UNIVERSIDAD NACIONAL DE SAN CRISTÓBAL DE HUAMANGA

FACULTAD DE INGENIERÍA DE MINAS, GEOLOGÍA Y CIVIL

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

"FRAMEWORK PARA IMPLEMENTAR APLICACIONES WEB EN DIFERENTES LENGUAJES DE PROGRAMACIÓN ORIENTADO A OBJETOS, 2017"

Tesis presentada por : Bach. Andree Ochoa Cabrera

Para obtener el título profesional de : Ingeniero de Sistemas

Tipo de investigación **:** Observacional, Retrospectivo, Transversal y Descriptivo.

Área de investigación **:** Ingeniería de software

Asesor **:** MSc**.** Ing. Efraín Elías Porras Flores.

AYACUCHO – PERÚ

2018

DEDICATORIA

A mis padres, en especial a mi madre, con quien estaré eternamente agradecido por cada momento de vida. A mi tía Eufemia, quien vela por mi desde el cielo. A Dios que en todo momento está conmigo.

CONTENIDO

Pág.

CAPÍTULO III

METODOLOGÍA DE LA INVESTIGACIÓN

CAPÍTULO IV

RESULTADOS DE LA INVESTIGACIÓN

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

RESUMEN

En el ámbito de las aplicaciones web han surgido nuevas tecnologías que ahora compiten entre sí, tal es el caso de los lenguajes de programación y los framework. Existen frameworks que agilizan la implementación de aplicaciones web, pero estos están desarrollados en un lenguaje de programación específico, donde el desarrollador debe conocer a profundidad el lenguaje en el que está escrito, para luego recién poder aprender a utilizar el framework, lo cual genera el problema de aprender a cabalidad un nuevo lenguaje y un nuevo framework.

Esta investigación se realizó con el propósito de desarrollar un framework como alternativa para una rápida implementación de aplicaciones web, con la particularidad de que es soportado por los lenguajes de programación orientado a objetos; Php, Java, JavaScript y C#.

La investigación se llevó a cabo bajo el patrón de diseño modelo, vista y controlador (MVC), como arquitectura del framework, y es una investigación de tipo observacional, retrospectivo, transversal y descriptivo.

Desarrollamos el framework utilizando el patrón de diseño MVC, realizamos pruebas en cuatro lenguajes de programación, Php, Java, JavaScript y C#, utilizamos una base de datos relacional, MySql, el marco de trabajo Scrum. Para la recolección de datos se utilizó la técnica de análisis documental.

El estudio ha desarrollado un prototipo de framework, usando una arquitectura técnica similar de un lenguaje a otro, que permite estandarizar una estructura para implementar aplicaciones web.

PALABRAS CLAVES

Framework para aplicaciones web, Patrón de diseño MVC, Scrum.

INTRODUCCIÓN

Según Sánchez (2006), un framework es un conjunto de bibliotecas utilizadas para implementar la estructura estándar de una aplicación, con el propósito de reutilizar código para cada nueva aplicación que se desee crear. Un framework ayuda a que se desarrolle una aplicación de una manera más rápida, evitando los detalles del diseño de la lógica de la aplicación y enfocándose a la lógica del negocio.

Para Alegsa (2010), una biblioteca es una colección o conjunto de subprogramas usados para desarrollar software, no son ejecutables, pero pueden ser usadas por ejecutables para poder funcionar correctamente. Para nuestra investigación, utilizaremos bibliotecas de modelo, vista y controlador. Una biblioteca de modelo es un conjunto de herramientas para ser utilizadas por los objetos que representan los datos de una aplicación; una biblioteca de vista es utilizada por la parte visual de la aplicación; y una biblioteca de controlador que es utilizada para implementar la lógica del negocio.

La presente investigación desarrolla un framework para la implementación de aplicaciones web en diferentes lenguajes de programación. Ya que en la actualidad existen frameworks escritos para un determinado lenguaje, lo cual los hace dependientes a ese lenguaje; esto conlleva a que el desarrollador utilice un framework para cada lenguaje de programación, donde cada uno de ellos tienen estructuras diferentes.

Los objetivos específicos son: (a) Crear los requerimientos del programador con el fin de realizar el diseño del framework. (b) Diseñar la arquitectura técnica del framework con la finalidad de realizar la implementación. (c) Implementar la biblioteca del modelo, con el fin de estandarizar la conexión de la base de datos, el filtro de inyecciones SQL y el manejo de consultas y sentencias utilizando SQL. (d) Implementar biblioteca de la vista, con la finalidad de estandarizar el envío de datos a la interfaz gráfica de usuario, manejo de plantillas y la reutilización de recursos estáticos. (e) Implementar la biblioteca de controlador con el fin de reutilizar funciones repetitivas, recibir datos de la vista y procesarlos según la lógica del negocio.

CAPÍTULO I
PLANTEAMIENTO DE LA INVESTIGACIÓN

1.1. DIAGNÓSTICO Y ENUNCIADO DEL PROBLEMA

En la actualidad existe una gran cantidad de lenguajes de programación, los cuales están aún en constantes cambios; tanto en sintaxis, licencias o costos, que a su vez determinan los costos de los servidores y de los costos de implementación por parte de los programadores. Los programadores utilizan sus conocimientos para poder desarrollar aplicaciones web escribiendo código en el lenguaje que más dominan, dificultando el trabajo en equipo, ya que no todo el grupo de programadores domina un mismo lenguaje de programación, algunas veces utilizando códigos repetitivos y haciendo dependiente a la aplicación web a un determinado lenguaje de programación dificultando su mantenimiento o ampliación del proyecto.

Algunos programadores utilizan frameworks para agilizar sus tareas, estos framework por lo general han sido implementados en un lenguaje específico. Si bien los frameworks pueden agilizar el proceso de implementación de una aplicación web, estos están estrictamente escritos para un determinado lenguaje, lo cual dificulta una eventual migración a otro lenguaje.

Las aplicaciones web son implementadas en un lenguaje de programación determinado, ya sea utilizando un framework o no, estas aplicaciones son dependientes a los cambios que puedan realizarse en el lenguaje implementado.

1.2. DEFINICIÓN DEL PROBLEMA DE INVESTIGACIÓN

PROBLEMA GENERAL

¿Cómo desarrollar un framework que soporte diferentes lenguajes de programación para implementar aplicaciones web, 2017?

PROBLEMAS ESPECÍFICOS

¿De qué manera implementar las bibliotecas de modelo, vista y controlador?

1.3. OBJETIVOS DE LA INVESTIGACIÓN

OBJETIVO GENERAL

Desarrollar un framework que soporte diferentes lenguajes de programación; mediante técnicas e instrumentos, utilizando el marco de trabajo Scrum, un lenguaje de programación orientado a objetos, una base de datos relacional y tecnologías de Internet, con la finalidad de obtener bibliotecas de código reusables para implementar aplicaciones web, 2017.

OBJETIVOS ESPECÍFICOS

a. Crear los requerimientos del programador con el fin de realizar el diseño del framework.
b. Diseñar la arquitectura técnica del framework con la finalidad de realizar la implementación.
c. Implementar la biblioteca del modelo, con el fin de estandarizar la conexión de la base de datos, el filtro de inyecciones SQL y el manejo de consultas y sentencias utilizando SQL.
d. Implementar biblioteca de la vista, con la finalidad de estandarizar el envío de datos a la interfaz gráfica de usuario, manejo de plantillas y la reutilización de recursos estáticos.
e. Implementar la biblioteca de controlador con el fin de reutilizar funciones repetitivas, recibir datos de la vista y procesarlos según la lógica del negocio.

1.4. JUSTIFICACIÓN Y DELIMITACIÓN DE LA INVESTIGACIÓN

1.4.1. IMPORTANCIA DEL TEMA

IMPORTANCIA ECONÓMICA

La presente investigación se llevó a cabo para desarrollar un framework en diferentes lenguajes de programación que pueda agilizar la implementación de aplicaciones web, reduciendo los gastos en relación al tiempo de desarrollo.

IMPORTANCIA TÉCNICA

Al desarrollar el framework, se tiene una alternativa en el uso de framework similares con la particularidad de contar con un framework en diferentes lenguajes de programación con una misma estructura para cada lenguaje y facilitando la migración de una aplicación

web en un lenguaje de programación a otro; que sirve a la comunidad de estudiantes, docentes y desarrolladores de software.

1.4.2. JUSTIFICACIÓN

En la actualidad existen diferentes framework para cada lenguaje de programación, lo que dificulta en utilizar un estándar para todo lenguaje de programación. Es así, que esta investigación busca estandarizar una arquitectura menos compleja, más rápida, donde el desarrollador pueda utilizar una única estructura para implementar aplicaciones web en el lenguaje que más domine, minimizando la dependencia del lenguaje de programación en el cual fue implementado.

1.4.3. DELIMITACIÓN

La investigación se realizará sobre los compontes software para el patrón de diseño modelo, vista y controlador como arquitectura del framework; utilizando los lenguajes de programación: Java, Php, JavaScript y C#.

CAPÍTULO II
REVISIÓN DE LA LITERATURA

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

Según la investigación de Montaldo (2005), concluye que la utilización de frameworks de trabajo, facilitan el desarrollo de la aplicación ya que imponen un orden y una estructura ya pensada para un tipo de arquitectura dada. Por lo que problemas comunes y genéricos ya están resueltos y optimizados por el framework y se permite hacer uso de ellos y especializarlos extendiendo al mismo. Esto brinda un rápido despegue de la aplicación. Mediante el uso de un framework y en conjunto con metodologías y procesos de desarrollo adecuados, se agiliza mucho el proceso de desarrollo de una aplicación, y es más fácil obtener un resultado exitoso.

Según Cuauhtemoc (2014), cuando el software a construir es un framework, que finalmente pretende facilitar la creación de alguna parte de una aplicación, es menester poner especial atención en el diseño del mismo, ya que, para ser útil, debe ser adaptable a una definida variedad de situaciones y, sobre todo, una vez en uso en diferentes proyectos de software, puede ser difícil modificar el framework sin afectar a las aplicaciones que lo usan.

Según Vásquez (2007), menciona que un programa puede ser codificado en una diversidad de lenguajes de programación y que de alguna forma todos los lenguajes de programación son equivalentes, en la actualidad existe también una diversidad en los paradigmas de la programación. A pesar de que los lenguajes de programación están diseñados para comunicar ideas sobre algoritmos entre personas y las computadoras, tienen limitadas características expresivas. Sin embargo, dichos lenguajes permiten un sorprendente y amplio margen de expresión algorítmica, que soporta una gran variedad de aplicaciones de computación a través de varios dominios de aplicación, para conseguir semejante versatilidad, las distintas comunidades de programadores de estos dominios han desarrollado caminos especiales y diferentes, o paradigmas para expresar algoritmos que se ajustan especialmente bien a sus propias áreas de aplicación. Por tanto, en el área

computacional, un paradigma es una forma de representar y manipular el conocimiento. Representa un enfoque particular o filosofía para la construcción de software como el caso de la programación orientado a objetos.

Según Gutiérrez y Vargas (1999), la lógica de programación basada en lenguajes imperativos tuvo su máxima expresión en el paradigma estructurado. Hoy por hoy, la programación estructurada se considera superada por nuevos paradigmas tales como el orientado a objetos. Sin embargo, no debemos perder de vista que la implementación del método de una clase no es más que código basado en programación estructurada. Se contempla que la lógica de la programación es independiente al lenguaje de programación, acercándose más al desarrollo del pensamiento algorítmico. La lógica de la programación estructurada y sus reglas tiene vigencia actual y futura.

2.2. MARCO TEÓRICO

2.2.1. FRAMEWOK

Un framework es un término utilizado en la computación en general, para referirse a un conjunto de bibliotecas, utilizadas para implementar la estructura estándar de una aplicación. Todo esto se realiza con el propósito de promover la reutilización de código, con el fin de ahorrarle trabajo al desarrollador al no tener que rescribir ese código para cada nueva aplicación que desee crear. Existen diferentes frameworks para diferentes propósitos, algunos orientados al desarrollo de aplicaciones web, otros para desarrollar aplicaciones multiplataforma, para un sistema operativo o lenguaje de programación en específico, entre otros. Un framework ayuda a que se desarrolle una aplicación de una manera más rápida, ya que no se pierde tiempo en algunos detalles de diseño que muchas veces quitan más tiempo del que tomo construir en si la lógica de la aplicación. Además, las aplicaciones que se construyen tienen estructuras similares y son más fáciles de mantener. (Sánchez, 2006).

Para Varela (2015), utilizar un framework es producir aplicaciones a partir de este framework. Un framework permite producir aplicaciones con el menor esfuerzo posible, y que también le proporcione un medio para profundizar la comprensión del proyecto del framework, con el fin de que se pueda producir aplicaciones más complejas, sin la necesidad de un esfuerzo excesivo.

Para Rumbaugh, Jacobson y Booch (2007), un framework es una arquitectura genérica que proporciona una plantilla extensible para las aplicaciones dentro de un dominio. Un marco de trabajo es el punto de partida para construir una arquitectura. Típicamente, los elementos se modifican, especializan y extienden para confeccionar la arquitectura genérica de un problema específico.

2.2.2. CLASIFICACION DE LOS FRAMEWORKS WEB

Varela (2005), clasifica los framework de la siguiente manera.

A. SEGÚN SU ESTRUCTURA

Framework con arquitectura en capas: ayuda a estructurar las aplicaciones que se pueden descomponer en grupos de subtareas, con diferentes niveles de abstracciones.

Framework con arquitectura de pipes y filters: se utiliza para estructurar aplicaciones que se pueden dividir en muchas subtareas totalmente independientes, que se deben realizar en una determinada secuencia o en paralelo.

Framework con arquitectura reflexiva: se utiliza en aplicaciones que necesitan considerar futuros ajustes a los cambios ambientales, a la tecnología y a los requisitos. Estos ajustes no afectan a la estructura, ni a la implementación.

Framework con arquitectura Blackboard: ayuda a estructurar aplicaciones complejas que contienen varios subsistemas especializados para diferentes dominios. Estos cooperan entre sí para construir una solución al problema.

B. SEGÚN EL MODO DE EMPLEO

Caja negra (black-box) Conocido como framework de arquitectura de dato-conducido, no es necesario conocer los detalles internos, pues se utiliza cada elemento a nivel de componentes. Los objetos son creados por medio de secuencias que utilizan y/o envían mensajes a cada clase para implementar el código**.**

Caja blanca (White-box) Conocido como framework de Arquitectura de Configuración-Conducida. Las instancias de cada clase deben realizarse por medio de la creación de nuevas clases (a través de la herencia) que utilizan la extensión del Marco de Trabajo.

C. SEGÚN LA CAPA DE APLICACIÓN WEB

Capa de presentación (Front-end) Son todas aquellas tecnologías que corren del lado del cliente, es decir, todas aquellas tecnologías que corren del lado del navegador web, generalizándose más que nada en tres lenguajes, Html, CSS Y JavaScript.

Capa lógica y de acceso a datos (Back-end) Es aquella que se encuentra del lado del servidor, es decir, se encarga de lenguajes como PHP, Python, .Net, Java, etc, se encarga de interactuar con la base de dato, verificar manejo de sesiones de usuarios, montar la página en un servidor, etc.

2.2.3. PROCESO DE DESARROLLO DE LOS FRAMEWORKS

Markiewicz y de Lucena (2000), un framework puede ser personalizado para una aplicación específica, a través de la creación de subclases específicas para la aplicación, siendo estas, subclases específicas de clases abstractas del framework. Los puntos de flexibilidad de un framework son llamados hot-spots. Sin embargo, no todos los aspectos de una aplicación se pueden crear de forma flexible. Por lo tanto, algunas características de un framework no serán modificables. Estos puntos son la inmutabilidad del núcleo de un framework, también llamados frozen-spots, es decir, partes que son compartidas por todas las aplicaciones generadas a partir del framework.

Fayad y Cline (1996), la característica principal al desarrollar un framework es la generalidad en relación a los conceptos y funcionalidades del dominio tratado. Por otra parte, es esencial que la estructura producida sea flexible, es decir, que esté presente las características mutabilidad y extensibilidad. Esta alterabilidad, se refiere a la capacidad de cambiar la presente funcionalidad, sin consecuencias imprevistas sobre el conjunto de la estructura, ya la extensibilidad se refiere a la posibilidad de extender la funcionalidad existente, sin consecuencias no deseadas en el conjunto de la estructura.

Según Mattson (1996), en el proceso general de desarrollo de frameworks, elementos comunes son identificados, tales como:

- Análisis del dominio del problema. Esto se lleva a cabo de forma sistemática o mediante el desarrollo de una o unas pocas aplicaciones de dominio, donde se encuentran las abstracciones clave.
- La primera versión del framework se desarrolla utilizando las abstracciones que se encuentran.
- Se desarrollan una o más aplicaciones basadas en el framework. La prueba es importante para asegurarse de que el framework es realmente reutilizable.
- Los problemas encontrados durante el desarrollo de la aplicación basada en el framework son identificados y se resuelven en la próxima versión.
- Después de repetir el ciclo varias veces, el framework alcanzará un nivel aceptable de madurez, lo que permite su re-utilización por múltiples usuarios.

Johnson (1993), afirma que el desarrollo de un framework para un dominio de aplicación es el resultado de un proceso de aprendizaje acerca de este dominio, que tiene lugar en concreto a través del desarrollo de aplicaciones, o el estudio de las aplicaciones desarrolladas. La abstracción de dominio, que es el framework en sí, se obtiene a partir de la generalización de casos específicos, en este caso las aplicaciones.

2.2.4. PATRON DE DISEÑO MODELO, VISTA Y CONTROLADOR

Para Gamma, Helm, Johnson y Vlissides (2003), el patrón de diseño modelo, vista y controlador (MVC) consiste en tres tipos de objetos. El Modelo es el objeto de aplicación, la Vista es su representación en pantalla y el Controlador define el modo en que la interfaz reacciona a la entrada del usuario. Antes de MVC, el diseño de interfaces de usuario tendría a agrupar estos tres en uno solo, MVC los separa para incrementar la flexibilidad y reutilización.

Sánchez (2006), define MVC como un patrón de diseño de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos de forma que las modificaciones al componente de la vista, o a cualquier parte del sistema puedan ser hechas con un mínimo impacto en el componente del modelo de

datos o en los otros componentes del sistema. Este patrón cumple perfectamente el cometido de modularizar un sistema.

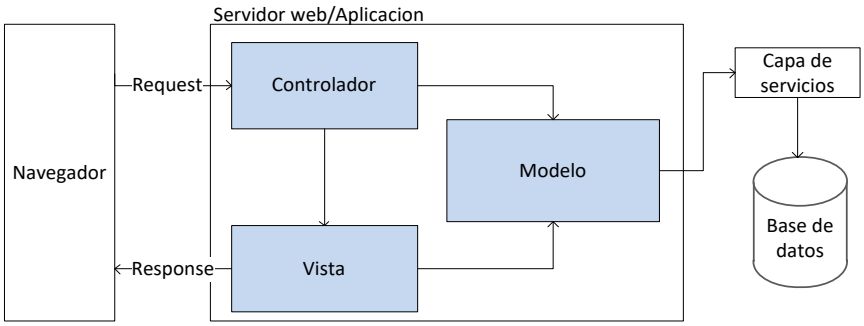


Figura 2.1. Arquitectura MVC (Walls, 2005)

Bahit (2014), define brevemente los tres niveles de abstracción del MVC:

Modelo: Es el encargado de acceder de forma directa a los datos actuando como intermediario con la base de datos.

Vista: Es la encargada de mostrar la información al usuario de forma gráfica y legible.

Controlador: es el intermediario entre la vista y el modelo; y es quien controla las interacciones del usuario solicitando los datos al modelo y entregándolos a la vista.

Marques (2011), afirma que el patrón de diseño MVC es una especialización de un modelo de capas, con la diferencia que se usa para entornos web como patrón por excelencia.

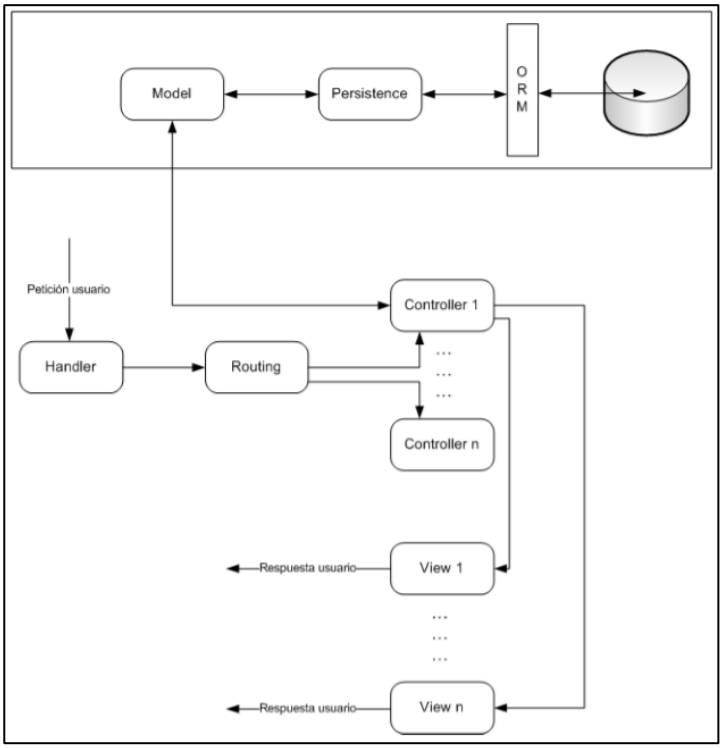


Figura 2.2. Ciclo de vida de un patrón MVC web (Márquez, 2011)

2.2.5. BIBLIOTECA

Según Alegsa (2010), una biblioteca es una colección o conjunto de subprogramas usados para desarrollar software. En general, las bibliotecas no son ejecutables, pero sí pueden ser usadas por ejecutables que las necesitan para poder funcionar correctamente. En español, biblioteca también suele referirse como "librería", una mala traducción de library, que es en realidad biblioteca.

Para Regalut (2012), una biblioteca es un kit de herramientas software pequeño y autónomo que ofrece una funcionalidad muy específica al usuario. Normalmente se usa junto con otras librerías y herramientas para hacer una aplicación completa, ya que por lo general las bibliotecas no son ejecutables, pero sí pueden ser usadas por ejecutables que las necesiten para poder funcionar.

A. BIBLIOTECA DE MODELO

Es un conjunto de herramientas preparadas para ser utilizadas por los objetos que representan los datos de una aplicación específica. Actúa como intermediario en la interacción con la base de datos proveyendo de métodos funcionales como mapeo de datos, consultas y sentencias.

B. BIBLIOTECA DE VISTA

Es un conjunto de herramientas preparadas para ser utilizadas por la parte visual de la aplicación, permiten transportar la información procesada o requerida hacia la vista.

C. BIBLIOTECA DE CONTROLADOR

Es un conjunto de herramientas preparadas para ser utilizadas al implementar la lógica del negocio, y reutilizar lo métodos de la lógica de la aplicación. Estas bibliotecas son intermediarias entre las solicitudes de la vista y la interacción con el modelo.

2.2.6. TECNOLOGÍAS DE INTERNET

Internet, también llamado autopista de información, designa un conjunto de redes informáticas relacionadas entre sí, para permitir que los usuarios puedan comunicarse entre sí; es una red abierta. Su principio básico es la trasmisión de datos de manera fiable entre ordenadores (Colección Esencial, 2011).

