ejecución rápida • Crear páginas del lado del servidor • Multiplataforma • Integridad con los módulos de Java • La parte dinámica está escrita en Java • Permite la utilización de los servlets (es un programa que se ejecuta en un servidor).
Desventajas:	• Complejidad de aprendizaje

RUBY

Características:	• Es un lenguaje interpretado de muy alto nivel y orientado a objetos • Es distribuido bajo la licencia de software libre (Open Source) • Permite desarrollos en áreas como: aplicaciones comerciales, acceso a base de datos, proceso y transformación de XML,

	aplicaciones Web
Ventajas:	• Muy fácil de aprender • Permite desarrollar soluciones a bajo costo • Software libre • Es un lenguaje de script, moderno y orientado a objetos, que combina una importante flexibilidad con alta productividad • Se encuentra presente en aplicaciones que van desde el desarrollo Web hasta la simulación de ambientes complejos • Es un lenguaje multiplataforma que se integra en gran cantidad de arquitecturas, incluso puede correr en dispositivos móviles
Desventajas:	• No todos los hostings soportan Ruby

Los conceptos teóricos descritos en el presente capítulo serán de gran utilidad para poder desarrollar los temas posteriores, además de proporcionar soluciones adecuadas para el problema planteado durante el capítulo 2 descrito a continuación.