Luján (2001) asevera que contrario a otros servicios online, que se controlan de forma centralizada, la Internet posee un diseño descentralizado. Dado que cada ordenador (host) en la Internet es independiente. Los operadores pueden elegir qué servicio usar y qué servicios locales proporcionar.

Cafassi (1998) asevera que Internet es una tecnología. Sin embargo, no es la acepción más correcta y su consecuencia se ve a la hora de analizar las implicaciones sociales que conlleva su uso.

A. APLICACIÓN WEB

Para Luján (2001), una aplicación web es un tipo especial de aplicación cliente/servidor, donde el cliente (el navegador, explorador o visualizador) como el servidor (el servidor web) y el protocolo mediante el que se comunican (HTTP) están estandarizados y no han de ser creados por el programador de aplicaciones.

Para Vértice (2010), una aplicación web es una aplicación informática que se utiliza accediendo a través de un sistema de red como puede ser internet o una intranet. La estructura común se basa en tres capas: Cliente, servidor y sistema de administración de base de datos.

a. CLASIFICACIÓN DE UNA APLICACIÓN WEB

Tahuiton (2011), afirma que existen cuatro tipos principales de aplicaciones web para todo tipo de necesidades:

- **Estáticas:** Este tipo de aplicaciones representan a la primera generación y están compuestas por páginas Web estáticas, imágenes y texto, pero no cuentan con una lógica de negocio. Dentro de este tipo de aplicaciones se encuentran las páginas personales.

- **Orientadas a servicios:** este tipo de aplicaciones intentan ofrecer un servicio especializado, por lo cual implementan una lógica de negocio acorde al servicio ofrecido. Durante su mantenimiento los desarrolladores necesitan comprender claramente la lógica de negocio. Un ejemplo de este tipo de aplicaciones son los servidores de correo electrónico.

- **De datos:** este tipo de aplicaciones está enfocado a proveer una interfaz para acceder a una gran cantidad de datos y no en la lógica de negocio, por lo tanto, los desarrolladores necesitan comprender el flujo de datos. Un ejemplo de este tipo de aplicaciones son los catálogos en línea de las bibliotecas.

- **Sistemas de información:** combinan las aplicaciones orientadas a servicios y de datos. Los desarrolladores necesitan comprender claramente el flujo de datos y la lógica de negocio (especialmente en la manipulación de los datos). Ejemplos de este tipo de aplicaciones son la banca en línea y los portales de comercio electrónico.

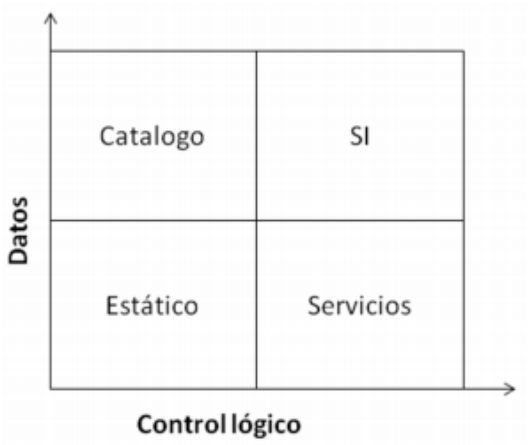


Figura 2.3. Taxonomía de las aplicaciones web (Tahuiton, 2011)

b. CARACTERÍSTICAS DE UNA APLICACIÓN WEB

Ruiz (2011), menciona que una aplicación Web presenta muchas características, pero hay cuatro que se pueden considerar como las más importantes:

- **Capacidad de gestionar información**: como toda aplicación, las aplicaciones Web permiten consultar, agregar, modificar y eliminar información.

- **Multiplataforma**: Dado que las interfaces de usuario son creadas mediante páginas Web, no hay restricciones para los clientes que quieran interactuar con el sistema, sólo necesitan utilizar un navegador Web como Internet Explorer, Firefox, Chrome, etc. funcionando sobre cualquier sistema operativo.

- **Disponibilidad permanente**: Dado que las aplicaciones se encuentran funcionando en servidores locales o en Internet, están disponibles para ser utilizadas las 24 horas, sin restricciones de tiempo.

- **Privacidad de la información**: Existe la opción para que los usuarios provean una contraseña de acceso para poder interactuar con el sistema, de esta forma se garantiza su identidad y sólo podrán acceder a la información para la que están autorizados.

B. SERVIDOR WEB

Mateu (2004), Un servidor web es un programa que atiende y responde a las diversas peticiones de los navegadores, proporcionándoles los recursos que solicitan mediante el protocolo HTTP o el protocolo HTTPS (la versión segura, cifrada y autenticada de HTTP).

C. PROTOCOLO

"Proviene del griego <<protókolon>>, que significa <<la primera hoja o tapa, encolada, de un manuscrito importante con notas sobre su contenido>>" (Antonio, 2001).

Según Gutiérrez y Tena (2003), un protocolo es un conjunto definido de etapas, que implica a dos o más partes, designado para realizar una tarea específica.

D. HTTP

Luján (2001), el protocolo forma parte de la familia de protocolos de comunicación TCP/IP, que son los empleados en internet. Permiten la conexión de sistemas heterogéneos, facilitando el intercambio de información ente distintos ordenadores.

"El protocolo de transferencia de hipertexto es un sencillo protocolo cliente–servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP" (Romero, 1997, p. 203).

Mateu (2004), menciona que las peticiones en HTTP pueden realizarse usando dos métodos. El método GET, en caso de enviar parámetros junto a la petición, las enviaría codificadas en la URL, éstos se añaden a la URL detrás del nombre del recurso, separados

de éste por un carácter ¿. Los diferentes parámetros se separan entre sí por el carácter &. Por su parte, el método POST, en caso de enviarlos, lo haría como parte del cuerpo de la petición, donde se envía información adicional.

E. TCP

Según Colección Esencial (2011), este protocolo, descompone el mensaje en paquetes y asegura la fiabilidad de la transmisión. Cuando los paquetes llegan a su destino, se agrupan automáticamente para formar un único mensaje, idéntico al original.

TCP (Transmission Control Protocol), utiliza mensajes IP para lograr una transferencia de datos libre de errores. Ambos establecen un diálogo con otro sistema a base de enviar servicios de mensajes IP (Romero, 1997, p.22).

F. IP

En el protocolo de internet, cada equipo posee una sola dirección única en la red a la que pertenece y cada red posee una dirección única en internet. Por lo tanto, cada ordenador tiene asignado un nombre y un número IP, y en algunos casos un nombre de dominio (Colección Esencial, 2011).

"IP (Internet Protocol) es capaz de enviar mensajes de pequeño tamaño (denominado datagrama) entre dos ordenadores conectados en red. No ofrece garantías de que los mensajes alcancen su destino, debido a los posibles fallos de las redes de comunicaciones." (Romero, 1997, p.20).

2.2.7. BASE DE DATOS RELACIONAL

Las bases de datos relacionales poseen ventajas como la independencia de datos y los programas; menor redundancia de datos, integridad de los datos, coherencia de los resultados, mayor seguridad en los datos, datos más documentados, acceso a datos más eficiente, reducción del espacio de almacenamiento, acceso simultaneo a los datos; pero tiene desventajas como instalación costosa, requiere personal calificado, implantación larga y difícil, ausencia de estándares reales, falta de rentabilidad a corto plazo (Nevado, 2010).

Jiménez (2014) afirma que una base de datos relacional es aquella que representa los datos y las relaciones entre los datos mediante una colección de tablas, las bases de datos relacionales poseen ventajas sobre otros tipos de bases de datos en aspectos como la independencia física, independencia lógica, flexibilidad, uniformidad, sencillez.

2.2.8. LENGUAJES DE PROGRAMACIÓN ORIENTADO A OBJETOS

Weitzenfeld (s.f.), asevera que los lenguajes de programación orientados a objetos varían en sus estructuras y flujos de control. Y a pesar de que estos estén diseñados para un mismo tipo de programación, existen aspectos que hacen que ciertos lenguajes ofrezcan mejor apoyo que otros durante el desarrollo de un sistema de software.

"Es un conjunto de símbolos, palabras y reglas que permiten implementar un algoritmo en una computadora. Dicha implementación se conoce como programa y se escribe como una secuencia de frases del lenguaje de programación" (Osorio, 2008, p.368).

Los lenguajes de programación orientados a objetos ofrecen las ventajas potenciales de los códigos reutilizables, costos inferiores, menos pruebas y rapidez en la puesta en marcha. Los programadores pueden combinar, modificar e integrar módulos pre-desarrollados en un programa unificado. Entre los más importantes están: Smalltalk, C++ y Java (Stair y Reynolds, 1999, p.461).

Para Craing (2002), los lenguajes orientados a objetos están definidos por un conjunto de propiedades. La medida en la que un lenguaje particular satisfaga estas propiedades define en cuánto es un lenguaje orientado a objetos.

2.2.9. MARCO DE TRABAJO SCRUM

Según Palacio (2008), Scrum es una metodología ágil de gestión de proyectos cuyo objetivo primordial es elevar al máximo la productividad de un equipo. Reduce al máximo la burocracia y actividades no orientadas a producir software que funcionen y produce resultados en periodos muy breves de tiempo. Como método, Scrum enfatiza valores y prácticas de gestión, sin pronunciarse sobre requerimientos, prácticas de desarrollo, implementación y demás cuestiones técnicas. Más bien delega completamente en el equipo la responsabilidad de decidir la mejor manera de trabajar para ser lo más productivos posibles.

Scrum define un conjunto de prácticas y roles, que pueden tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. Se basa en: el desarrollo de los requisitos del proyecto en bloques temporales cortos y fijos; la priorización de los requisitos por valor para el cliente y coste de desarrollo en cada iteración; potenciación del equipo, que se compromete a entregar unos requisitos y para ello se le otorga la autoridad necesaria para organizar su trabajo; y la sistematización de la colaboración y la comunicación tanto entre el equipo y con el cliente (Münch, 2010).

A. ROLES SCRUM

ScrumStudy (2016), menciona que entender los roles y responsabilidades definidos en un proyecto Scrum es muy importante a fin de asegurar la implementación exitosa del método de Scrum. Los roles de Scrum se dividen en dos grandes categorías:

ROLES CENTRALES

a. **Propietario del producto (Product Owner):** Es la persona responsable de lograr el máximo valor empresarial para el proyecto. Este rol también es responsable de la articulación de requisitos del cliente y de mantener la justificación del negocio para el proyecto. El propietario del producto representa la voz del cliente (ScrumStudy, 2016).

b. **Facilitador (Scrum Master):** Es el líder del equipo responsable de que todos los participantes sigan las reglas y los procesos de Scrum. Asegura la disponibilidad de las herramientas en cada iteración, facilita las reuniones de Scrum y está encargado de eliminar los impedimentos que el equipo tenga para continuar con su trabajo (Pham, 2010).

c. **Equipo scrum (Team):** Es el grupo o equipo de personas responsables de la comprensión de los requisitos especificados por el propietario del producto y de la creación de los entregables del proyecto (ScrumStudy, 2016).

   Es el grupo de personas, responsable de transformar el Backlog de la iteración en un incremento de la funcionalidad del software. Tiene autoridad para reorganizarse y definir las acciones necesarias o sugerir remoción de

impedimentos: Auto-gestionado, Auto-organizado, Multi-funcional (Palacios, 2010).

ROLES NO CENTRALES

a. **Los socios (Stakeholder):** Es un término colectivo que incluye a los destinatarios finales, como clientes, usuarios, patrocinadores e interesados, con frecuencia interactúan con el equipo principal de Scrum, e influyen en el proyecto a lo largo de su desarrollo. Lo más importante es que el proyecto produzca beneficios de colaboración para los socios (ScrumStudy, 2016).

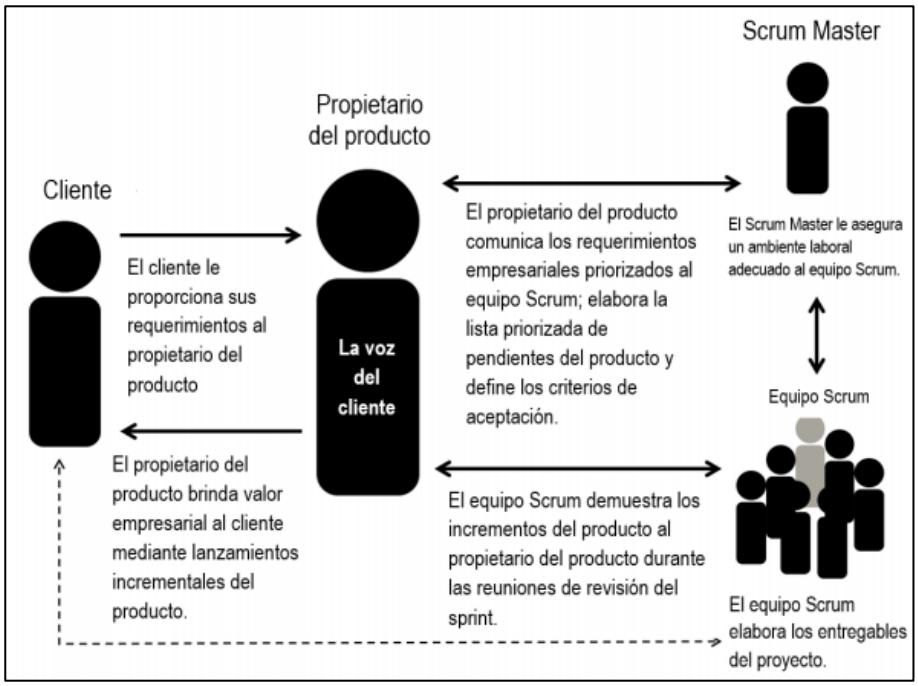


Figura 2.4. Roles de scrum, descripción general (ScrumStudy, 2016).

B. ELEMENTOS SCRUM

El proceso de Scrum posee una mínima cantidad necesaria de elementos formales para poder llevar adelante un proyecto de desarrollo (Alaimo, 2013).

a. **Pila del producto (Product Backlog):** Representa la visión y expectativas del cliente respecto a los objetivos y entregas del proyecto. El cliente es el responsable de crear y gestionar esta lista, y plasma en ella las expectativas de los resultados a obtener. Contiene los requisitos de alto nivel, el esfuerzo estimando en horas de trabajo para cada requisito, las posibles iteraciones, las entregas a realizar, y si es necesario que se considere dentro de cada requisito (Pham, 2010).

b. **Pila de la iteración (Sprint Backlog):** Elaborada en la reunión de planificación del Sprint (Sprint Planning Meeting), es el plan para completar los requisitos seleccionados para la iteración y que se compromete a demostrar al cliente al finalizar la iteración, en forma de un entregable. Permite identificar las tareas de negocio y los problemas que se presentan a lo largo de una iteración (Palacios, 2010).

c. **Gráfico de trabajo pendiente (Burndown charts):** Está diseñado para ayudar al equipo en la monitorización de su progreso y para ser el indicador principal que informará sobre sus posibilidades de alcanzar su compromiso al finalizar el sprint. El formato clásico requiere que el equipo estime la duración de cada tarea en horas de forma diaria. El burndown deberá completarse de forma tal que grafique cuántas horas de trabajo restan para concluir el sprint.

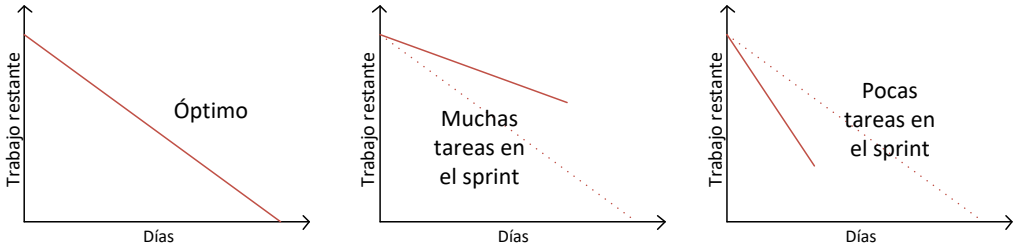


Figura 2.5 Gráficos de Burdown chart (Kniberg, 2007)

C. PROCESO SCRUM

Según Pham (2010), está basada en un proceso constructivo iterativo e incremental donde las iteraciones tienen duración fija.

a. **Primera reunión Scrum (Scrum meeting):** Se realiza una primera reunión donde serán generados los acuerdos y compromisos entre el equipo de desarrollo, se elige al Product Owner y se identifica al Scrum Master. Esta reunión es la parte estratégica y enfocada en el "¿Qué trabajo será realizado?", donde el equipo se comprometerá a transformar los requerimientos del cliente en un producto funcional y utilizable (Alaimo, 2013).

b. **Planeación del sprint (Sprint Planning):** Consiste en llevar a cabo la redacción de historias de usuario partiendo de los requerimientos del cliente.

Es conveniente contar con el cliente en esta etapa. Sin lugar a dudas el equipo de Scrum completo debe participar (Hundermark, 2009).

El equipo de desarrollo determinará la forma en la que llevará adelante el trabajo, se responde a la pregunta "¿Cómo será realizado el trabajo?". Esto implica la definición inicial de un product backlog, el cual será refinado durante el Sprint mismo y la identificación de las tareas con la estimación del tiempo de desarrollo que el equipo en su conjunto tendrá que llevar a cabo (Alaimo, 2013).

c. **Ejecución del sprint (Sprint execution):** Es un proceso de desarrollo incremental e iterativo, las iteraciones en Scrum se conocen como Sprints. Esto significa que el producto se construye en incrementos funcionales entregados en periodos cortos para obtener realimentación frecuente. En general, Scrum recomienda una duración de Sprint de entre 1 y 4 semanas, siendo 2 o 3 semanas lo más habitual que encontraremos en la industria. Una de las decisiones que debemos tomar al comenzar un proyecto o al adoptar Scrum es justamente la duración de los Sprints. Luego, el objetivo será mantener esta duración constante a lo largo del desarrollo del producto, lo que implicará que la duración de una iteración no cambie una vez que sea establecida (Alaimo, 2013).

d. **Reunión diaria (Daily meeting):** En esta reunión, el equipo comparte información relativa al desarrollo y colaboración para hacer las adaptaciones necesarias, aumentando su productividad. En esta reunión se tendrá como referencia el backlog y el burdown con la reunión de la información anterior, la reunión no debe durar más de 15 minutos y se debe contestar las preguntas: ¿qué se ha hecho de nuevo con respecto a la última reunión diaria?, ¿qué será lo siguiente a realizar? Y ¿qué problemas hay para realizarlo? (Trigas, 2012).

e. **Revisión del sprint (Spritn review):** Al finalizar cada Sprint se realiza una reunión de revisión del Sprint, donde se evalúa el incremento funcional potencialmente entregable construido por el equipo de desarrollo. En esta reunión el Equipo Scrum y los socios revisan el resultado del Sprint, donde los

interesados utilizan y evalúan durante esta misma reunión, aceptando o rechazando así las funcionalidades construidas (Alaimo, 2013).

f. **Retrospectiva (Retrospective):** En esta reunión el equipo debatirá temas relacionados con el sprint recientemente finalizado y los cambios que se podrían hacer para mejorar el próximo sprint y que sea más productivo, se responden las preguntas: ¿qué hemos hecho bien?, ¿qué debemos mejorar? y ¿qué no debemos seguir haciendo? (Trigas, 2012).

g. **Refinamiento (Refinament):** Es una actividad constante a lo largo de todo el Sprint, aunque algunos equipos prefieren concentrarla en una reunión que se realiza durante el Sprint y en función de las necesidades. Su objetivo es profundizar en el entendimiento de los requerimientos que se encuentran más allá del Sprint actual y así dividirlos en tareas más pequeñas, y estimarlos. Idealmente se revisan y detallan aquellos que potencialmente se encuentren involucrados en los próximos dos o tres Sprints (Alaimo, 2013).

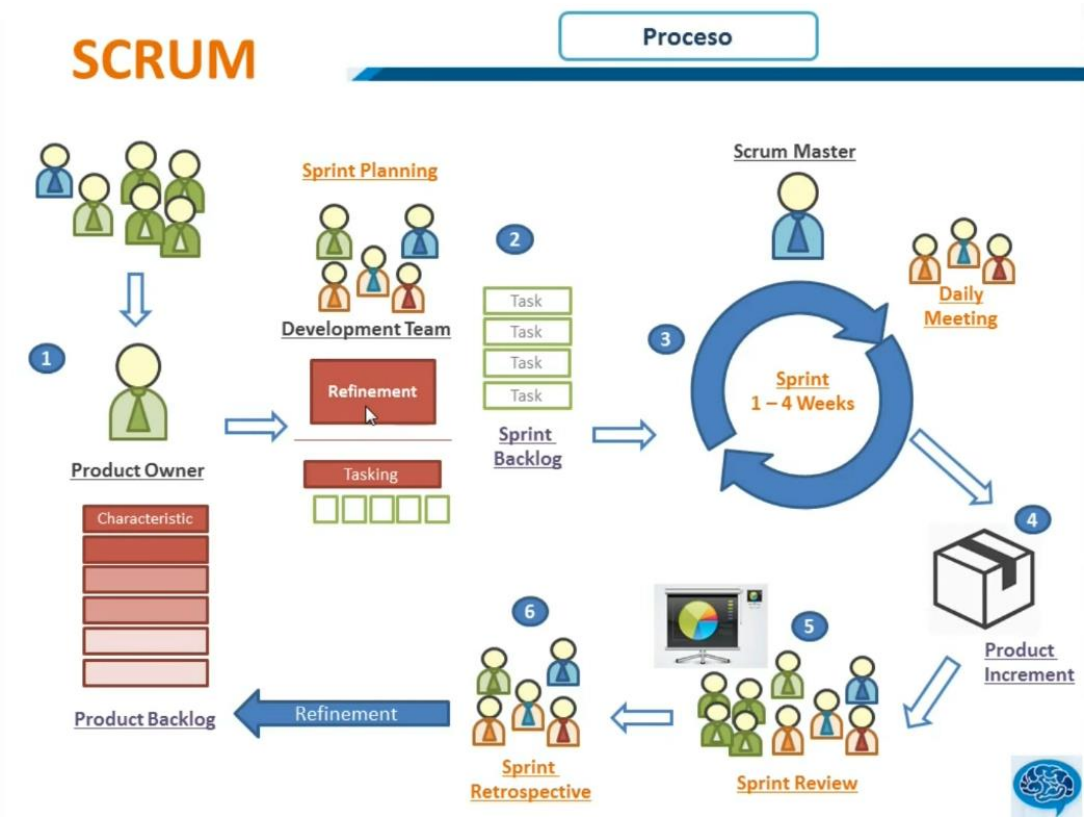


Figura 2.6 Proceso Scrum (Rodríguez, 2012).

2.2.10. ARTEFACTOS INTEGRADOS CON SCRUM

A. REQUERIMIENTOS DEL DOMINIO

Somerville (2005). Clasifica los requerimientos como funcionales y no funcionales, o como requerimientos del dominio. Los requerimientos del dominio se derivan del dominio de la aplicación del sistema más que de las necesidades específicas de los usuarios. Normalmente incluyen terminología especializada del dominio o referencias a conceptos del dominio. Pueden ser requerimientos funcionales o no funcionales.

B. HISTORIAS DE USUARIO

ScrumStudy (2016), define que las historias de usuario se apegan a una estructura específica predefinida y son una forma simple de documentar los requerimientos y funcionalidades que desea el usuario final. Una historia de usuario incluye tres elementos sobre el requerimiento: Como, quiero y para. Los requerimientos expresados en las historias de usuario son oraciones breves, sencillas y fáciles de entender. El formato estándar predefinido da como resultado en una comunicación mejorada entre los socios, así como en mejores estimaciones por parte del equipo.

C. DIAGRAMA DE FLUJO

Ramonet (2013), los diagramas de flujo es un método para realizar el diseño gráfico de procesos, ilustra la secuencia de las operaciones que se realizarán para conseguir la solución de un problema. Los diagramas de flujo se dibujan generalmente antes de comenzar a programar el código frente a la computadora, facilitan la comunicación entre los programadores y la gente del negocio. Estos diagramas de flujo desempeñan un papel vital en la programación de un problema y facilitan la comprensión de problemas complicados y sobre todo muy largos. Una vez que se dibuja el diagrama de flujo, llega a ser fácil escribir el programa en cualquier lenguaje de alto nivel.

D. ARQUITECTURA DE SOFTWARE

Una Arquitectura de Software, también denominada Arquitectura lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco. (Kruchten y Philippe, 1995).

Una arquitectura de software se selecciona y diseña con base en objetivos y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. Unas arquitecturas son más recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para determinadas arquitecturas. Por ejemplo, no es viable emplear una arquitectura de software de tres capas para implementar sistemas en tiempo real (Jacobson, Ivar, Booch, y Rumbaugh, 2000).

E. DIAGRAMA DE CLASES

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro (Jacobson, 1999).

2.2.11. POBLACIÓN

La población se define como la totalidad del fenómeno a estudiar donde las unidades de población poseen una característica común la cual se estudia y da origen a los datos de la investigación (Tamayo y Tamayo, 1997).

La población es el conjunto de todos los casos que concuerdan con una serie de especificaciones, podemos decir que la población es la totalidad del fenómeno a estudiar, en donde la unidad de población posee una característica común la cual estudia y da origen a los datos. (Hernández, 2000).

2.2.12. MUESTRA

La muestra se define como un subgrupo de la población. Para delimitar las características de la población. (Hernández, 2000).

Bavaresco (2006), refiere que “cuando se hace difícil el estudio de toda la población, es necesario extraer una muestra, la cual no es más que un subconjunto de la población con la que se va a trabajar”.

2.2.13. MUESTREO

MUESTREO POR CONVENIENCIA

También recibe el nombre de sesgado. El investigador selecciona los elementos que a su juicio son representativos, lo que exige un conocimiento previo de la población que se investiga (Tamayo y Tamayo, 1997).

Según Bavaresco (2006), los sujetos de una investigación específica, son seleccionados para el estudio porque son más fáciles de reclutar y el investigador no está considerando las características de inclusión de los sujetos representativos de toda la población.

CAPÍTULO III
METODOLOGÍA DE LA INVESTIGACIÓN

3.1. TIPO DE INVESTIGACIÓN

Según Hernández, Fernández y Baptista (2006), define la investigación observacional como aquella investigación que se sustenta en el uso de técnicas que permiten al investigador adquirir información por medio de la observación directa y el registro de fenómenos, pero sin ejercer ninguna intervención.

Según Salinas (2007), la investigación retrospectiva estudia o analiza los casos, fenómenos, características, eventos, situaciones, relaciones entre causa y efecto, etc, presentes y pasados, que pueden o no haber sido tomados por el propio investigador o tomarse de archivos, registros, informes, investigaciones, etc.

Según Hernández, Fernández y Baptista (2006), el tipo de investigación transaccional o transversal recolecta datos en un solo momento, en un tiempo único. Su propósito es describir variables, y analizar su incidencia e interrelación en un momento dado.

El tipo de investigación según la intervención del investigador es observacional, porque se recolecta métodos que existen para desarrollar el framework, según la planificación de las mediciones es retrospectivo, porque se toma la información en base a frameworks similares existentes y la experiencia del investigador como desarrollador de aplicaciones web; y según el número de mediciones de la variable de estudio es transversal, porque se seleccionan métodos de los lenguajes de programación orientado a objetos con poca evolución en el tiempo.

3.2. NIVEL DE INVESTIGACIÓN

Para Salinas (2007), la investigación descriptiva es aquella que se refiere a la descripción de algún objeto, sujeto, fenómeno, etc. en total o parte del mismo, tal como un aparato, técnica, método, procedimiento, proceso, etc, es decir, que en este tipo de investigación se parte del supuesto que la descripción que se va a realizar no ha sido hecha

anteriormente. Sin embargo, se acepta como perfectamente válida y original, la descripción de alguna variación o modificación de algo ya descrito, por ejemplo, en un aparato o técnica o proceso, se pueden modificar sus componentes y así obtener resultados diferentes y mejores a los anteriormente descritos. Para realizar una investigación descriptiva de cualquier naturaleza hay que realizar la búsqueda documental sobre los antecedentes del tema, como se puede observar, la investigación descriptiva incluye a la investigación documental.

El nivel de la investigación es descriptivo, porque se especifica la estructura estándar del framework, las reglas de uso, y se desarrolla el framework describiendo su funcionamiento como herramienta para crear nuevas aplicaciones web a partir de ella.

3.3. POBLACIÓN Y MUESTRA

POBLACIÓN

La población estuvo compuesta por los framework de diferentes lenguajes de programación orientado a objetos, que permiten implementar aplicaciones web, 2017.

MUESTRA

Se tomó una muestra por conveniencia de los frameworks más populares para los lenguajes de programación orientado a objetos en estudio; Java, Php, JavaScript y C#, 2017.

3.4. VARIABLES E INDICADORES

3.4.1. DEFINICIÓN CONCEPTUAL DE LAS VARIABLES

VARIABLE DE INTERÉS

Framework. - Un framework, o marco de trabajo, es un conjunto de bibliotecas de código con una estructura estándar y definida que permiten la implementación ágil de aplicaciones mediante funcionalidades ya creadas que resuelven problemas genéricos.

VARIABLES DESCRIPTIVAS

Biblioteca de modelo. - Es un conjunto de herramientas preparadas para ser utilizadas por los objetos que representan los datos de una aplicación específica. Actúa como

intermediario en la interacción con la base de datos proveyendo de métodos funcionales como mapeo de datos, consultas y sentencias.

Biblioteca de vista. - Es un conjunto de herramientas preparadas para ser utilizadas por la parte visual de la aplicación, permiten transportar la información procesada o requerida hacia la vista.

Biblioteca de controlador. - Es un conjunto de herramientas preparadas para ser utilizadas al implementar la lógica del negocio, y reutilizar lo métodos de la lógica de la aplicación. Estas bibliotecas son intermediarias entre las solicitudes de la vista y la interacción con el modelo.

3.4.2. DEFINICIÓN OPERACIONAL DE LAS VARIABLES

VARIABLE DE INTERÉS

X: Framework.

VARIABLES DESCRIPTIVAS

X1: Biblioteca de modelo.

X2: Biblioteca de vista.

X3: Biblioteca de controlador.

La operacionalización de las variables se muestra en el anexo A.

3.5. TÉCNICAS E INSTRUMENTOS PARA RECOLECTAR INFORMACIÓN

3.5.1. TÉCNICAS

Se consideró la técnica “Análisis Documental”, para aplicar a los manuales de frameworks existentes, arquitecturas técnicas, y manuales de los lenguajes de programación Php, Java, JavaScript y C#.

3.5.2. INSTRUMENTO

Se ha diseñado el instrumento “Ficha de análisis documental” para recolectar y documentar información sobre las arquitecturas, patrones de diseño e implementación para elaborar un modelo arquitectónico estándar, capas de abstracción, interfaces,

metodologías y tecnologías que permiten el acceso a la base de datos, interacción con la interfaz gráfica de usuario, y el procesamiento de datos.

3.5.3. HERRAMIENTAS PARA EL TRATAMIENTO DE DATOS E INFORMACIÓN

Las herramientas tecnológicas que se utilizarán son elegidas de acuerdo a interoperabilidad que presenta el framework, tanto en su desarrollo como en su despliegue. Por lo cual seleccionamos las tecnologías según la siguiente tabla.

SOFTWARE	FABRICANTE	SERVICIO
Windows 10	Microsoft Corporation	Sistema Operativo de Microsoft, con licencia producida por Microsoft Corporación.
Sublime Text 3	Jon Skinner	Editor de texto y código fuente en varios lenguajes de programación.
XAMPP	Apache Friends	Servidor web Apache con interprete para PHP y base de datos MySql.
Java	Oracle	Plataforma de programación con soporte web.
NetBeans 8.2	Apache Software Foundation	Entorno de desarrollo integrado libre, principalmente para Java.
NodeJS	Node.js Developers	Entorno en tiempo de ejecución multiplataforma de código abierto, ejecuta JavaScript del lado del servidor.
VisualStudio 2015	Microsoft Corporation	Entorno de desarrollo integrado con soporte a múltiples lenguajes, para sistemas operativos Windows.
Chrome	Google	Es un navegador web desarrollado por Google de distribución gratuita.
MS Office 2016	Microsoft Corporation	Es una suite ofimática con licencia producida por Microsoft.
MS Visio 2013	Microsoft Corporation	Es un software de dibujo vectorial con licencia producida por Microsoft.

Tabla Nº 3.1. Herramientas tecnológicas para tratamiento de datos.

3.5.4. TÉCNICAS PARA APLICAR SCRUM

Basado en el marco teórico desarrollado en el Capítulo II, formulamos las fases para desarrollar el framework, usando Scrum donde dictamina la utilización de tres artefactos esenciales en un proyecto: la pila del proyecto (product backlog), con los requerimientos totales del aplicativo a desarrollar; la pila de iteración (sprint backlog), que determina los requerimientos a desarrollar en cada iteración; y la gráfica de trabajo pendiente (burndown charts), que representa el trabajo que falta por hacer en un proyecto en el tiempo, como se muestra en las siguientes tablas.

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
Asignar roles scrum	Roles Scrum	Identificar los responsables de los roles centrales y no centrales.	Propietario del producto
Recopilar los requerimientos de dominio	Requerimientos de dominio	Recopilar los requerimientos del cliente.	Propietario del producto y los socios
Elaboración de las historias de usuarios	Historias de usuario	Agrupar en historias de usuarios los requerimientos del cliente.	Equipo
Elaboración de la pila del proyecto	Pila del proyecto	Revisión de las historias de usuario y elaboración de la pila del proyecto.	Equipo
Definir la cantidad de sprint	Sprints del proyecto	Clasificar las historias de usuario de la pila del proyecto.	Equipo
Asignar tareas a los sprints	Tareas de los sprints	Evaluar y asignar tareas a los sprint, estimando las horas necesarias para su ejecución.	Equipo
Estimar tiempo de esfuerzo	Gráfico de trabajo pendiente	Calcular la capacidad diaria de desarrollo del equipo y el total de horas de los sprints.	Facilitador

Tabla Nº 3.2. Entregables Scrum.

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLES
Ejecución de cada sprint del proyecto	Ejecución de los sprint	Desarrollo de cada tarea de los sprint.	Equipo
Reunión diaria de planificación y análisis del avance del sprint.	Reunión diaria	Reunión del equipo para planificar el trabajo por hacer y analizar el trabajo realizado y problemas encontrados.	Facilitador y Equipo
Realizar el gráfico de trabajo pendiente del sprint	Grafica de trabajo pendiente	Revisar la iteración y consolidar los esfuerzos realizados para elaborar el gráfico de trabajo pendiente.	Facilitador
Realizar la reunión de revisión y aprobación del sprint.	Revisión del sprint	Revisión de cumplimiento de cada tarea del sprint para su aprobación o rechazo.	Propietario del producto y socios
Reunión de análisis de trabajo realizado al finalizar los sprint	Retrospectiva general	Reunión de debate de dificultades y aciertos en la ejecución del sprint.	Facilitador y Equipo
Reunión de análisis de trabajo realizado y pendiente con el objetivo de mejorar los sprint	Refinamiento general	Reunión de profundización en el entendimiento de los requerimientos del socio.	Propietario del producto y equipo

Tabla Nº 3.3. Entregables de Scrum en el ciclo de desarrollo.

CAPÍTULO IV
RESULTADOS DE LA INVESTIGACIÓN

4.1 ENTREGABLES SCRUM

Según el proceso de desarrollo del marco de trabajo Scrum, se define los roles, artefactos y actividades Scrum. A continuación, mostramos los resultados del proceso Scrum, los artefactos y la notación UML utilizados.

4.1.1 ROLES SCRUM

Se realizó una primera reunión entre el grupo de trabajo, se eligió al Scrum Master basado en los conocimientos o certificaciones Scrum; se eligió al propietario del producto de acuerdo al conocimiento y la experiencia relacionada al tema de desarrollo, en este caso el Framework, y el resto forma parte del equipo de desarrollo.

ROLES CENTRALES

PROPIETARIO DEL PRODUCTO

Responsable: Bach. Andree Ochoa Cabrera.

El responsable, representa al propietario del producto, como representante con el rol de arquitecto de software. Representa a todos los interesados con el proyecto, verificando continuamente el avance del mismo, evaluando los diferentes logros alcanzados al final de cada iteración y tomando decisiones respecto a las futuras iteraciones, de tal modo que el framework, sea acorde a las expectativas de los socios (Stakeholders).

FACILITADOR

Responsable: Bach. Andree Ochoa Cabrera.

El responsable está a cargo de la correcta dirección de los avances del proyecto. Durante cada iteración se comprometió a verificar la disponibilidad de todos los elementos que requiera el equipo para su correcto desarrollo del proyecto. Además, actúa como intermediario para facilitar una correcta comunicación entre el equipo de trabajo y el propietario del producto.

EQUIPO

Responsable: Bach. Andree Ochoa Cabrera.

Encargado del desarrollo del framework en cada una de las iteraciones. Comprometido con el correcto funcionamiento del producto entregado al final de cada iteración. En concordancia con las buenas prácticas de Scrum, realiza el modelado arquitectónico, análisis, diseño, implementación, evaluación y puesta en marcha del proyecto.

ROLES NO CENTRALES

SOCIOS

Los socios son los usuarios y clientes, en adelante llamaremos usuarios a los que optaron por utilizar el framework como una alternativa para la implementación de aplicaciones web.

4.1.2 REQUERIMIENTOS DE DOMINIO

El propietario del producto se reunió con los socios (stakeholders), para nuestro caso el usuario del framework. El usuario lista los requerimientos que debe cumplir el framework, como se muestra en la siguiente tabla.

N°	Requerimientos
01	El framework debe enrutar los request del browser según sea el caso, teniendo en cuenta que desde la llamada se mencione el nombre del controlador, seguido del nombre de la acción. Ejemplo: controlador/acción.
02	Si desde la llamada se requiere enviar datos como parámetros a la acción del controlador, se deben mencionar después de la acción, por ejemplo: controlador/acción/param1/param2/../paramN
03	El framework debe tener carpetas definidas, donde se puedan organizar las clases de controlador, modelo, y vistas. Además de carpetas adicionales que se requieran, como recursos estáticos, ficheros de configuración, librerías externas, etc.

04	El framework debe estar estructurado del tal forma que al implementarlo en un lenguaje pueda tener una estructura y arquitectura estándar, para así poder obtener un framework independiente al lenguaje.
05	Desde las clases heredadas del controlador, se debe poder enviar datos organizados en arrays con índices numéricos enteros, cadenas alfanuméricas.
06	Los controladores son capaces de seleccionar la vista que se mostrará, por defecto la vista seleccionada para una acción, es la vista con el nombre de la acción contenida en la carpeta de la vista con el nombre del controlador. Si se requiere, esta puede ser cambiada seleccionándolo desde el controlador con el nombre del controlador seguido de la acción. "controller/acción"
07	De igual forma, se puede seleccionar los templates, el template por defecto se llama default. Si se quiere cambiar, se llama por el nombre del template desde el controlador.
08	En algunos casos es necesario redirigir una acción a otra acción, ya sea del mismo controlador o no. El controlador debe tener la funcionalidad de poder redirigir el curso de la acción a otra acción, llamando el nombre del controlador seguido del nombre de la acción.
09	Algunas veces es necesario imprimir texto simple, o etiquetada para usos como Ajax, o si la acción no necesita mostrar una vista. El controlador debe poder permitir imprimir texto plano o etiquetado si necesidad de ir hasta la vista.
10	El controlador debe permitir recuperar los datos enviados en la petición, ya sean parámetros GET o POST.

11	El framework debe permitir el uso de Sesiones para diferentes usos. Desde el controlador se debe permitir iniciar una sesión, recuperarla si existe, modificarla o eliminarla si se desea.
12	Desde la vista se debe poder recuperar los datos enviados del controlador, utilizando los mismos índices con los que fueron enviados, ya seas números enteros o cadenas alfanuméricas.
13	En la vista se debe poder imprimir datos reutilizables, como el nombre de la acción o controlador seleccionado.
14	En la vista se debe poder imprimir código html, javascript o css de forma reutilizable, también llamados partials.
15	Las clases heredadas del modelo deben heredar la conexión a la base de datos.
16	El modelo debe tener funciones para poder realizar consultas a la base de datos.
17	El modelo debe tener funciones para realizar sentencias a la base de datos, como Insert, Updates, Delete.
18	Se debe contar con un manual para los usuarios que deseen desarrollar aplicaciones web con el framework.
19	El controlador debe poder importar la clase del modelo e instanciar desde el controlador para utilizar los métodos y atributos de la clase heredada del modelo.
20	El controlador debe tener filtros de acceso al controlador, que permitan ejecutar acciones antes y después de ejecutar la acción requerida por el request.
21	El framework debe traer por defecto una plantilla libre y gratuita, además debe ser ejecutable a modo de ejemplo para poder iniciar rápidamente en el uso del framework.

Tabla 4.1. Requerimientos del framework.

4.1.3 HISTORIAS DE USUARIO

Con participación del facilitador (scrum master) se organizó los requisitos de la tabla anterior, se agruparon de ser necesarios, y se utilizó una herramienta externa a la metodología scrum, que son las historias de usuario.

HU01: Enrutar peticiones (request)

Como : Usuario del framework.

Quiero : Enrutar las peticiones (request) del browser.

Para : Poder ejecutar la acción de controlador solicitado.

Condiciones :

- Las peticiones deben responder a la llamada de una acción dentro de un controlador de la forma: **?k=controlador/accion**.
- Si las peticiones requieren parámetros en la url, la forma de la llamada es **?k=controlador/accion/param1/param2**

Figura 4.1. Historia de usuario: Enrutar peticiones

HU02: Diseñar una arquitectura

Como : Usuario del framework.

Quiero : Estructurar una arquitectura MVC.

Para : Que desde la vista se pueda mostrar lo datos contenidos en los objetos.

Condiciones :

- Poder desarrollar la aplicación de forma estándar en diferentes lenguajes de programación.

Figura 4.2. Historia de usuario: Diseñar una arquitectura

HU03: Enviar datos a la vista

Como : Usuario del framework.

Quiero : Enviar datos desde el controlador hacia la vista

Para : Que desde la vista se pueda mostrar lo datos contenidos en los objetos.

Condiciones :

- Se debe poder enviar datos, y array de datos agrupados con índices numéricos y/o índices string.

Figura 4.3. Historia de usuario: Enviar datos a la vista

HU04: Seleccionar plantillas y vistas

Como : Usuario del framework.

Quiero : Seleccionar la platilla y/o vista desde el controlador.

Para : Que se pueda mostrar la vista y platilla seleccionada.

Condiciones :

- Se debe poder seleccionar la plantilla si se desea cambiar, la plantilla por defecto es el de nombre default.
- La vista por defecto es la que tiene el nombre de la acción invocada, pero esta puede cambiarse si se desea.

Figura 4.4. Historia de usuario: Seleccionar plantillas y vistas

HU05: Redireccionar peticiones

Como : Usuario del framework.

Quiero : Reenviar una petición a otra dirección

Para : Continuar el flujo de la aplicación en un curso alterno.

Condiciones :

- Se debe redirigir una petición, si se requiere, a una acción dentro de un controlador, llamado de la forma ("controlador/accion")

Figura 4.5. Historia de usuario: Redireccionar peticiones

HU06: Imprimir desde el controlador

Como : Usuario del framework.

Quiero : Imprimir texto desde el controlador.

Para : Mostrarlo directamente, sin pasar por la vista..

Condiciones :

- Para casos en que se necesite una vista muy simple, ya sea textual o json, o para usos de Ajax, se debe permitir escribir desde el controlador texto o código html, css o javascript, sin tener que pasar por la vista.

Figura 4.6. Historia de usuario: Imprimir desde el controlador

HU07: Obtener parámetros

Como : Usuario del framework.

Quiero : Obtener los parámetros POST y GET de la petición

Para : Procesar dichos parámetros desde el controlador

Condiciones :

- Se debe obtener los parámetros enviados en el request, tanto como Get o Post, y devolver el valor recuperado en el controlador para ser procesado según la necesidad.

Figura 4.7. Historia de usuario: Obtener parámetros

HU08: Obtener sesiones

Como : Usuario del framework.

Quiero : Iniciar, recuperar y eliminar sesiones

Para : Procesar las sesiones desde el controlador

Condiciones :

- Se debe iniciar sesiones, recuperarlas si existe, y eliminarlas desde el controlador para ser procesado según la necesidad.

Figura 4.8. Historia de usuario: Obtener Sesiones

HU09: Importar el modelo

Como : Usuario del framework.

Quiero : Instanciar el modelo dentro del controlador

Para : Realizar operaciones y obtener datos del modelo.

Condiciones :

- Las clases heredadas del modelo podrán ser instanciadas en el controlador para fines que el desarrollador necesite.

Figura 4.9. Historia de usuario: Importar el modelo

HU10: Recuperar datos en la vista

Como : Usuario del framework.

Quiero : Obtener los datos enviados desde el controlador

Para : Mostrarlos en la vista

Condiciones :

- Así como desde el controlador se enviaron los datos agrupados con índices enteros y string. La vista debe ser capaz de poder recuperar los datos utilizando los índices.
- Además, deben existir funciones que permitan conocer el número de datos agrupados.

Figura 4.10. Historia de usuario: Recuperar datos de la vista

HU11: Imprimir texto y código en la vista (partials)

Como : Usuario del framework.

Quiero : Imprimir en la vista texto plano, y códigos html (partials)

Para : Reutilizar y facilitar códigos y datos en la vista.

Condiciones :

- Desde la vista podemos enviar imprimir datos reutilizables, como son el nombre de la acción o controlador, o crear fragmentos de código html, javascript o css reutilizables.

Figura 4.11. Historia de usuario: Imprimir texto y código desde la vista

HU12: Consultas y sentencias

Como : Usuario del framework.

Quiero : Realizar consultas y sentencias sql.

Para : Guardar y obtener datos de la base de datos en el

Condiciones :

- Las clases heredadas de la clase Model, deben heredar la conexión con la base de datos, y permitir realizar consultas, insertar, eliminar o actualizar la base de datos.

Figura 4.12. Historia de usuario: Consultas y sentencias

HU13: Manual de usuario

Como : Usuario del framework.

Quiero : Contar con una manual de usuario

Para : Poder aprender a utilizar el framework.

Condiciones :

- El framework debe contar con un manual donde se muestren las herramientas que provee el framework, como usarlos y algunos ejemplos.

Figura 4.13. Historia de usuario: Manual de usuario

4.1.4 PILA DEL PROYECTO

Una vez identificados las historias de usuario, ordenamos las historias de acuerdo a la secuencia de ejecución o importancia para el proyecto, además estimamos las historias de usuario utilizando la técnica del Planning Poker, esta técnica utiliza una sucesión de números basados en la serie de Fibonaci, si bien utilizamos números para representarlos, estos no representan una valoración cuantitativa, sino más bien cualitativa, los cuales indican cuales son más importantes que otros. Nuestra primera versión del backlog.

Item	**ID**	**Historia de usuario**	**Valoración (Planning Poker)**
1	HU02	Diseñar una arquitectura	40
2	HU01	Enrutar peticiones (request)	100
3	HU04	Seleccionar plantillas y vistas	20
4	HU05	Redireccionar peticiones	13
5	HU06	Imprimir desde el controlador	5
6	HU07	Obtener parámetros	20
7	HU08	Obtener sesiones	3
8	HU03	Enviar datos a la vista	40
9	HU10	Recuperar datos en la vista	40
10	HU11	Imprimir texto y código en la vista (partials)	13
11	HU09	Importar el modelo	20
12	HU12	Consultas y sentencias	40
13	HU13	Manual de usuario	8

Tabla 4.2. Pila de Proyecto con valoración Planning Póker

La valoración Plannig Póker nos indica que historias son más importantes que otras de forma cualitativa, por esa razón no tiene una unidad definida.

Valoración Planning Poker								
…	3	5	8	13	20	40	100	…
Menos Importante ⟷ Mas importante								

Tabla 4.2.1. Tabla de valoración de Planning Poker

4.1.5 SPRINTS DEL PROYECTO

Para poder definir los sprint, se consideró la valoración del Planning Póker, y la secuencia del ciclo de ejecución planteado anteriormente, agrupados de tal forma que cada sprint pueda generar un sub producto funcional. Los sprint están ordenados de acuerdo a la secuencia de ejecución de una petición (request), y cada sprint debe resolverse de forma secuencial. Algunas historias de usuario no se tomaron en cuenta para desarrollar los sprint, porque tienen poca relevancia y pueden ser resueltos en un sprint posterior.

a. Sprint 1: Estandarización de arquitectura y enrutamiento.

Sprint	Ítem	ID	Historia de usuario	Valoración (Planning Póker)
1	1	HU02	Diseñar una arquitectura	40
	2	HU01	Enrutar peticiones (request)	40
			Valoración total	80

Table 4.3. Sprint 1: Estandarización de arquitectura y enrutamiento.

b. Sprint 2: Biblioteca de controlador.

Sprint	Ítem	ID	Historia de usuario	Valoración (Planning Póker)
2	3	HU04	Seleccionar plantillas y vistas	20
	4	HU05	Redireccionar peticiones	13
	5	HU06	Imprimir desde el controlador	8
	6	HU07	Obtener parámetros	20
	7	HU08	Obtener sesiones	5
			Valoración total	66

Table 4.4. Sprint 2: Biblioteca de controlador.

c. Sprint 3: Biblioteca de vista.

Sprint	Ítem	ID	Historia de usuario	Valoración (Planning poker)
3	8	HU03	Enviar datos a la vista	20
	9	HU10	Recuperar datos en la vista	20

	10	HU11	Imprimir texto y código en la vista	13
			Valoración total	53

Table 4.5. Sprint 3: Biblioteca de vista.

d. Sprint 4: Biblioteca de modelo.

Sprint	Ítem	ID	Historia de usuario	Valoración (Planning Póker)
4	11	HU09	Importar el modelo	20
	12	HU12	Consultas y sentencias	20
			Valoración total	40

Table 4.6. Sprint 4: Biblioteca de modelo.

4.1.6 TAREAS DE LOS SPRINTS

Asignamos tareas para las historias de usuario en cada sprint, y estimamos el tiempo de desarrollo de cada tarea, el tiempo de desarrollo para una tarea no debe ser mayor a las 8 horas, en el caso de que la tarea sea mayor a las 8 horas, esta se debe dividir en sub tareas.

a. Sprint 1: Arquitectura estándar y enrutamiento.

Sprint	ID	Historia de usuario	Estimación (horas)
1	**HU02**	**Diseñar una arquitectura**	
	S1T01	Analizar el patrón arquitectónico	8
	S1T02	Estructurar los ficheros	6
	S1T03	Identificar las clases a utilizar	6
	S1T04	Analizar palabras reservadas de los lenguajes de programación.	4
	S1T05	Identificar los nombres de los métodos a utilizar	6
	HU01	**Enrutar peticiones (request)**	
	S1T06	Definir el formato de peticiones desde la url	6
	S1T07	Diseñar el diagrama del enrutamiento	8
	S1T09	Desarrollar el enrutamiento	8
		Estimación total	52

Table 4.7. Sprint 1: Tareas asignadas y estimación en horas.

b. Sprint 2: Biblioteca de controlador.

Sprint	ID	Historia de usuario	Estimación (horas)
2	**HU04**	**Seleccionar plantillas y vistas**	
	S2T01	Definir el formato de selección del template	2
	S2T02	Definir el formato de selección de la vista	2
	S2T03	Desarrollar los métodos dentro del controlador	8
	HU05	**Redireccionar peticiones**	
	S2T04	Definir el formato de la redirección	2
	S2T05	Desarrollar el método dentro del controlador	6
	HU06	**Imprimir desde el controlador**	
	S2T06	Definir el formato del envío del texto	2
	S2T07	Desarrollar el método dentro del controlador	6
	HU07	**Obtener parámetros**	
	S2T08	Definir el formato de la obtención del parámetro	2
	S2T09	Desarrollar el método dentro del controlador	6
	HU08	**Obtener sesiones**	
	S2T10	Definir el formato de iniciar sesiones	2
	S2T11	Definir el formato de recuperar y destruir sesiones	2
	S2T12	Desarrollar el método dentro del controlador	6
		Estimación total	46

Table 4.8. Sprint 2: Tareas asignadas y estimación en horas.

c. Sprint 3: Biblioteca de vista

Sprint	ID	Historia de usuario	Estimación (horas)
3	**HU03**	**Enviar datos a la vista**	
	S3T01	Definir el formato de envío de datos	4
	S3T02	Desarrollar el método dentro del controlador	8

	HU10	**Recuperar datos en la vista**	
	S3T03	Definir el formato de recuperación de datos	4
	S3T04	Desarrollar el método dentro de la vista	8
	HU11	**Imprimir texto y código en la vista**	
	S3T05	Definir el formato de selección de partials	4
	S3T06	Desarrollar el método dentro de la vista	8
		Estimación total	36

Table 4.9. Sprint 3: Tareas asignadas y estimación en horas.

d. Sprint 4: Biblioteca de modelo

Sprint	**ID**	**Historia de usuario**	**Estimación (horas)**
4	**HU09**	**Importar el modelo**	
	S4T01	Definir el formato de importación del modelo	2
	S4T02	Desarrollar el método dentro del controlador	4
	HU12	**Consultas y sentencias**	
	S4T03	Definir el formato de consultas simples de tabla.	2
	S4T04	Desarrollar la función en el modelo	2
	S4T05	Definir el formato de instanciación de la tabla consultada	6
	S4T06	Definir el formato de retorno de valores de la tabla instanciada	8
	S4T07	Desarrollar los métodos en el modelo	6
		Estimación total	30

Table 4.10. Sprint 4: Tareas asignadas y estimación en horas.

4.1.7 GRAFICO DE TRABAJO PENDIENTE

De las tablas anteriores, calculamos los días totales y las horas totales para todo el desarrollo.

Horas de los Sprint	
Sprint 1	52 horas
Sprint 2	46 horas
Sprint 3	36 horas
Sprint 4	30 horas
Total	**164 horas**

Tabla 4.11. Calculo del total de horas de los sprint.

Para poder estimar las horas requeridas para cada sprint, primero se estimó el tiempo disponible de los recursos humanos. Para nuestro caso, el equipo de desarrollo está integrado por una sola persona. En scrum se debe definir algunas políticas de trabajo, para nuestra investigación se consideró que la duración mínima de un sprint es de dos semanas, considerando que solo se trabaja 8 horas diarias y solo 5 días a la semana.

Desarrollador (Andree Ochoa Cabrera)	**(horas)**
Actividades scrum	10
Permisos	6
Otros compromisos	14
Total	30
Sprint de 2 semanas (80 horas-total)	50
Capacidad (80%)	**40**
Capacidad diaria (capacidad/días del sprint**)**	40/10 = **4**

Tabla 4.12. Estimación de tiempo de desarrollo del equipo.

En la tabla anterior se calculó que la capacidad de desarrollo del equipo es de 4 horas diarias.

Tiempo de desarrollo en días	
Total, horas	164 horas
Capacidad diaria	4 horas
Total, de días de desarrollo	164 / 4 = **41 días**

Tabla 4.13. Calculo del total de días para el desarrollo del framework.

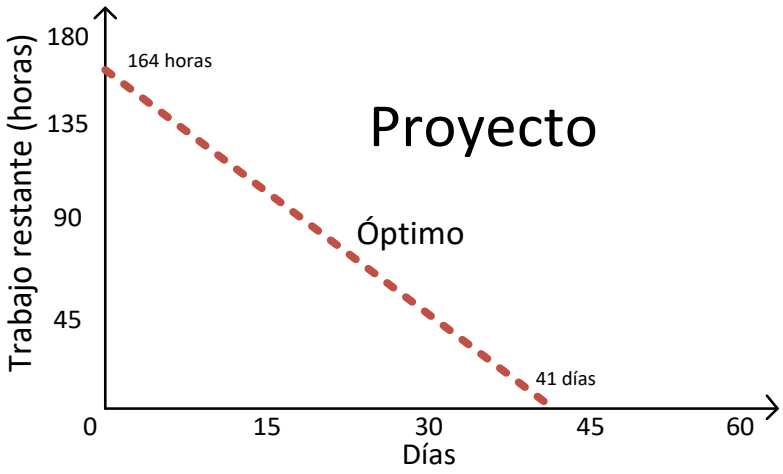


Figura 4.14 Grafico de trabajo pendiente óptimo del proyecto

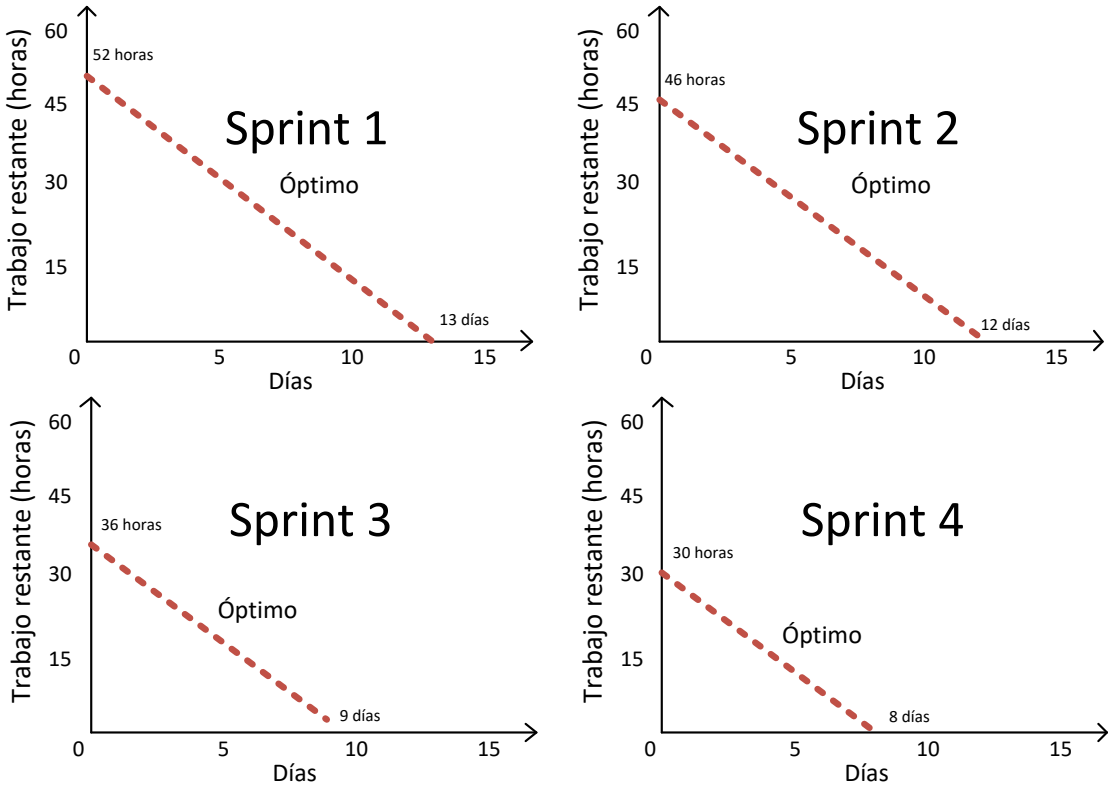


Figura 4.15 Grafico de trabajo pendiente óptimo de cada sprint

4.2 ENTREGABLES DE SCRUM EN EL CICLO DE DESARROLLO

4.2.1 EJECUCIÓN DE LOS SPRINT

Una vez que se planeó los sprints, se procedió a desarrollar cada uno de ellos.

SPRINT 1: ARQUITECTURA ESTANDAR Y ENRUTAMIENTO

Este sprint es de fundamental importancia, es donde se diseñó la arquitectura inicial del framework, y se desarrolla el enrutamiento.

a. Analizar el patrón arquitectónico

El patrón arquitectónico elegido es el de Modelo, Vista y Controlador, MVC ya que permite modularizar la lógica, las interfaces de usuario y modelado de las entidades, además que es el patrón de diseño más utilizados para aplicaciones web.

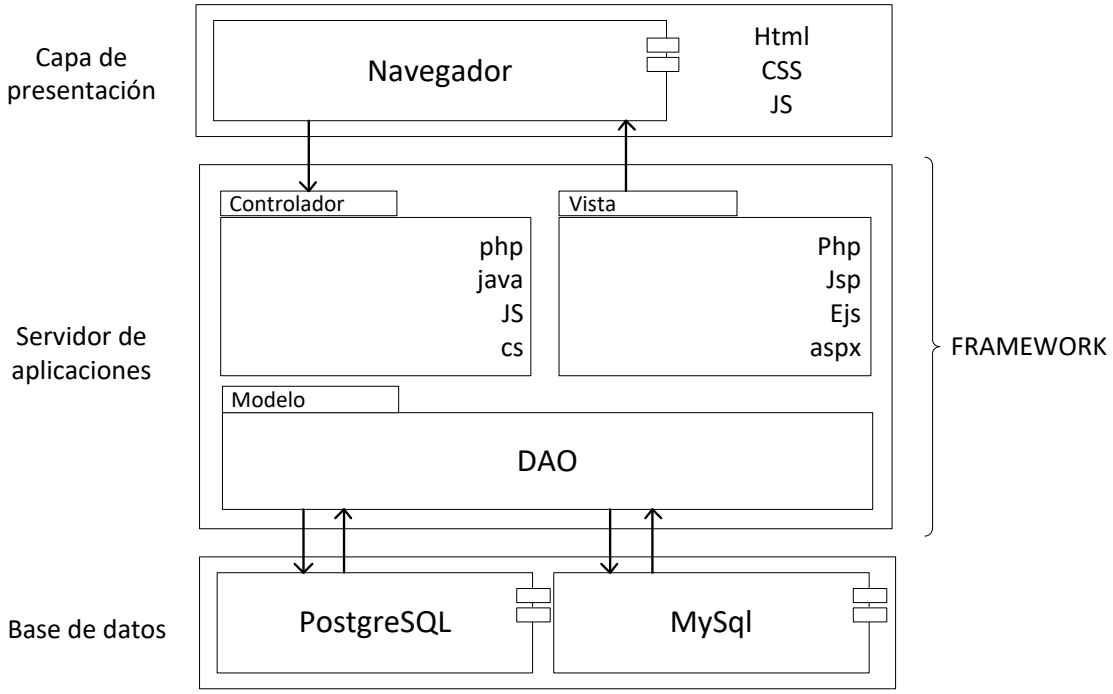


Figura 4.16. Arquitectura técnica inicial del framework.

En la figura anterior, se diseñó la primera arquitectura técnica del framework, donde se distingue la modularización en el servidor de aplicaciones utilizando MVC. En la capa de base de datos se muestra dos bases de datos, la razón es que el framework ha sido probado para estas dos bases de datos, además puede utilizarse con cualquier otra base de datos compatible al lenguaje. Tratándose de un framework estándar, la base de datos debe ser a elección del usuario.

b. Estructurar los ficheros

Se estructuró los ficheros donde se almacenarán las librerías del framework y los archivos a desarrollar por el usuario, consideramos que se debe tener una arquitectura MVC.

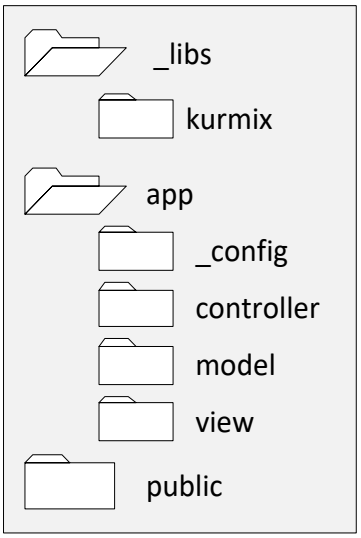


Figura 4.17. Estructura estándar de los ficheros del framework

Para todos los escenarios del framework, se tiene las siguientes carpetas:

kurmix : Contiene todas las librerías del framework.

_config : Contiene los archivos de configuración del framework.

controller : Contiene los archivos controladores creados por el usuario.

model : Contiene los archivos de modelo creados por el usuario.

view: Contiene los archivos de vista creados por el usuario.

public : Contiene los recursos estáticos de la web (imágenes, css, js, etc).

Tanto los lenguajes como los IDEs de desarrollo, tienen estructuras propias, sin embargo, se ha logrado estandarizar estas estructuras, para nuestro caso de estudio tendríamos las siguientes estructuras para cada lenguaje.

El usuario tiene libre elección de elegir un IDE o utilizar un editor de código, para nuestra investigación se utilizó para PHP en editor de texto SublimeText.

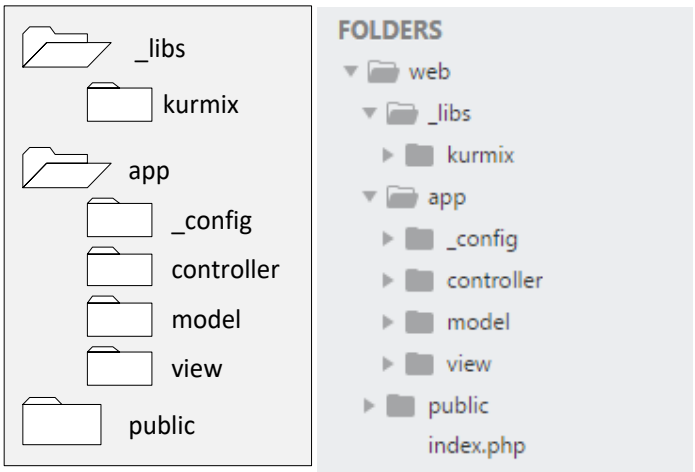


Figura 4.18. Estructura de ficheros en PHP con SublimeText.

En el caso de Java, se utilizó el IDE NetBeans y Eclipse por ser los más usados por los desarrolladores Java. En este caso se tiene algunos problemas en la estandarización de la estructura de ficheros del framework, cada IDE tiene predefinida una estructura al cual tuvimos que adaptarnos y buscar una estructura similar.

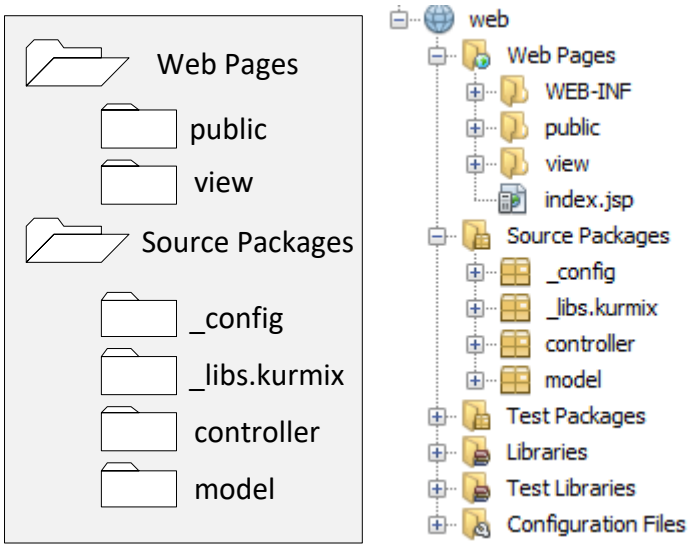


Figura 4.19. Estructura de ficheros en Java con NetBeans 8.2.

Para utilizar Java con el IDE Eclipse, la estructura es muy similar. La desventaja de usar estos IDEs, es que obligan a usar paquetes y no carpetas.

En el caso de JavaScript se utilizó el editor de texto SublimeText. Es necesario mencionar que JavaScript aparte de ser un lenguaje interpretado del lado del cliente, también es capaz de correr del lado del servidor, y esto es gracias al servidor NodeJS.

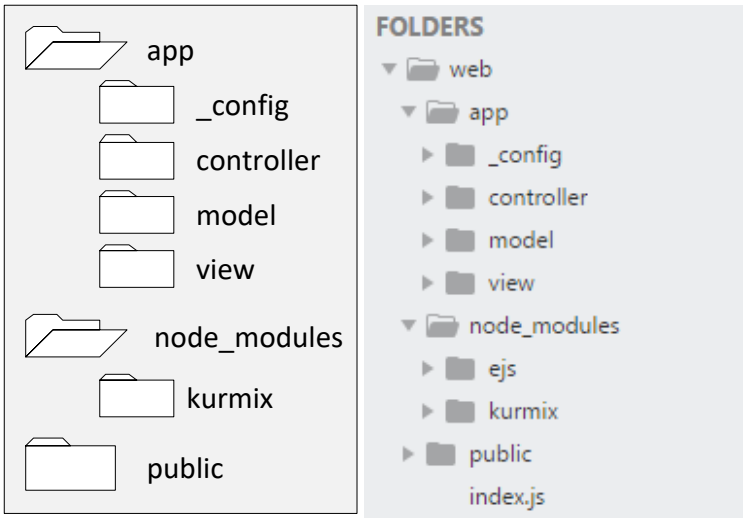


Figura 4.20. Estructura de ficheros en JavaScript-NodeJS con SublimeText.

El lenguaje de programación CSharp es un lenguaje propietario de Microsoft, es por ello que utilizamos su propio IDE que es el Visual Studio, para esta implementación, se utilizó la versión 2015. Al igual que PHP y JavaScript, no se tuvo problemas en la estructuración, ya que su entorno de desarrollo permite crear carpetas del lado del servidor.

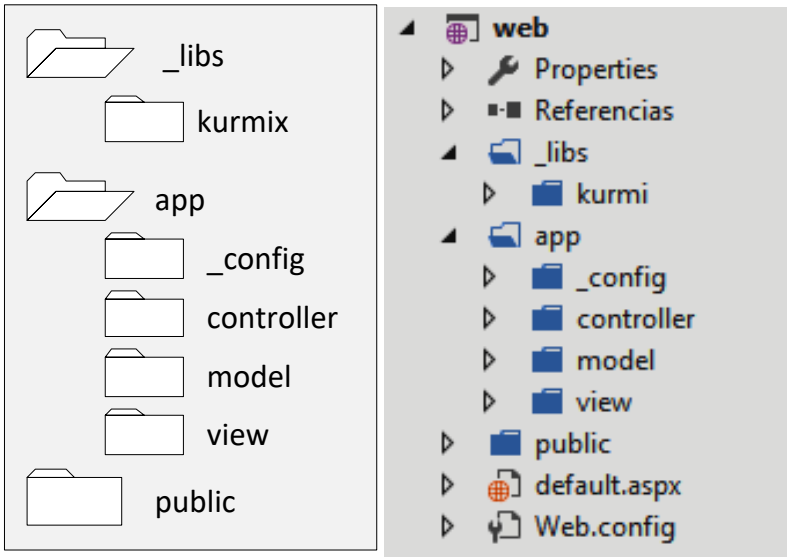


Figura 4.21. Estructura de ficheros en C# con Visual Studio 2015.

c. Identificar las clases a utilizar

Basándonos en la arquitectura MVC identificamos el flujo de una llamada y su respuesta.

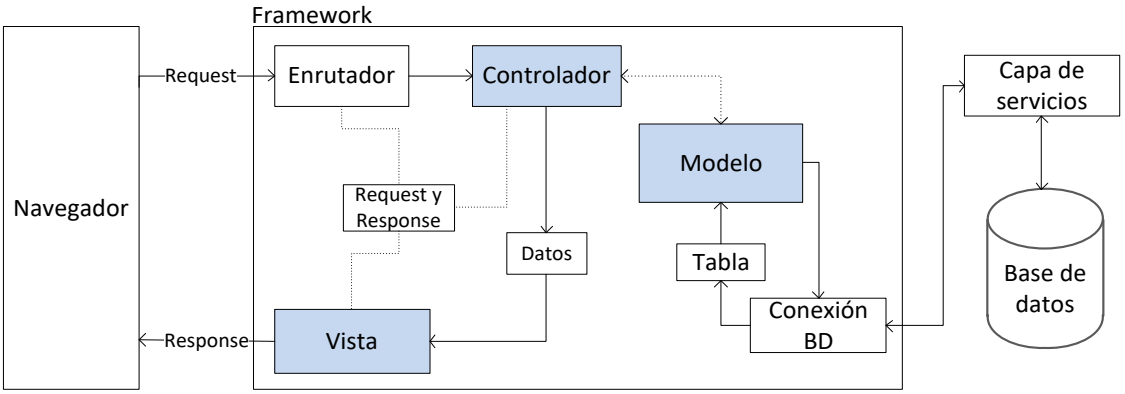


Figura 4.22. Arquitectura MVC con identificación de clases del framework

Claramente distinguimos 8 clases fundamentales para el funcionamiento del framework, además podemos observar que existe cierta relación entre ellas. Nombramos las clases con nombres en inglés para una mejor estandarización.

Router (enrutador): Es una clase estática cuya función es direccionar al controlador requerido.

ReqRes (request y response): Esta clase auxiliar almacena el evento y objeto request y response de la petición.

Views (vista): Esta clase contiene las funciones necesarias para que la aplicación pueda mostrar una interfaz gráfica de usuario.

Controller (controlador): Esta clase contiene las funciones necesarias para procesar datos y enviar datos a la vista.

Data (datos): Es una clase auxiliar encapsula los datos a enviar del controlador a la vista.

Model (modelo): Esta clase contiene funciones para ayudar en el modelado de entidades en una aplicación.

Connection (conexión): Esta clase estática contiene la conexión a la base de datos.

Table (tabla): Esta clase ayuda a manipular los datos extraídas de una base de datos.

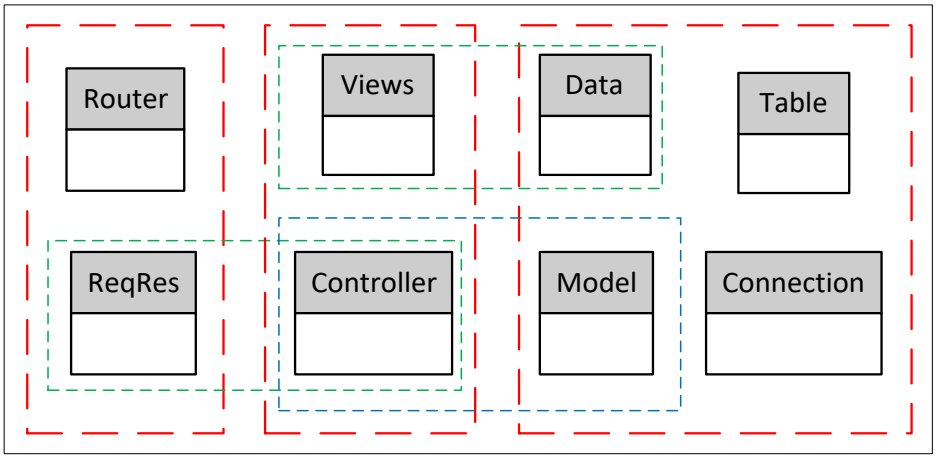


Figura 4.23. Clases identificadas para el framework

La razón de nombrar la Vista como Views y no solo View, es porque al implementar el framework en C#, nos encontramos con una clase latente con el nombre View.

d. Analizar palabras reservadas de los lenguajes de programación.

Las palabras reservadas son identificadores propios de un determinado lenguaje, y no se pueden utilizar para nombrar nuevos identificadores. Identificamos las palabras reservadas, para así poder estandarizar nombres comunes en los lenguajes de estudio.

Palabras reservadas PHP			
abstract	and	array	as
class	default	elseif	endif
break	callable	case	catch
clone	const	continue	declare
die	do	echo	else
empty	enddeclare	endfor	endforeach
endswitch	endwhile	eval	exit
final	finally	for	foreach
global	goto	if	implements
include_once	instanceof	insteadof	interface
list	namespace	new	or
private	protected	public	require
return	static	switch	throw
try	unset	use	var
extends	function	include	isset
print	require_once	trait	while
xor	yield		

Tabla 4.14. Palabras reservadas para el lenguaje PHP

La tabla anterior muestra las palabras reservadas usualmente utilizadas de PHP, sin embargo, php tiene una mayor cantidad de palabras reservadas debido a que existen funciones que no están ligadas a una clase u objeto, y estas están latentes en cada hoja de código. Por otra parte, Java es un lenguaje que muy poca restricción tiene debido a que es completamente orientado a objetos.

Palabras reservadas JAVA			
abstract	continue	for	new
assert	default	goto	package
boolean	do	if	private
break	double	implements	protected
byte	else	import	public
case	enum	instanceof	return
catch	extends	int	short
char	final	interface	static
class	finally	long	strictfp
const	float	native	super
switch	synchronized	this	Throw
throws	transient	try	void
volatile	while		

Tabla 4.15. Palabras reservadas para el lenguaje JAVA

JavaScript es un caso especial debido a su constante evolución tanto en sintaxis como en incremento de funcionalidades.

Palabras reservadas JavaScript			
break	case	catch	Class
const	continue	debugger	default
delete	do	else	export
extends	finally	for	function
if	import	in	instanceof
new	of	return	super
switch	this	throw	try
typeof	var	void	while
whit	yield	await	enum
implements	interface	let	package
private	protected	public	static
abstract	boolean	byte	char
double	final	float	goto
int	long	native	short
synchronized	throws	transient	volatile
null	true	false	

Tabla 4.16. Palabras reservadas para el lenguaje JavaScript

C# también tiene clases latentes en sus hojas aspx, tal es el caso de la clase View.

Palabras reservadas C#			
abstract	enum	long	stackalloc
as	event	namespace	static
base	explicit	new	string
bool	extern	null	struct
break	false	object	switch
byte	finally	operator	this
case	Fixed	out	throw
catch	float	override	true
char	for	params	try
checked	foreach	private	typeof
class	goto	protected	uint
const	if	public	ulong
continue	implicit	readonly	unchecked
decimal	in	ref	unsafe
default	int	return	ushort
delegate	interface	sbyte	using
do	internal	sealed	virtual
double	is	short	void
else	lock	sizeof	while

Tabla 4.17. Palabras reservadas para el lenguaje C#

e. Identificar los nombres los métodos a utilizar

Determinamos algunas funciones básicas a implementar en el framework, teniendo en cuenta no utilizar las palabras reservadas identificadas en la tarea anterior.

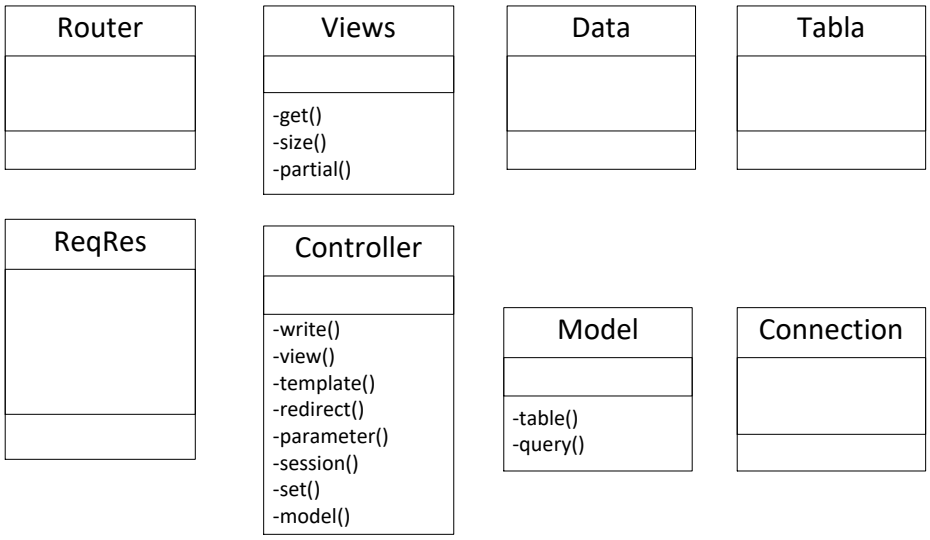


Figura 4.24. Funciones básicas identificadas para el framework.

f. Definir el formato de peticiones desde la url

Las peticiones que se hacen desde el navegador deben tener un estándar para que el enrutador pueda dirigir al controlador requerido. El formato que definimos es el siguiente:

http://localhost:3000/web/?k=cotrolador/accion

Donde:

- **http://localhost:3000/** Dominio y puerto
- **web/** paquete del proyecto con el framework.
- **?k=** Muletilla para el llamado del controlador
- **controlador/** Controlador
- **accion** Acción del controlador

Además, definimos como regla que, si no se especifica el nombre de la acción, este por defecto será **index.**

http://localhost:3000/web/?k=controlador/

< >

http://localhost:3000/web/?k= controlador/**index/**

Y si no se define el nombre del controlador, este por defecto será **index** como controlador y llamará a la acción **index** del controlador **index.**

http://localhost:3000/web/ < > http://localhost:3000/web/**?k=index/index/**

Para algunos casos necesitaremos enviar variables desde la url, en esos casos, los siguientes parámetros denotarán las variables enviadas.

http://localhost:3000/web/?k=controlador/accion/**param1/param2/.../paramN**/

Para los lenguajes altamente tipados, como es el caso de Java o C#, se entenderán que los parámetros enviados son de tipo string. En el caso de requerir parámetros numéricos, se debe realizar una conversión de tipo desde el lado del servidor.

g. Diseñar el diagrama del enrutamiento

En la tarea anterior definimos el formato de las peticiones desde la url, siguiendo dicho formato, diseñamos como seria el comportamiento de un enrutamiento estático. A continuación, diseñamos el flujo del enrutamiento para una petición (request) desde una url.

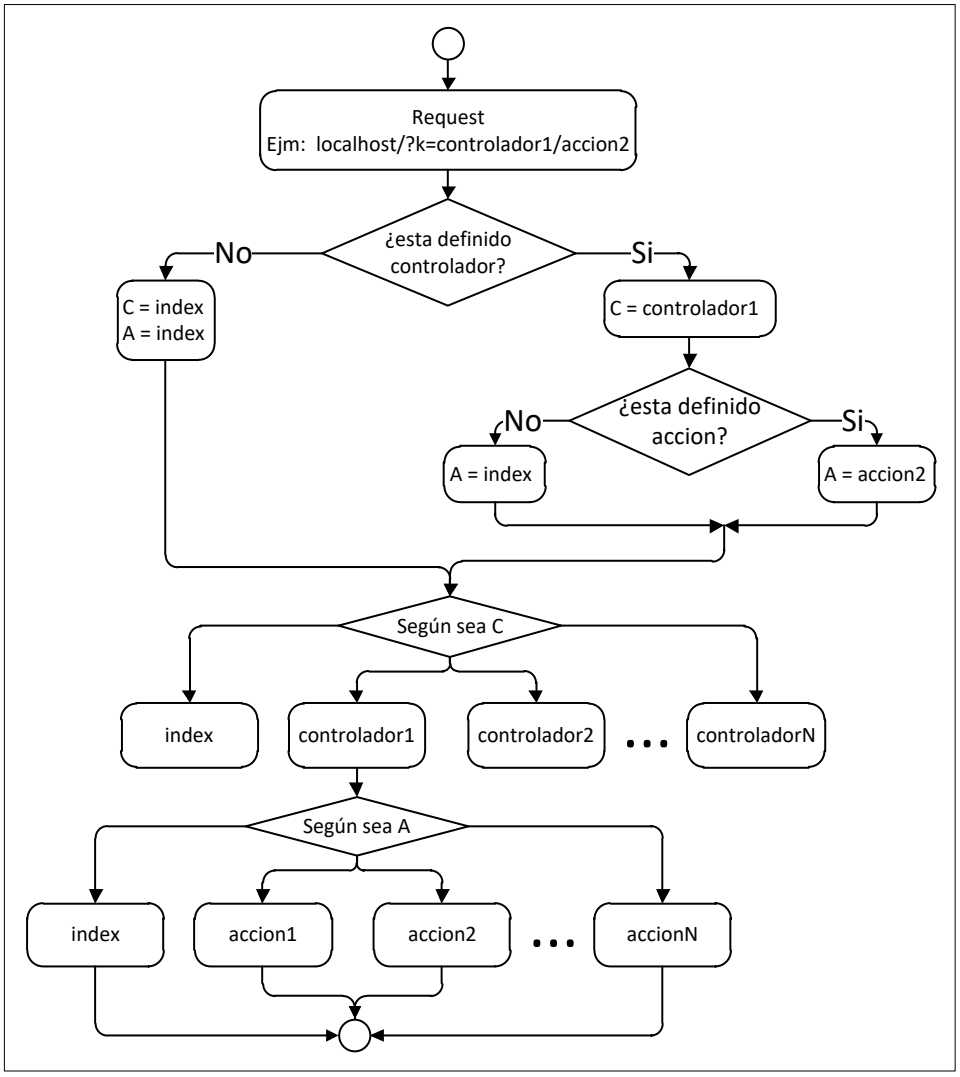


Figura 4.25. Diagrama de flujo del enrutamiento

Utilizamos un diagrama de flujo para representar el enrutamiento, debido a que el enrutamiento es bastante algorítmico.

h. Desarrollar el enrutamiento

Desarrollamos la clase Router, quien está encargado de enrutar las peticiones del request, mostramos a continuación la lógica esencial en código para cada lenguaje de programación en estudio.

PHP

```php
$controller = 'index';
$accion = 'index';

$a = explode('/',$_GET['k']);

if(sizeof($a)>0) $controller = $a[0];
if(sizeof($a)>1) $accion = $a[1];

require ('app/controller/'.$controller.'_controller.php');
$controlador_class = $controller.'_controller';
$controler = new $controlador_class();

call_user_func_array(array($controler, $accion), $params);
```

Figura 4.26. Código de enrutamiento PHP

JAVA

```java
String controller = "index";
String action = "index";

String[] a = k.split("/");

if(a.length>0) if(!a[0].trim().equals("")) controller = a[0];
if(a.length>1) action = a[1];

try{
   Class<?> c = Class.forName("controller."+controller+"_controller");
   Object o = c.newInstance();
   Method method;

        method = o.getClass().getMethod(action,types);
        method.invoke(o,params);

}catch (Exception e) {   }
```

Figura 4.27. Código de enrutamiento Java

JavaScript

```
var controller = 'index';
var action = 'index';

var a = query.k;
a = a.split("/");

if(a.length>0) controller = a[0];
if(a.length>1) if(a[1].length>0) action = a[1];

var Controller_class = require('../../app/controller/'+controller+'_controller');
var controller_obj = new Controller_class();

eval('controller_obj.'+action+parms);
```

Figura 4.28. Código de enrutamiento JavaScript

C#

```
string controller = "index";
string action = "index";

string url = "" + resquest["k"];
string[] a = url.Split('/');

if (a.Length > 0) if (!a[0].Trim().Equals("")) controller = a[0];
if (a.Length > 1) action = a[1];

var x = new Object();

try{
   x = Activator.CreateInstance(Type.GetType("web.app.controller." + controller + "_controller"));

   x.GetType().GetMethod(action).Invoke(x, param);

}catch (Exception e) { }
```

Figura 4.29. Código de enrutamiento C#

SPRINT 2: BIBLIOTECA DE CONTROLADOR

En este sprint se desarrolló las librerías del Controlador del patrón MVC.

a. Definir el formato de selección del template

Definimos una función con el cual podamos cambiar las plantillas de la vista. Si bien esta función hace referencia a la vista a mostrar, debe manipularse desde el controlador, ya que es el controlador quien direcciona la vista a mostrar. El formato de esta función será de la siguiente manera:

template(“nombre_de_la_plantilla”)

Esta función debe ser heredada de la clase controlador, y por tanto ser llamada con la notación **this**, y se envía como parámetro el nombre de la plantilla, la extensión del archivo de la plantilla no será necesario, pero el framework entenderá las siguientes extensiones para cada lenguaje, tendríamos el siguiente cuadro.

Template		
Lenguaje	**Formato**	**Extensión**
PHP	$this->**template**("*nombre_de_la_plantilla*")	.php
JAVA	this.**template**("*nombre_de_la_plantilla*")	.jsp
JavaScript	this.**template**("*nombre_de_la_plantilla*")	.ejs
C#	this.**template**("*nombre_de_la_plantilla*")	.aspx

Tabla 4.18. Formato de la función template y su extensión para cada lenguaje.

Las plantillas se encuentran en la carpeta _templates de la carpeta view, la plantilla por defecto es **default.**

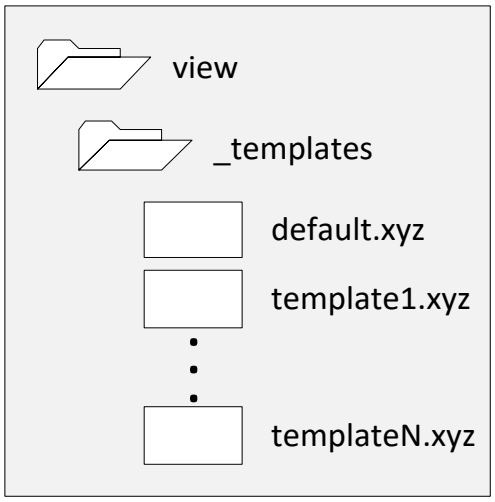


Figura 4.30. Carpeta _templates contenida en la carpeta view.

b. Definir el formato de selección de la vista

Desde el controlador se puede manejar un direccionamiento de la vista a mostrase, se definió el nombre como **view**.

view("nombre_controlador/nombre_accion")

Debemos enviar como parámetro el nombre del controlador y el nombre de la acción.

Controller | View		
Lenguaje	**Formato**	**Extensión**
PHP	$this->**view**("*nombre_controlador/nombre_accion*")	.php
JAVA	this.**view**("*nombre_controlador/nombre_accion*")	.jsp
JavaScript	this.**view**("*nombre_controlador/nombre_accion*")	.ejs
C#	this.**view**("*nombre_controlador/nombre_accion*")	.aspx

Tabla 4.19. Formato de la función view y su extension para cada lenguaje.

Las vistas se encuentran dentro de la carpeta con el nombre del controlador, y está dentro de la carpeta view. Como regla, si no se especifica la vista, el framework entiende que llama a la vista con el nombre de la acción dentro de la carpeta con el nombre del controlador.

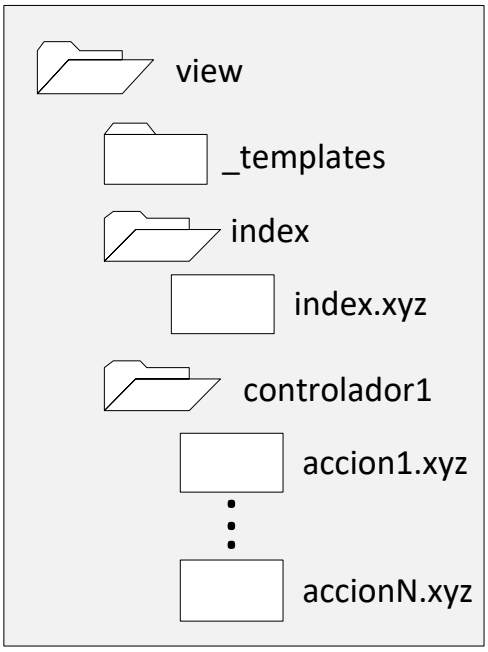


Figura 4.31. Carpeta de vistas contenidas en carpetas de controlador.

c. Desarrollar los métodos dentro del controlador

Mostramos a continuación la lógica esencial en código para cada lenguaje de programación en estudio.

PHP

```
$this->template='default';

if($this->view == null) return;

$url="view/".$this->view;

if($this->template!=null){
        $url = "view/_templates/".$this->template;
}

$v = $this->data;

if($this->template!=null) {
        $this->data->setKurmi($this->view);
}
include('app/'.$url.'.php');
```

Figura 4.32. Código de selección de vista y template PHP

JAVA

```
this.template = "default";

if(this.view == null) return "";

String url="view/"+this.view;

if(this.template!=null){
   url = "view/_templates/"+this.template;
   this.request.setAttribute("kurmi_body", this.view);
}

try{

this.request.getRequestDispatcher(url+".jsp").include(this.req
uest,this.response);
}catch(Exception e){ }
```

Figura 4.33. Código de selección de vista y template Java

JavaScript

```
this.template='default';

if(this.view == null) { this.res.end();return; }

var vis="app/view/"+this.view+'.ejs';
var tmp='';

if(this.template!=null){
   tmp = "app/view/_templates/"+this.template+'.ejs';
}

view.setKurmix(ejs.render({'v':view}));
if(this.template!=null)
            this.res.write(ejs.render({'v':view}));
else
            this.res.write(view.body());
```

Figura 4.34. Código de selección de vista y template JavaScript

C#

```
this.template = "default";

if (this.view == null) return;

string url = "app/view/" + this.view;

if (this.template != null)
{
   url = "app/view/_templates/" + this.template;
   this.context.Items["kurmi_body"] = this.view;
}

try
{
   this.server.Execute(url+".aspx");
}catch(Exception e){  }
```

Figura 4.35. Código de selección de vista y template C#

d. Definir el formato de la redirección

En algunos casos, el usuario del framework necesitará dirigirse a otra acción del mismo controlador o de otro controlador. Definimos la forma de llamar a la función como:

redirect(“nombre_controlador/nombre_accion”)

Debemos enviar como parámetro el nombre del controlador y el nombre de la acción.

Cotroller | Redirect	
Lenguaje	**Formato**
PHP	$this->**redirect**("*nombre_controlador/nombre_accion*")
JAVA	this.**redirect**("*nombre_controlador/nombre_accion*")
JavaScript	this. **redirect**("*nombre_controlador/nombre_accion*")
C#	this.**redirect**("*nombre_controlador/nombre_accion*")

Tabla 4.20. Formato de la función redirect para cada lenguaje

e. Desarrollar el método dentro del controlador

PHP

```
public function redirect($url){
        if(trim($url)!='')     $url='k='.$url;
        header("Location: ?".$url);
}
```

Figura 4.36. Código de redirección en PHP

JAVA

```
public void redirect(String url) {
   if(url.trim().equals("")) url = "index/index";
   try{
      this.request.getRequestDispatcher("/index.jsp?k="+url)
            .include(this.request,this.response);
   }catch(Exception e){}
}
```

Figura 4.37. Código de redirección en Java

JavaScript

```
redirect(url){
        if(url.trim()!='') url = 'k='+url;
        this.res.writeHead(302, { 'Location': '?'+url});
}
```

Figura 4.38. Código de redirección en JavaScript

C#

```
public void redirect(string url){
   if (url.Trim().Equals("")) url = "index/index";
   try{
      this.response.Redirect("?k="+url);
   }
   catch (Exception e){ }
}
```

Figura 4.39. Código de redirección en C#

f. Definir el formato del envío del texto

En algunos casos el usuario del framework necesitara imprimir texto desde el controlador, sin necesidad de crear una vista, por ejemplo, en los casos que se espere una respuesta utilizando AJAX. Definimos la forma de llamar a la función como:

write(“texto a imprimir”)

Enviamos como parámetro el texto a imprimir.

Cotroller | Write	
Lenguaje	**Formato**
PHP	$this->**write**(“texto a imprimir”)
JAVA	this.**write**(“texto a imprimir”)
JavaScript	this.**write**(“texto a imprimir”)
C#	this.**write**(“texto a imprimir”)

Tabla 4.21. Formato de la función write para cada lenguaje.

Puesto que el texto enviado será impreso en el navegador, también podemos enviar texto enriquecido, por ejemplo html.

g. Desarrollar el método dentro del controlador

PHP

```
public  function  write($value){
          echo $value;
}
```

Figura 4.40. Código de impresión desde el controlador PHP

JAVA

```
public   void   write(String value){
   try{
      PrintWriter out = this.response.getWriter();
      out.println(value);
   }catch(Exception e){}
}
```

Figura 4.41. Código de impresión desde el controlador Java.

C#

```
public void write(string value)
{
        this.response.Write(value);
}
```

Figura 4.42. Código de impresión desde el controlador C#.

JavaScript

```
write(value){
        this.res.write(value);
}
```

Figura 4.43. Código de impresión desde el controlador JavaScript.

h. Definir el formato de la obtención del parámetro

En la tarea "f" del sprint 1 definimos el formato de la url, con opción a enviar parámetros, pero para algunos casos necesitaremos enviar parámetros en modo POST el cual no se puede desde la url. Este caso es utilizado frecuentemente en ajax. Para resolverlo, definimos la obtención de parámetros como:

parameter("nombre_del_parametro")

Enviamos el nombre del parámetro con el que fue llamado.

Cotroller | Parameter	
Lenguaje	**Formato**
PHP	$this->**parameter**(“nombre_del_parametro”)
JAVA	this.**parameter**(“nombre_del_parametro”)
JavaScript	this.**parameter**(“nombre_del_parametro”)
C#	this.**parameter**(“nombre_del_parametro”)

Tabla 4.22. Formato de la función write para cada lenguaje.

La función **parameter** debe preferentemente usarse para método POST, pero también funciona para método GET, como regla, si se tiene dos parámetros post y get con el mismo nombre, el framework devolverá primero el de método get.

i. Desarrollar el método dentro del controlador

PHP

```
function parameter($name){
        if(!empty($_GET[$name])) return $_GET[$name];

        if(!empty($_POST[$name])) return $_POST[$name];

        return null;
}
```

Figura 4.44. Código obtener parámetro en Php

JAVA

```
public String parameter(String name){
   return this.request.getParameter(name);
}
```

Figura 4.45. Código obtener parámetro en Java.

JavaScript

```
parameter(name){
        var url = require('url');
        var query = url.parse(this.req.url, true).query;
        if(eval('query.'+name)!==undefined){
                return eval('query.'+name);
        }
}
```

Figura 4.46. Código obtener parámetro en JavaScript.

C#

```
public string parameter(string name)
{
        return this.request[name];
}
```

Figura 4.47. Código obtener parámetro en C#.

j. Definir el formato de iniciar sesiones

Las sesiones web son necesarias para guardar información temporalmente en el servidor del usuario en línea, las sesiones son a menudo muy utilizadas para implementar un login o inicio de sesión de usuarios registrados, para iniciar una sesión utilizamos la siguiente función.

session(“nombre_de_la_sesion”, “valor de la sesión”)

Al llamar a la función **session**, enviamos como parámetros el nombre de la sesión a crear con su respectivo valor; el nombre debe ser una cadena string y el valor puede ser de cualquier tipo o un objeto.

Cotroller | Session	
Lenguaje	**Formato**
PHP	$this->**session**(“nombre_de_la_sesion”, valor_de_la_sesion)
JAVA	this.**session**(“nombre_de_la_sesion”, valor_de_la_sesion)
JavaScript	this.**session**(“nombre_de_la_sesion”, valor_de_la_sesion)
C#	this.**session**(“nombre_de_la_sesion”, valor_de_la_sesion)

Tabla 4.23. Formato de la función session asignando valor para cada lenguaje.

k. Definir el formato de recuperar y destruir sesiones

En la tarea anterior definimos como iniciar una sesión asignando su valor. Ahora definimos la función que recupera una sesión iniciada, y nos devuelve el valor asignado.

session(“nombre_de_la_sesion”)

Al llamar a la función **session**, enviamos como parámetros el nombre de la sesión y este nos devuelve el valor asignado a esa sesión, el valor retornado puede ser de cualquier tipo o un objeto, para lenguajes altamente tipado se debe realizar una conversión de tipo.

Cotroller | Session	
Lenguaje	**Formato**
PHP	$this->**session**(“nombre_de_la_sesion”)
JAVA	this.**session**(“nombre_de_la_sesion”)
JavaScript	this.**session**(“nombre_de_la_sesion”)
C#	this.**session**(“nombre_de_la_sesion”)

Tabla 4.24. Formato de la función session devolviendo el valor asignado.

Tenemos también el caso de que, si queremos eliminar las sesiones creadas, enviamos como parámetro **null**.

Cotroller | Session	
Lenguaje	**Formato**
PHP	$this->**session**(null)
JAVA	this.**session**(null)
JavaScript	this.**session**(null)
C#	this.**session**(null)

Tabla 4.25. Formato de la función session, eliminando las sesiones.

l. Desarrollar el método dentro del controlador

PHP

```
function session($name,$value=null){
        if (session_status() == PHP_SESSION_NONE) {
                session_start();
        }

        if($name===null) {
                session_destroy(); return;
        }

        if($value===null){
                return @$_SESSION[$name];
        }

        $_SESSION[$name] = $value;
}
```

Figura 4.48. Código obtener sesiones en Php

JAVA

```
public <T> T session(String name){
   HttpSession sesion = this.view.getRequest().getSession();

   if(name==null){
          sesion.invalidate(); return null;
   }

   return (T)sesion.getAttribute(name);
}

public void session(String name,Object value){
   HttpSession sesion = this.view.getRequest().getSession();
   sesion.setAttribute(name, value);
}
```

Figura 4.49. Código obtener sesiones en Java

JavaScript

```
session(name, value){
        if(name==null){
                this.req.session = null;
                return;
        }

        if(typeof value !== 'undefined'){
                this.req.session.set(name,value);
                return;
        }

        return this.req.session.get(name);
}
```

Figura 4.50. Código obtener sesiones en JavaScript.

C#

```
public void session(string index, object value){
        this.context.Session[index] = value;
}

public object session(string index){
        if (index == null) {
                this.context.Session.Clear();
                return null;
        }
        return this.context.Session[index];
}
```

Figura 4.51. Código obtener sesiones en C#.

SPRINT 3: BIBLIOTECA DE LA VISTA

a. Definir el formato de envío de datos

Necesitamos enviar datos a la vista desde el controlador. Si bien esta tarea pertenecería al sprint del controlador, la desarrollamos paralelamente con la Vista por la naturaleza de su comportamiento.

Definimos la forma de enviar los datos como:

set(valor_enviado)

Al llamar a la función **set**, debemos enviar como parámetro el valor. Este valor puede ser un texto, numero, array o un objeto.

Cotroller | Set	
Lenguaje	**Formato**
PHP	$this->**set**(valor_enviado)
JAVA	this.**set**(valor_enviado)
JavaScript	this.**set**(valor_enviado)
C#	this.**set**(valor_enviado)

Tabla 4.26. Formato de la función set, envío simple.

Si necesitaríamos enviar datos diversos, por ejemplo, un array de strings y un array de doubles, necesitaríamos enviar de forma separada, y asignar un índice para diferenciarlo. Por tal motivo definimos la función set con índice.

set(indice, valor_enviado)

Debemos enviar como parámetros el índice, puede ser texto o un entero, que hace referencia al valor enviado.

Cotroller | Set	
Lenguaje	**Formato**
PHP	$this->**set**(indice, valor_enviado)
JAVA	this.**set**(indice, valor_enviado)
JavaScript	this.**set**(indice, valor_enviado)
C#	this.**set**(indice, valor_enviado)
indice puede ser un texto o un entero.	

Tabla 4.27. Formato de la función set, envío con índice.

Para los lenguajes altamente tipados (Java, C#), se considera como valor enviado un objeto genérico, por esta razón al ser recibido en la vista, tiene que ser convertido en el tipo requerido. Para los lenguajes bajamente tipados (Php, JavaScript) es innecesario.

b. Desarrollar el método dentro del controlador

PHP

```
function set($dat1,$dat2=null){
        $this->view->setData($dat1,$dat2);
}

// funciones auxiliares

function setData($data1,$data2){
  if($data2===null)
    $this->obj = $data1;
  else{
    if(is_int($data1))
      $this->objy[$data1]=$data2;
    else
      $this->objx[$data1]=$data2;
  }
}
```

Figura 4.52. Código de envío de datos a la vista en Php

JavaScript

```
set(obj1,obj2){
        if(obj2===undefined)
                this.kview.setData1(obj1);
        else
                this.kview.setData2(obj1,obj2);
}

// funciones auxiliares

setData1(obj1){
        this.obj = obj1;
}

setData2(obj1,obj2){
        if (typeof obj1 === 'string' || obj1 instanceof String)  {
                this.objx[obj1] = obj2;
                this.x++;
        }
        else{
                this.objy[obj1] = obj2;
                this.y++;
        }
}
```

Figura 4.53. Código de envío de datos a la vista en JavaScript

JAVA

```
public void set(Object value){
   this.data.setData(value);
   this.request.setAttribute("data", this.data);
}

public void set(String stringIndex,Object value){
   this.data.setData(stringIndex,value);
   this.request.setAttribute("data", this.data);
}

public void set(int index,Object value){
   this.data.setData(index,value);
   this.request.setAttribute("data", this.data);
}

// Funciones auxiliares

public void setData(Object value){
   this.obj = value;
}

public void setData(String stringIndex,Object value){
   Object[] aux = new Object[this.objx.length+1];
   String[] str = new String[this.objx.length+1];
   for (int i = 0; i < this.objx.length; i++) {
      aux[i]=this.objx[i];
      str[i]=this.strIndex[i];
   }
   aux[this.objx.length]=value;
   str[this.objx.length]=stringIndex;
   this.objx = aux;
   this.strIndex = str;
}

public void setData(int index,Object value){
   Object[] aux = new Object[this.objy.length+1];
   int[] str = new int[this.objy.length+1];
   for (int i = 0; i < this.objy.length; i++) {
      aux[i]=this.objy[i];
      str[i]=this.intIndex[i];
   }
   aux[this.objy.length]=value;
   str[this.objy.length]=index;
   this.objy = aux;
   this.intIndex = str;
}
```

Figura 4.54. Código de envío de datos a la vista en Java.

C#

```
public void set(String stringIndex, Object value){
   this.viewk.data(stringIndex, value);
}

public void set(int index, Object value){
   this.viewk.data(index, value);
}

// Funciones auxiliares

public void setData(Object value){
   this.obj = value;
}

public void setData(String stringIndex, Object value){
   object[] aux = new object[this.objx.Length + 1];
   string[] str = new string[this.objx.Length + 1];
   for (int i = 0; i < this.objx.Length; i++)
   {
      aux[i] = this.objx[i];
      str[i] = this.strIndex[i];
   }
   aux[this.objx.Length] = value;
   str[this.objx.Length] = stringIndex;
   this.objx = aux;
   this.strIndex = str;
}

public void setData(int index, Object value){
   object[] aux = new object[this.objy.Length + 1];
   int[] str = new int[this.objy.Length + 1];
   for (int i = 0; i < this.objy.Length; i++)
   {
      aux[i] = this.objy[i];
      str[i] = this.intIndex[i];
   }
   aux[this.objy.Length] = value;
   str[this.objy.Length] = index;
   this.objy = aux;
   this.intIndex = str;
}
```

Figura 4.55. Código de envío de datos a la vista en C#.

c. Definir el formato de recuperación de datos

En la tarea anterior se describió como enviar datos utilizando la función **set** desde el controlador a la vista, la clase Views debe recuperar los datos desde el lado de la vista, para ello definimos la función:

get()

Esta función no tiene parámetro porque hace referencia al envío de un dato sin índice, y nos devuelve dicho valor. La forma de llamar a esta función tiene ciertas variaciones para cada lenguaje.

Views | Get	
Lenguaje	**Formato**
PHP	$v->**get**()
JAVA	Views v = new Views(request); v.**get**()
JavaScript	v.**get**()
C#	Views v = new Views(Context); v.**get**()

Tabla 4.28. Formato de la función set, envío con índice.

Como se muestra en la tabla anterior, para lenguajes como Java y C# se tiene que importar y declarar el objeto de la clase Views. Para los casos donde se envió datos con índice, el formato de recuperación sería el siguiente:

get(indice)

Se debe enviar el índice como parámetro, este puede ser un texto o un entero, esta función nos devuelve el valor que corresponde a dicho índice.

Views | Get	
Lenguaje	**Formato**
PHP	$v->**get**(indice)
JAVA	Views v = new Views(request); v.**get**(indice)
JavaScript	v.**get**(indice)

C#	Views v = new Views(Context); v.**get**(indice)
indice puede ser un texto o un entero.	

Tabla 4.29. Formato de la función set, envío con índice.

El envío y recuperación de datos se establecieron pensando en todos los leguajes de programación, en algunos casos puede que no sea necesario utilizar índices, ya que leguajes bajamente tipados como php, nos permiten crear arrays de diferentes tipos de datos, además que se puede usar como índice cadenas de texto.

d. Desarrollar el método dentro de la vista

JAVA

```
public <T> T get(){
    return (T) this.obj;
}

public <T> T get(String index){
   int aux=-1;
   for (int i = 0; i < this.strIndex.length; i++) {
      aux=i;
      if(index.equalsIgnoreCase(this.strIndex[i])) break;
   }
   return (T) this.objx[aux];
}

public <T> T get(int index){
   int aux=-1;
   for (int i = 0; i < this.intIndex.length; i++) {
      aux=i;
      if(index==this.intIndex[i]) break;
   }
   return (T) this.objy[aux];
}
```

Figura 4.56. Código para obtener datos enviados en Java.

JavaScript

```
get(val){
	if(val===undefined)
		return	this.data.get1();

	return this.data.get2(val);
}

// funciones auxiliares
get1(){
	return this.obj;
}

get2(val){
	//if(Number.isInteger(val))
		if(typeof val === "number")
		return this.objy[val];
	else
		return this.objx[val];
}
```

Figura 4.57. Código para obtener datos enviados en JavaScript.

C#

```
public object get(){
   return this.obj;
}

public object get(String index){
   int aux = -1;
   for (int i = 0; i < this.strIndex.Length; i++)
   {
      aux = i;
      if (index.Equals(this.strIndex[i],
StringComparison.OrdinalIgnoreCase)) break;
   }
   return this.objx[aux];
}

public object get(int index){
   int aux = -1;
   for (int i = 0; i < this.intIndex.Length; i++)
   {
      aux = i;
      if (index == this.intIndex[i]) break;
   }
   return this.objy[aux];
}
```

Figura 4.58. Código para obtener datos enviados en C#.

PHP

```
function get($val=null){
      if($val===null) {
         return $this->obj;
      }
      else{
         if(is_int($val))
            return $this->objy[$val];
         else
            return $this->objx[$val];
      }
}
```

Figura 4.59. Código para obtener datos enviados en Php.

e. Definir el formato de selección del partial

Los partial son funciones que nos devuelven un valor, o un fragmento de código para ser utilizado en la vista. Por ejemplo, si tenemos el caso de que siempre se repite un formulario para diferentes páginas, podemos declararlo como un partial. Definimos la función:

partial(“nombre del partial”)

Al llamar a la función partial, enviamos como parámetro el nombre del partial.

Views | Partial		
Lenguaje	**Formato**	**Extensión**
PHP	<?php include $v->**partial**(“nombre”) ?>	.php
JAVA	Views v = new Views(request); <jsp:include page="<%= v.**partial**(“nombre”) %>" />	.jsp
JavaScript	<%- include(v.**partial**(“nombre”)) %>	.ejs
C#	Views v = new Views(Context); <% Server.Execute(v.**partial**(“nombre”)); %>	.aspx

Tabla 4.30. Formato de la función partials con sus extensiones.

Si se necesita datos de variables en los partial, estos pueden ser recuperados utilizando la función **get()** de la vista.

Los partial se encuentran ubicados dentro de la carpeta **view**, en la carpeta _partials.

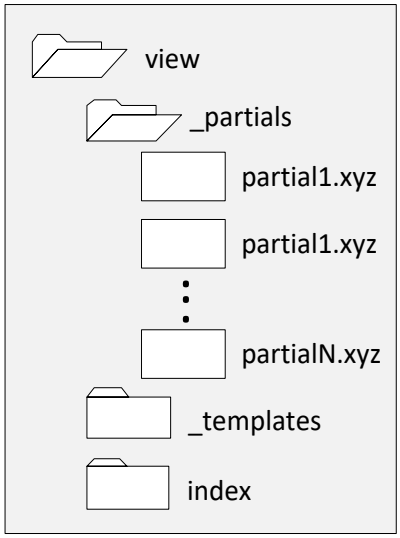


Figura 4.60. Estructura de los ficheros de los partials.

f. Desarrollar el método dentro de la vista

PHP

```
function partial($name){
        return "app/view/_partials/".$name.".php";
}
```

Figura 4.61. Código de los Partials en Php.

JAVA

```
public  String  partial( String name ){
        return "/view/_partials/"+name+".jsp";
}
```

Figura 4.62. Código de los Partials en Java.

JavaScript

```
partial( name ){
        return "app/view/_partials/"+name+".ejs";
}
```

Figura 4.63. Código de los Partials en JavaScript.

C#

```
public string partial( string name ){
        return "app/view/_partials/"+name+".aspx";
}
```

Figura 4.64. Código de los Partials en C#.

SPRINT 4: BIBLIOTECA DEL MODELO

a. Definir el formato de importación del modelo

Los modelos creados por el usuario del framework, estarán en la carpeta **model**.

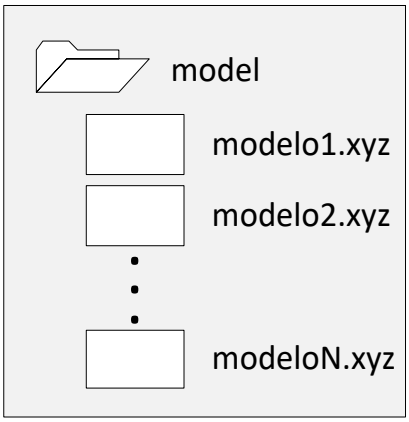


Figura 4.65 Carpeta que contiene los modelos.

Modelo	
Lenguaje	**Extensión**
PHP	.php
JAVA	.java
JavaScript	.js
C#	.cs

Tabla 4.31. Extensión de las clases modelo.

Desde el controlador podemos llamar a la instancia del modelo de la siguiente forma.

model(“nombre del modelo”)

Enviamos como parámetro el nombre del modelo, esta función nos devuelve un objeto instanciado de esa clase.

Cotroller | Model	
Lenguaje	**Formato**
PHP	$this->**model**(“nombre”)
JAVA	this.**model**(“nombre”)
JavaScript	this.**model**(“nombre”)
C#	this.**model**(“nombre”)

Tabla 4.32. Formato de la función model.

b. Desarrollar el método dentro del controlador

PHP

```
function model( $model ){
        if (!file_exists ('app/model/'.$model.'.php'))
                die('No exite el modelo: '.$model);

        require ('app/model/'.$model.'.php');
        return new $model();
}
```

Figura 4.66. Código obtención del modelo en Php

JAVA

```
public <T> T model(String model){
    try{
            Class<?> c = Class.forName("model."+model);
            Object o = c.newInstance();
            return (T) o;
    }catch(Exception e){ }
    return null;
}
```

Figura 4.67. Código obtención del modelo en Java.

JavaScript

```
model(modelName){
        var Model = require("app/model/"+modelName);
        var obj = new Model();
        return obj;
}
```

Figura 4.68. Código obtención del modelo en JavaScript.

C#

```
public object model(String model){
   try{
      return Activator.CreateInstance(Type.GetType("web.app.model." + model));
   }
   catch (Exception) {
      return null;
   }
}
```

Figura 4.69. Código obtención del modelo en C#

c. Definir el formato de consultas simples de tabla

En muchos casos el usuario del framework necesita guardar información de forma permanente, para ello hace uso de una base de datos relacional. Se ha desarrollado en el framework un método para realizar consultas y sentencias a la base de datos, utilizando código sql.

query(“Codigo sql”)

Enviamos como parámetro código sql, generalmente para consultas utilizamos SELECT, y para sentencias utilizamos INSERT, DELETE, UPDATE.

Model | Table	
Lenguaje	**Formato**
PHP	$this->**query**(“Codigo sql”)
JAVA	this.**query**(“Codigo sql”)

JavaScript	this.**query**(“Codigo sql”)
C#	this.**query**(“Codigo sql”)

Tabla 4.33. Formato del método query.

El método **query()** retorna un array bidimensional de tipo [x][y], donde “x” es la fila de la tabla y “y” es la columna de la tabla consultada. Si la consulta devuelve vacío, la función **query()** devuelve **null**.

d. Desarrollar la función en el modelo

PHP

```
function query($sql) {
        if(strcasecmp(substr(trim($sql),0,6),'select')!=0) return $result;

        $c=0;
        $lista=array();
        while( $row = mysqli_fetch_array($result)) {
                if(sizeof($row)/2==2)
                        $lista[$c] = $row[0];
                else
                        for ($i=0; $i < sizeof($row)/2; $i++) {
                                $lista[$c][$i] = $row[$i];
                        }
                $c++;
        }
        return $lista;
}
```

Figura 4.70. Código del método query en Php.

JavaScript

```
query( sql ) {

  Statement query = Connection.start();

  if(!sql.trim().substring(0,6).equalsIgnoreCase("select")){

     try {
        query.execute(sql);
        Connection.con.close();
        return (T)"1";
     } catch (Exception ex) { }
  }
}
```

Figura 4.71. Código del método query en JavaScript.

JAVA

```
static <T> T query(String sql) {
  Statement query = Connection.start();
  if(!sql.trim().substring(0,6).equalsIgnoreCase("select")){
     try {
        query.execute(sql);
        Connection.con.close();
        return (T)"1";
     } catch (Exception ex) { }
  }

   try {
      ResultSet tabla = query.executeQuery(sql);
      ResultSetMetaData rsmd = tabla.getMetaData();

      int n = rsmd.getColumnCount();
      if(n>1){
         ArrayList<String[]> a=new ArrayList<>();
         while(tabla.next()){
            ArrayList<String> b=new ArrayList<>();
            for (int i = 0; i < n; i++) {
               b.add(tabla.getString(i+1));
            }
            String[] c = new String[b.size()];
            for (int i = 0; i < b.size(); i++) {
               c[i]=b.get(i);
            }
            a.add(c);
         }

         String[][] d = new String[a.size()][n];
         for (int i = 0; i < a.size(); i++) {
            d[i] = a.get(i);
         }
         if(d.length==1) return (T)d[0];
         return (T)d;
      }else{
         ArrayList<String> a=new ArrayList<>();
         while(tabla.next()){
            a.add(tabla.getString(1));
         }

         String[] b = new String[a.size()];
         for (int i = 0; i < a.size(); i++) {
            b[i]=a.get(i);
         }
         if(b.length==1) return (T)b[0];
         return (T)b;
      }
   }catch (SQLException e) { }
}
```

Figura 4.72. Código del método query en Php.

C#

```
Public object query( string sql ) {

  Statement query = Connection.start();

  if(!sql.trim().substring(0,6).equalsIgnoreCase("select")){

     try {
        query.execute(sql);
        Connection.con.close();
        return (T)"1";
     } catch (Exception ex) { }
  }
  try {
        query.execute(sql);
        Connection.con.close();
  } catch (Exception ex) { }

}
```

Figura 4.73. Código del método query en C#.

e. Definir el formato de instanciación de la tabla consultada

Para poder facilitar el trabajo del usuario, se ha desarrollado para el framework, el procedimiento Table, donde realiza una consulta a la base de datos e instancia un objeto de la clase Table, de la siguiente forma:

table(“Consulta sql”)

Enviamos como parámetro la consulta a la base de datos, este método instancia la clase auxiliar Table, quien va a contener la tabla consulta.

Model | Table	
Lenguaje	**Formato**
PHP	$this->**table**(“Consulta sql”)
JAVA	this.**table**(“Consulta sql”)
JavaScript	this.**table**(“Consulta sql”)
C#	this.**table**(“Consulta sql”)

Tabla 4.34. Formato del procedimiento table, al instanciar la clase Table.

f. Definir el formato de retorno de valores de la tabla instanciada

En la tarea anterior definimos como instanciar la clase auxiliar Table; una vez instanciado podemos realizar consultas a este objeto. Aprovechando el polimorfismo de los lenguajes de programación orientado a objetos, definimos la forma de la siguiente manera:

table(índice_fila, índice_columna)

Enviamos como parámetro el índice de la fila, y como segundo parámetro el índice de la columna, ambos como enteros. También se desarrolló para el método el envío del nombre de la columna de la tabla de la base de datos como segundo parámetro, es decir:

table(índice_fila, "nombre de la columna")

Esta forma, nos permite consultar a la tabla a través de los nombres de la columna, por ejemplo: Si tenemos la tabla persona con los campos nombre y edad, utilizando el formato instanciaríamos como table("Select * from Persona") y para obtener la edad del primer registro, table(0,"edad").

Model | Table	
Lenguaje	**Formato**
PHP	$this->**table**(índice_fila, índice_columna | "nombre de columna")
JAVA	this.**table**(índice_fila, índice_columna | "nombre de columna")
JavaScript	this.**table**(índice_fila, índice_columna | "nombre de columna")
C#	this.**table**(índice_fila, índice_columna | "nombre de columna")

Tabla 4.35. Formato de la función table, al realizar una consulta al objeto Table.

En muchos casos el usuario necesita consultar una determinada fila de la tabla consultada, para ello utilizamos la forma:

table(índice_fila)

Enviamos como parámetro el índice de la fila, y la función nos retorna un array de string de la fila requerida.

El método table también nos permite devolver el objeto instanciado de la clase Table,

para ello no enviamos ningún parámetro, y el framework entenderá que se está requiriendo devolver el objeto.

table()

Al llamar al método table sin enviar parámetros, este nos retorna el objeto instanciado de la clase Table. Esta forma nos permite aprovechar el encapsulamiento del objeto y poder consultar el número de filas o columnas:

table().rows() : Retorna el número de filas de la tabla consultada.

table().columns() : Retorna el número de columnas de la tabla consultada.

table().get() : Retorna todo el contenido de la tabla consultada en un array bidimensional de tipo string.

g. Desarrollar los métodos en el modelo

JAVA

```
static Table getTable(String sql){
   Table table = null;
   Statement query = Connection.start();
   try {
      ResultSet tabla = query.executeQuery(sql);
      ResultSetMetaData rsmd = tabla.getMetaData();

      int n = rsmd.getColumnCount();
      String[] columnNames = new String[n];
      for (int i = 0; i < n; i++) {
         columnNames[i]=rsmd.getColumnName(i+1);
      }

      table = new Table(columnNames);

      while(tabla.next()){
         String[] row = new String[n];
         for (int i = 0; i < n; i++) {
            row[i] = tabla.getString(i+1);
         }
         table.add(row);
      }
      Connection.con.close();
   }catch (Exception e) {  }
   return table;
}
```

Figura 4.74. Código del método Table den Java.

PHP

```
public function table($row = null,$column = null){
        if($row === null && $column === null) return $this->table;

        if(is_string($row) && $column === null) {
                $this->table = Connection::getTable($row); return;}
        return $this->table->get($row,$column);
}
//funciones auxiliares
function getTable($sql) {
        while( $row = mysqli_fetch_array($result)) {
                for ($i=0; $i < sizeof($row)/2; $i++) {
                                $lista[$c][$i] = $row[$i];
                                $names=array_keys($row) ;
                }$c++;
        } $c=0;
        foreach ($names as $value) {
                if(!is_int($value)){
                        $aaa[$c]=$value; $c++;
                }
        }
        return $aaa;
}
```

Figura 4.75. Código del método Table den PHP

JavaScript

```
table(String sql){

   Table table = null;

   Statement query = Connection.start();
   int n = rsmd.getColumnCount();
      String[] columnNames = new String[n];
      for (int i = 0; i < n; i++) {
         columnNames[i]=rsmd.getColumnName(i+1);
      }

   try {
      ResultSet tabla = query.executeQuery(sql);
      ResultSetMetaData rsmd = tabla.getMetaData();

     for (int i = 0; i < n; i++) {
         columnNames[i]=rsmd.getColumnName(i+1);
      }
      int n = rsmd.getColumnCount();
      String[] columnNames = new String[n];
      for (int i = 0; i < n; i++) {
         columnNames[i]=rsmd.getColumnName(i+1);
      }
}
```

Figura 4.76. Código del método Table den JavaScript.

C#

```
Public object table(String sql){

   Table table = null;

   Statement query = Connection.start();
   int n = rsmd.getColumnCount();
      String[] columnNames = new String[n];
      for (int i = 0; i < n; i++) {
         columnNames[i]=rsmd.getColumnName(i+1);
      }

   try {
      ResultSet tabla = query.executeQuery(sql);
      ResultSetMetaData rsmd = tabla.getMetaData();

     for (int i = 0; i < n; i++) {
         columnNames[i]=rsmd.getColumnName(i+1);
      }
      int n = rsmd.getColumnCount();
      String[] columnNames = new String[n];
      for (int i = 0; i < n; i++) {
         columnNames[i]=rsmd.getColumnName(i+1);
      }
}
```

Figura 4.77. Código del método Table den C#.

4.2.2 REUNION DIARIA DE LOS SPRINT

Las reuniones se realizaron a diario durante la ejecución de los sprint, a continuación, mostramos las respuestas más importantes de las preguntas de las reuniones diarias (daily meeting).

SPRINT 1

- **¿Qué se ha hecho de nuevo con respecto a la última reunión diaria?**
  - Se realizo el diseño de arquitectura técnica.
  - Se diseño el enrutamiento de las peticiones request.

- **¿Qué será lo siguiente a realizar?**
  - Estructuración de ficheros en diferentes lenguajes.
  - Implementar el enrutamiento.

- **¿Qué problemas hay para realizarlo?**
    - Netbeans no permite crear carpetas del lado del servidor.
    - JavaScript con NodeJS, usan por defecto la carpeta node_modules como contenedor de sus librerías.

SPRINT 2

- **¿Qué se ha hecho de nuevo con respecto a la última reunión diaria?**
    - Se realizó funciones del controlador que tiene relación con la vista.
    - Se implementó las funciones básicas del controlador.

- **¿Qué será lo siguiente a realizar?**
    - Implementar la obtención de parámetros POST.
    - Implementar el redireccionamiento.

- **¿Qué problemas hay para realizarlo?**
    - Los leguajes como PHP y JavaScript no permiten funciones polimórficas, es decir solo debe existir un método con el mismo nombre.
    - El retorno de valores en lenguajes altamente tipados, como C# y Java se tiene que realizar una conversión de tipo previamente a usar el valor devuelto.

SPRINT 3

- **¿Qué se ha hecho de nuevo con respecto a la última reunión diaria?**
    - Se realizó funciones polimórficas para enviar datos a la vista.
    - Se implementó la recuperación de datos en la vista.

- **¿Qué será lo siguiente a realizar?**
    - Estructuración de ficheros de los partials.
    - Implementar el envío de texto sin pasar por la vista.

- **¿Qué problemas hay para realizarlo?**
    - En los leguajes altamente tipados, se tiene que hacer una conversión de tipo en el lado de la vista.

Para poder hacer una inclusión de un fragmento de código html, existe mucha variación entre las vistas de los lenguajes de programación dificultando la estandarización.

SPRINT 4

- **¿Qué se ha hecho de nuevo con respecto a la última reunión diaria?**
  - Se desarrolló la función para importar el modelo en el controlador.
  - Se implementó la función de consulta simple en el modelo.
  - Se implementó la función de consulta con la clase table.

- **¿Qué será lo siguiente a realizar?**
  - Crear una clase auxiliar para la conexión a la base de datos.
  - Implementar las funciones de consulta de tablas.

- **¿Qué problemas hay para realizarlo?**
  - Importar daros en un lenguaje altamente tipado, requiere de la importación de la clase para poder hacer una conversión al modelo.
  - La sintaxis SQL tiene ciertas variaciones en diferentes bases de datos, va a depender del usuario del framework utilizar la base de datos de su preferencia.

4.2.3 GRAFICAS DE TRABAJO PENDIENTE

SPRINT 1

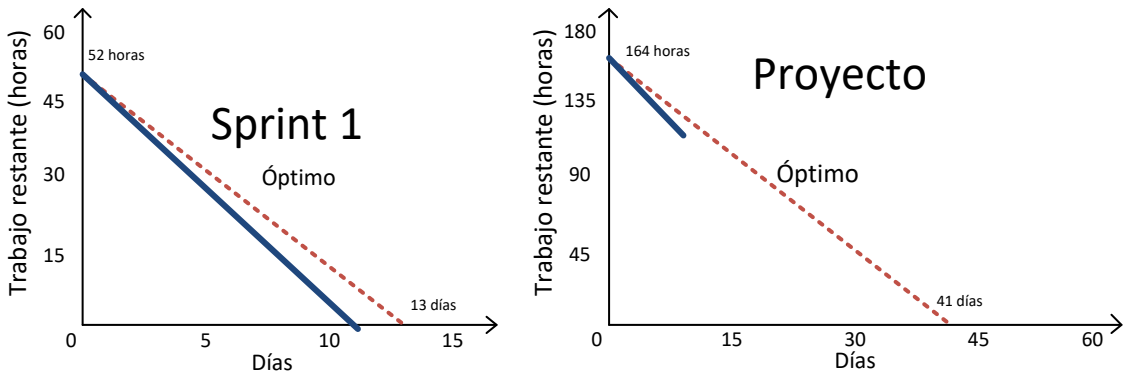


Figura 4.78. Trabajo pendiente del Sprint 1 y del proyecto al finalizar el sprint 1.

SPRINT 2

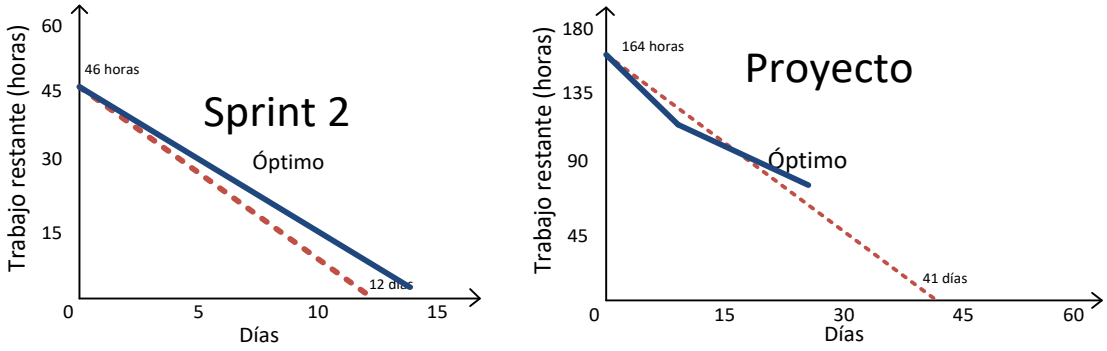


Figura 4.79. Trabajo pendiente del Sprint 2 y del proyecto al finalizar el sprint 2.

SPRINT 3

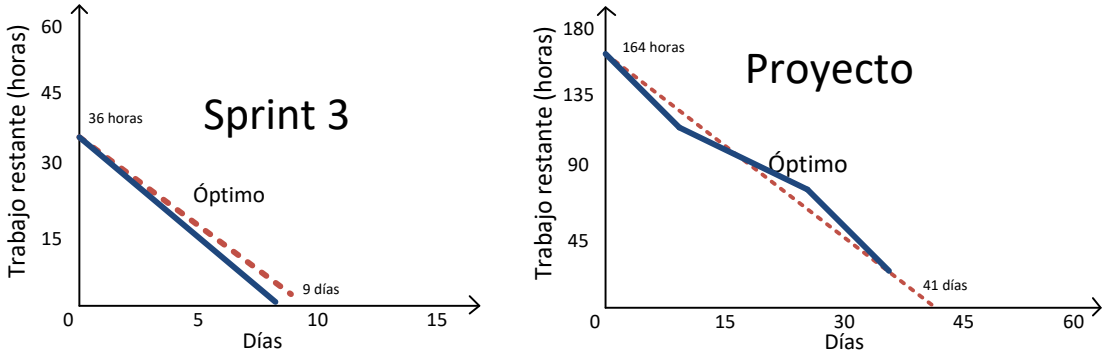


Figura 4.80. Trabajo pendiente del Sprint 3 y del proyecto al finalizar el sprint 3.

SPRINT 4

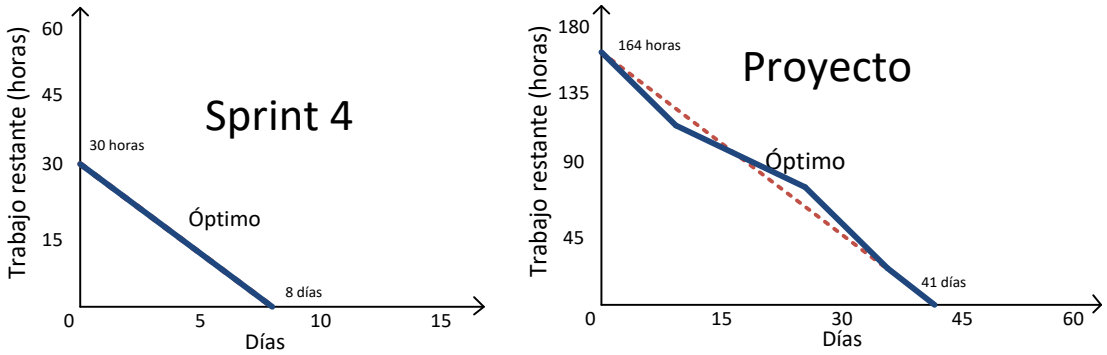


Figura 4.81. Trabajo pendiente del Sprint 4 y del proyecto al finalizar el sprint 4.

4.2.4 REVISÓN DE LOS SPRINT

La revisión de los sprint se realizó al finalizar cada sprint, donde el propietario del producto y los socios dieron el visto bueno al trabajo realizado. Para comprobar el correcto funcionamiento de los sprint, desarrollamos un test de pruebas para cada sprint.

SPRINT 1: ARQUITECTURA ESTÁNDAR Y ENRUTAMIENTO

a. Diseñar una arquitectura

La arquitectura adoptada está presente en todo el framework, por ello las pruebas siguientes demuestran el funcionamiento correcto de esta arquitectura.

No es necesario instalar el framework, solo se debe abrir el framework como un proyecto nuevo en el editor de código de preferencia y en el lenguaje que desee, al ejecutar el proyecto debe ingresar a un navegador web e ingresar la dirección url de la ejecución activa, por lo general **http://localhost/web** y debe mostrar la siguiente página de bienvenida.

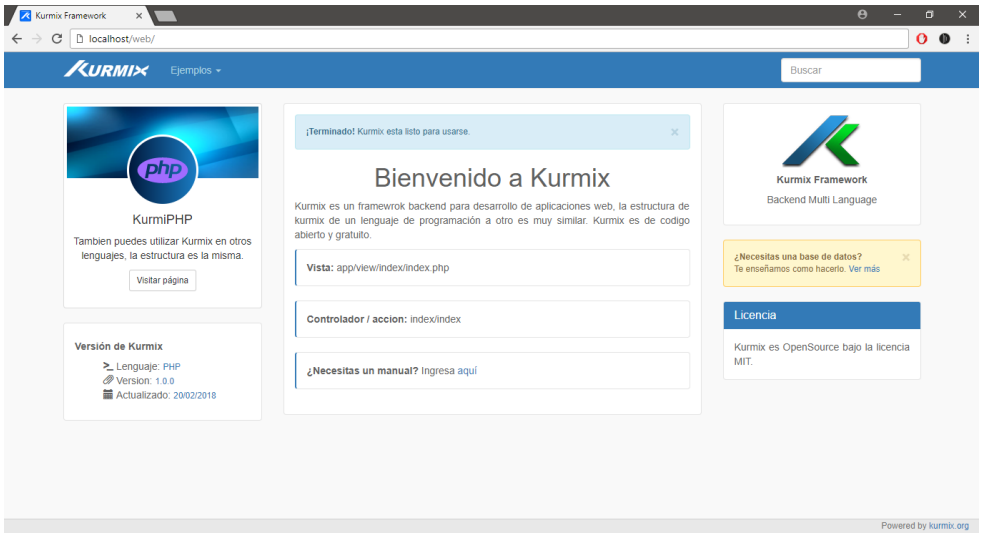


Figura 4.82. Página de bienvenida al iniciar el framework.

b. Enrutar peticiones (request)

Creamos el controlador “calculadora” y dentro la función “sumar”, dentro de esta función llamamos al método heredado: *this.write(“Aquí mostraremos la suma”)*. Desde la url llamamos a nuestra acción del controlador de la forma:

Figura 4.83. Petición y respuesta desde la url.

La llamada desde la url *?k=calculadora/sumar*, se ha enrutado al controlador calcular y la acción sumar. Con lo cual probamos que el enrutamiento funciona correctamente.

SPRINT 2: BIBLIOTECA DE CONTROLADOR

a. Seleccionar plantillas y vistas

El framework trae como ejemplo el uso de plantillas y vistas dentro del código. Desde la función "otra plantilla" se llama al método heredado *this.template("esmeralda")* y a método *this.view("index/index"*). Para ver el ejemplo, llamamos desde la url de la forma: ?k=index/otra_plantilla.

Figura 4.84. Selección de template y vista.

Al llamar a dicha url, no muestra la vista del index con otra plantilla, en este caso la platilla esmeralda. De este modo probamos que funciona correctamente.

b. Redireccionar peticiones

En el controlador "calculadora" creado anteriormente, creamos una nueva función con el nombre "restar", dentro de ella llamamos al método heredado *this.redirect("calculadora/sumar").* Ejecutamos la acción desde la url.

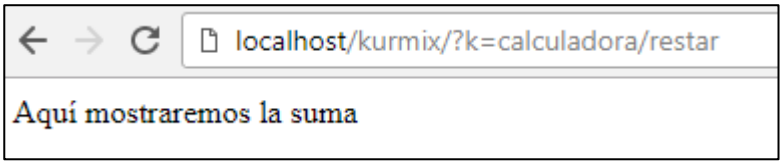


Figura 4.85. Redireccionamiento de la acción restar a la acción sumar.

Si bien se llamó a la acción "restar", esta ha sido redirigida a la acción "sumar". Con el cual demostramos que el redireccionamiento de petición funciona correctamente.

c. Imprimir desde el controlado

Dentro del controlador "calculadora", creamos una función con el nombre "mostrar" y llamamos a la función *this.write("<h1>Prueba</h1>imprimir html")***.** Y llamamos a la acción desde la url. Con lo cual probamos que podemos imprimir html como texto plano desde el controlador.

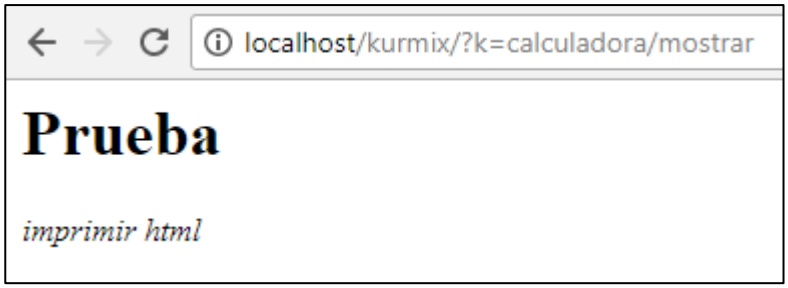


Figura 4.86. Imprimir html desde el controlador.

d. Obtener parámetros

Modificamos la función "sumar" dentro del controlador "calculadora", agregamos dos parámetros en la función, por ejemplo para el caso en el lenguaje php, tendríamos: *function sumar($a,$b).* Dentro de la función llamamos al *método this.write("La suma de "+$a+" y "+$b+" es "+($a+$b)).* Llamamos a la acción desde la url.

Figura 4.87. Obtener parámetros desde la url.

e. Obtener sesiones

Creamos una función dentro del controlador "calculadora" con el nombre "guardar", y agregamos le método *this.session("nombre","Andree")* y *this.write("guardado")*. Luego en la función "mostrar" agregamos *String a = this.session("nombre")* y *this.write(a)*. Llamamos a las dos acciones desde la url.

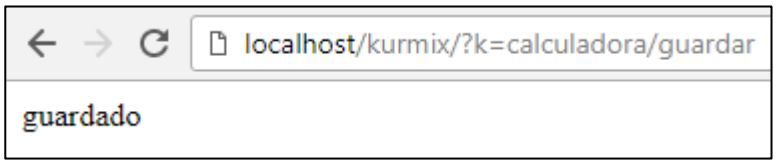


Figura 4.88. Ejecución de la acción guardar.

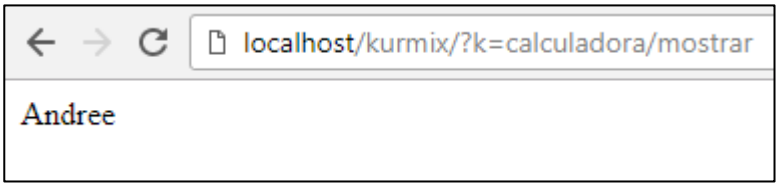


Figura 4.89. Ejecución de la acción mostrar.

Demostramos que la sesión ha guardado el valor enviado con índice "nombre" y valor "Andree".

SPRINT 3: BIBLIOTECA DE VISTA

a. Enviar y recuperar datos en la vista

Creamos una función con el nombre "enviar" en el controlador "calculadora", y agregamos el método: *this.set("Hola mundo..!")*. Dentro de la carpeta View, creamos una carpeta con el nombre calculadora, dentro de ella un fichero con el nombre enviar seguido de la extensión según el lenguaje. Dentro de este fichero llamamos al método get del objeto view: *v.get()*. Ejecutamos la acción, con el cual demostramos que se puede enviar datos desde el controlador y recibirlos en la vista desde el objeto view.

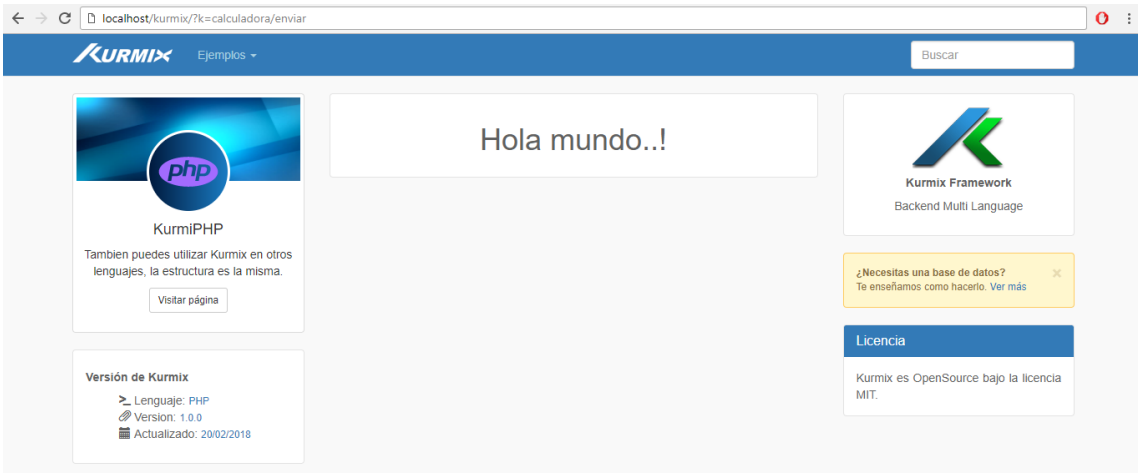


Figura 4.90. Ejecución del envío y recepción de datos desde el controlador.

b. Imprimir texto y código en la vista (patials)

Dentro de la carpeta partial se encuentra algunos ejemplos que el framework trae por defecto, al ejecutar la plantilla estos llaman a los elementos dentro de la carpeta partials, como por ejemplo un menú, o el logo. Con lo cual demostramos que estas funcionando correctamente.

SPRINT 4: BIBLIOTECA DE MODELO

a. Importar el modelo y realizar consultas y sentencias

Dentro de la carpeta Model, creamos un fichero con el nombre "usuario", dentro de ella creamos la clase "usuario" que hereda de la clase "Model", dentro de ella creamos una función mostrar_usuarios, y llamamos al método *this.query("SELECT * FROM usuario")*. Desde el controlador "calculadora" creamos una funcion con el nombre "usuarios" y llamamos al método heredado: *var u=this.model("usuario"); this.write(u.mostrar_uuarios).* Ejecutamos la accion, y verificamos que se muestren los datos consultados a la base de datos.

Figura 4.90. Ejecución de consulta a la base de datos.

4.2.5 RETROSPECTIVA Y REFINAMIENTO

La retrospectiva se realizó al final de cada sprint; por ser un proyecto unipersonal, mostramos a continuación una retrospectiva general del desarrollo de los 4 sprint.

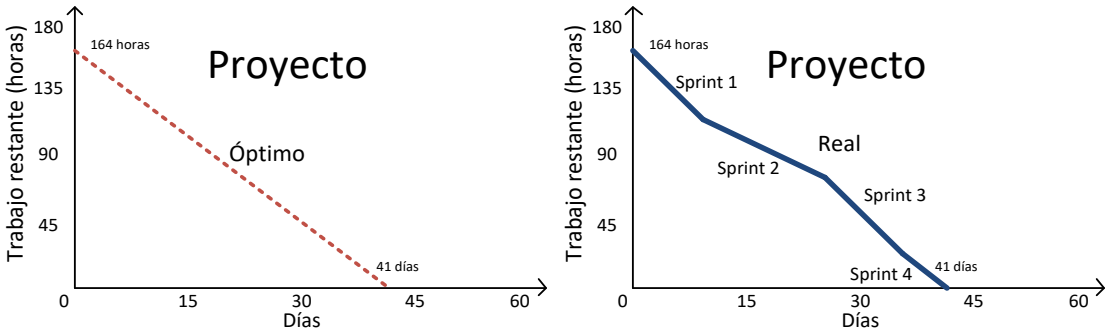


Figura 4.22. Trabajo pendiente del proyecto al finalizado, curva optima y curva real.

Como muestra la figura anterior, se pudo terminar en el tiempo establecido. Scrum exige respetar los tiempos establecidos, sin embargo, podemos notar que la curva en ocasiones se ha salido de la línea optima esperada.

- En el primer sprint la curva tiende a pegarse hacia el eje Y, quiere decir que las tareas asignadas para este sprint fueron menores respecto a la estimación en el tiempo de trabajo.

- El segundo sprint tiende a alejarse del eje Y, quiere decir que las tareas asignadas fueron mayores respecto a la estimación en los tiempos de trabajo.

- En el tercer y cuarto sprint las curvas se desvían ligeramente, llegando a equilibrase para el final del desarrollo.

Como se había mencionado en el marco teórico, la retrospectiva de Scrum responde a tres preguntas:

¿Qué hemos hecho bien?

- Se pudo culminar el proyecto en el tiempo establecido.
- Se pudo desarrollar en cuatro lenguajes diferentes al mismo tiempo.

¿Qué debemos mejorar?

- Debemos mejorar la estimación de los tiempos para cada sprint.
- Debemos mejorar las pruebas en diferentes escenarios del framework.

¿Qué no debemos seguir haciendo?

- No debemos dejar para después las pruebas de lo que hoy se desarrolló.
- No debemos dejar para después los comentarios en el código fuente.

CAPÍTULO V
CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

a. Según el marco teórico desarrollado en el capítulo II, sección 2.2.9 y sección 2.2.10; las técnicas para aplicar el marco de trabajo Scrum desarrollado en el capítulo III, sección 3.5.4; los artefactos obtenidos en el capítulo IV; los requerimientos del framework tabla N° 4.1, la pila del proyecto en la tabla N° 4.2, las tareas desarrollados en los sprints en las tablas N° 4.3, 4.4, 4.5, y 4.6, y la revisión de los sprint 1,2,3 y 4 en la sección 4.2.4. Se concluye que se puede desarrollar un framework a medida de los requerimientos del cliente (programador).

b. Según el marco teórico desarrollado en el capítulo II, sección 2.2.4 y sección 2.2.9; las técnicas para aplicar el marco de trabajo Scrum desarrollado en el capítulo III, sección 3.5.4; los artefactos obtenidos en el capítulo IV; La pila de historias de usuario de la tabla N° 4.3, las tareas desarrollados en el sprint 1, de la tabla N° 4.7, análisis y diseño de la arquitectura en la figura 4.16, 4.22 y 4.25. Se concluye que para la implementación del framework se puede utilizar una arquitectura estándar basado en la arquitectura de frameworks existentes.

c. Según el marco teórico desarrollado en el capítulo II, sección 2.2.14 y sección 2.2.9; las técnicas para aplicar el marco de trabajo Scrum desarrollado en el capítulo III, sección 3.5.4; los artefactos obtenidos en el capítulo IV; La pila de historias de usuario de la tabla N° 4.6, las tareas desarrollados del sprint 4, de la tabla N° 4.10, obteniendo las herramientas para consultas sql mostradas en las tablas N° 4.32, 4.33 y 4.34. Se concluye que se puede desarrollar un modelo de conexión estándar a la base de datos, y así utilizar consultas y sentencias SQL puras evitando inyecciones SQL con herramientas del framework y buenas prácticas de programación.

d. Según el marco teórico desarrollado en el capítulo II, sección 2.2.9; las técnicas para aplicar el marco de trabajo Scrum desarrollado en el capítulo III, sección 3.5.4; los artefactos obtenidos en el capítulo IV; La pila de historias de usuario de la tabla N° 4.5, las tareas desarrollados del sprint 3, de la tabla N° 4.9, obteniendo

las herramientas para enviar datos enviados como se muestra en la tabla N° 4.26 y 4.27, y obtener dichos datos como se muestra en las tablas N° 4.29 y 4.30. Se concluye que se puede desarrollar la renderización de vistas y plantillas en aplicaciones web obteniendo datos enviados desde el controlador.

e. Según el marco teórico desarrollado en el capítulo II, sección 2.2.4 y 2.2.9; las técnicas para aplicar el marco de trabajo Scrum desarrollado en el capítulo III, sección 3.5.4; los artefactos obtenidos en el capítulo IV; La pila de historias de usuario de la tabla N° 4.4, las tareas desarrollados del sprint 2, de la tabla N° 4.8, obteniendo las herramientas para selección de plantillas y vistas como se muestra en la tabla N° 4.18 y 4.19, y herramientas para manejo de sesiones y parámetros como se muestra en las tablas N° 4.23 y 4.22. Se concluye que son necesarias la utilización de funciones básicas desarrollado en el controlador para la implementación de aplicaciones web, selección de plantillas y vistas, manejo de sesiones, obtención de parámetros, etc.

5.2 RECOMENDACIONES

a. Se debe investigar otros framework en los lenguajes de programación Python, Go, Perl, Ruby, VisualBasic.Net, etc, a fin de ampliar los alcances del framework.

b. Al ser JavaScript potente y rápido en ejecuciones asíncronas, se recomienda ampliar el framework en su versión de JavaScript para poder realizar consultas a la base de datos de forma asíncrona.

c. Se debe investigar la adaptación del framework para el desarrollo de aplicaciones de escritorio y móviles nativas manteniendo la arquitectura estándar para aplicaciones web.

d. Se debe investigar el uso de base de datos NoSql para ser adaptados al framework, ya que en la actualidad existe un creciente uso de base de datos NoSql mayormente para aplicaciones desarrollados con JavaScript-NodeJS.

BIBLIOGRAFÍA

1. Alegsa, L (2010). *Diccionario de Informática y Tecnología.* Recuperado de: http://www.alegsa.com.ar
2. Ambler, S. (2003). *Agile Database Techniques: Effective Strategies for the Agile Software Developer* [Técnicas de Base de Datos Ágiles: Estrategias Eficaces para el Desarrollador de Software Ágil]. Canadá, Estados Unidos: Wiley Publishing.
3. Antonio, J. (2001). *El gran libro del protocolo*. Madrid, España: Autor.
4. Bahit E. (2014*). El paradigma de la Programación Orientada a Objetos en PHP y el patrón de arquitectura de Software MVC*. Argentina: Creative Commons.
5. Bavaresco, A. (2006). *Proceso Metodológico en la Investigación. (Cómo hacer un diseño de investigación)*. Maracaibo: La Universidad del Zulia.
6. Beneken et al. (2003). *See several advantages of using components. USA: Object Management Group,* CORBA Componentes.
7. Cafassi, E. (1998). *Internet: Políticas y comunicaciones.* Buenos Aires, Argentina: Biblos.
8. Colección Esencial (2011). *Esencial Internet Explorer 9.* Cataluña, España: Editions ENI.
9. Councill y Heineman (2001). *Overview of COM+*. *In Component-Based Software Engineering*. Salzburg, Austria: Addison-Wesley Professional.
10. Craing, I. (2002). *The Interpretation of object-oriented programming languages. (2 ª ed.).* Gran Bretaña: Springer-Verlag London.
11. Cuauhtemoc S. (2014*). Creación de Frameworks con Patrones de Diseño para el Desarrollo de Aplicaciones Empresariales* (Tesis de Título). Universidad Nacional Autónoma de México, México D.F., México.
12. Gamma E., Helm R., Johnson R. y Vlissides J. (2003*). Patrones de Diseño, Elementos de software orientado a objetos reutilizable.* Madrid: Pearson Educación.
13. Gutiérrez, J. y Tena, J. (2003). *Protocolos criptográficos y seguridad en redes*. Cantabria, España: Gráficas Calima.
14. Gutiérrez A. y Vargas R (1999). *La Lógica de la Programación como Habilidad para Diseñar Algoritmos*. Instituto Politécnico Nacional, México D.F., México

15. Hernández, M. (2000). *Metodología de la investigación – Tipos y Niveles de Investigación.* Maracaibo, Venezuela: Departamento de investigación.
16. Kroenke, D. (2003). *Procesamiento de base de datos: fundamentos, diseño e implementación*. (8ª ed.). Juárez, México: Pearson.
17. Luján, S. (2001). *Programación en internet: Clientes web*. Alicante, España: Club Universitario.
18. Montaldo, D. (2005). *Patrones de diseño de Arquitecturas de Software Enterprise* (Tesis de grado). Universidad de Buenos Aires, Argentina.
19. Münch, J. (2010). *New Modelling Concepts for Today´s Software Prosses*. Paderbom: Proceedings
20. Nevado, V. (2010). *Introducción a las bases de datos relacionales*. Madrid, España: Vision Libros.
21. Osorio, F. (2008). *Base de datos relacionales: Teoría y práctica. (1ª ed.).* Madrid, España: Thomson.
22. Palacio, J. (2008) *Flexibilidad con Scrum*. Recuperado de http://www.safecreative.org/work/ 0710210187520
23. Regalut (2012). *Desarrollo móvil multiplataforma.* Recuperado de: http://desarrollomovilmultiplataforma.blogspot.pe/2012/08/aspectos-teoricos-libreria-biblioteca.html
24. Romero, L. (1997). *Publicar en Internet: guía práctica para la creación de documentos HTML*. Cantabria, España: Universidad de Cantabria.
25. Rosenberg, D. y Stephens, M. (2007). *Use Case Driven Object Modeling with UML: Theory and Practice*. Washington, Estados Unidos de América: Apress.
26. Rosenberg, D., Stephens, M., Collins-Cope, M. (2005). *Agile development with ICONIX Process: People, process, and pragmatism* [El Desarrollo Ágil con el Proceso ICONIX: Personas, procesos y pragmatismo]. New York, USA: Apress.
27. Rumbaugh, J., Jacobson, I. y Booch, G. (2007*). El leguaje unificado de modelado, manual de referencia*. Madrid: Pearson Educación.
28. Sanchez M. (2006*). Sistema de administración y control de renta de películas y libros vía web utilizando Spring* (Tesis de Título). Universidad de las Américas. Puebla, México.
29. Sommerville I. (2005). *Ingeniería de software. (2a ed.).* España: PEARSON ADDISON WESLEY.

30. Stair, R. y Reynolds, G. (1999). *Principios de Sistemas de Información: Enfoque administrativo. (4ª ed.).* Madrid, España: Thomson.
31. Szyperski, C. (2002). *Component Software: Beyond Object-oriented Programming. Boston*, USA: Addison-Wesley Longman Publishing.
32. Tamayo, M. y Tamayo, A. (1997*). El Proceso de la Investigación Científica. (Tercera Edición).* México: LIMUSA.
33. Varela A. (2015). *Adopción de métodos, técnicas y Herramientas para la innovación: Framework en función de casos reales*. (Tesis de título). Universidad Politécnica de Catalunya. Barcelona, España.
34. Vásquez E. (2007). *Una alternativa didáctica en la enseñanza de la lógica para estudiantes de primero de bachillerato en computación.* (Tesis de título). Universidad Nacional Francisco Morazán, San Pedro Sula, Honduras.
35. Vértice (2010). *Técnicas avanzadas de diseño web.* España: Vértice.
36. Virvou, M. y Matsuura, S. (2012). *Knowledge-based software engineering: Proceedings of the Tenth Joint Conference on Knowledge-based software engineering* [La Ingeniería de Software Basada en el Conocimiento: Actas de la Décima Conferencia Conjunta sobre la Ingeniería de Software Basada en el Conocimiento]. Amsterdam, Netherlands: IOS Press BV.
37. Weitzenfeld, A. (s.f.). *Ingeniería de software orientada a objetos: con UML, Java e internet.* Madrid, España: Thomson.

ANEXO A: OPERACIONALIZACIÓN DE VARIABLES

VARIABLE DE INTERES	DIMENSIONES	INDICADORES	PREGUNTA	INSTRUMENTO
Framework	Biblioteca de Modelo	Métodos de abstracción	¿Cuáles son los métodos utilizados para una abstracción objetiva en el modelado de una entidad?	Ficha de análisis documental
			¿Cómo los métodos de abstracción estandarizada ayudan a un patrón arquitectónico que permite el acceso a una base de datos genérica?	Ficha de análisis documental
		Procesamiento de consultas y sentencias SQL	¿Cómo gestiona un sistema de base de datos el procesamiento de consultas SQL?	Ficha de análisis documental
			¿Cómo gestiona un sistema de base de datos el procesamiento de sentencias SQL?	Ficha de análisis documental
		Seguridad de acceso a datos	¿Cómo garantizar la conexión segura a la base de datos?	Ficha de análisis documental
			¿Cómo evitar las inyecciones SQL no deseadas a la base de datos?	Ficha de análisis documental
	Biblioteca de Vista	Estandarización de envío de datos	¿Cómo se envía datos a la vista de una aplicación web?	Ficha de análisis documental
			¿De qué manera se recibe los datos en la vista de una aplicación web?	Ficha de análisis documental
		Renderización de vistas	¿De qué manera utilizar plantillas html para mostrarlas en la vista de una aplicación web?	Ficha de análisis documental
		Reutilización de	¿Cómo reutilizar archivos estáticos, html, css, javascript, etc. en la vista de	Ficha de análisis

		recursos estáticos.	una aplicación web?	documental
	Biblioteca de Controlador	Patrón de diseño	¿Cuáles son los patrones de diseño que permiten separar la vista de la lógica en una aplicación web?	Ficha de análisis documental
			¿El patrón de diseño MVC permite abstraer las entidades, controlar la lógica y mostrar una vista en una aplicación web?	Ficha de análisis documental
		Arquitectura técnica	¿De qué manera estandarizar una estructura de ficheros, compatible con diversos lenguajes de programación?	Ficha de análisis documental
			¿Qué nombres utilizar para las clases y métodos reutilizables del framework?	Ficha de análisis documental
		Enrutamiento de peticiones (request)	¿Cómo enrutar una petición (request) para ejecutar una acción en un determinado controlador?	Ficha de análisis documental
		Control de vistas y plantillas	¿De qué manera poder seleccionar desde el controlador las vistas a mostrar en la vista?	Ficha de análisis documental
			¿De qué manera poder seleccionar desde el controlador las plantillas a mostrar en la vista?	Ficha de análisis documental
		Funciones básicas de control	¿Qué funciones son necesarias para implementar aplicaciones web desde un framework?	Ficha de análisis documental

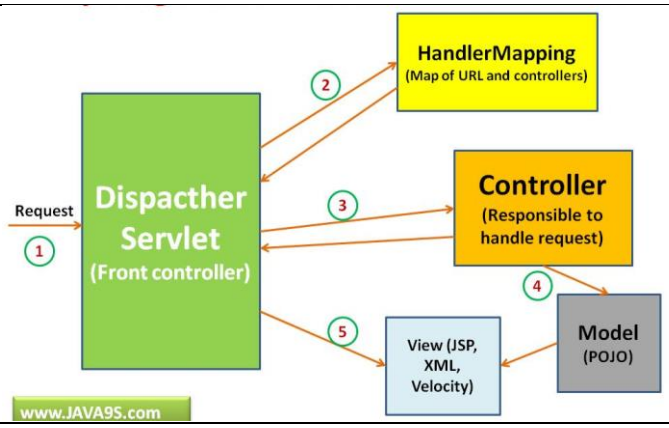
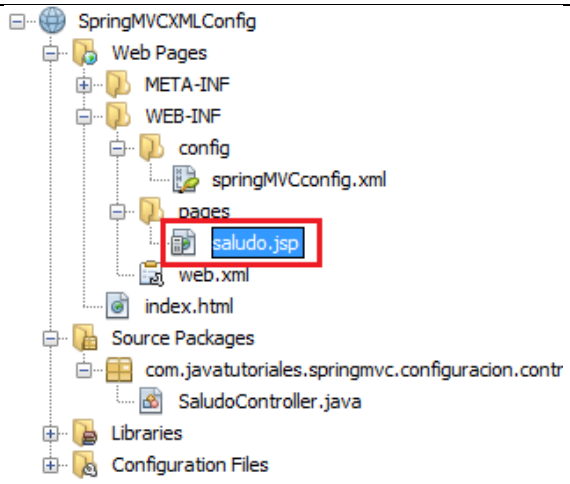
Tabla A.1. Operacionalización de variables.

ANEXO B: FICHA DE ANALISIS DOCUMENTAL

Fecha		**Nombre (opcional)**				
Pregunta:						
Respuesta:						
Opción (valoración)	**Item 1**	**Item 2**	**Item 3**	**Item 4**	**Item 5**	**Item N**
A						
B						
C						
D						
Z						

Tabla B.1 Ficha de Análisis de Documental para respuesta a preguntas operacionales

ANEXO C: FICHA ADICIONAL DE ANALISIS DE FRAMEWORKS EXISTENTES

<table>
<tr><th colspan="8">FICHA DE ANÁLISIS DOCUMENTAL: ESTUDIO DE FRAMEWORK</th></tr>
<tr><td colspan="2">Nombre del Framework</td><td colspan="3">Spring Framework</td><td>Lenguaje</td><td colspan="2">Java</td></tr>
<tr><td>Versión</td><td>5.0.0</td><td>Fecha</td><td>28-09-2017</td><td>Licencia</td><td colspan="3">Apache License 2.0</td></tr>
<tr><td colspan="2">Sistema operativo</td><td colspan="2">Multiplataforma</td><td colspan="2">Plataforma</td><td colspan="2">Máquina virtual Java</td></tr>
<tr><td colspan="2">Desarrolladores</td><td colspan="2">Pivotal Software</td><td>Autor</td><td>Rod Johnson</td><td>Lanzamiento</td><td>2002</td></tr>
<tr><td colspan="8">ARQUITECTURA DEL FRAMEWORK</td></tr>
<tr><td colspan="8"> 
</td></tr>
<tr><td colspan="8">ESTRUCTURA DE FICHEROS</td></tr>
<tr><td colspan="8"> 
</td></tr>
<tr><td colspan="2">Extensión de clases</td><td colspan="2">.java</td><td colspan="2">Extensión de vistas</td><td colspan="2">.jsp</td></tr>
<tr><td colspan="8">ESTRUCTURA DE UN CONTROLADOR</td></tr>
<tr><td colspan="8">

```
@Controller
public class SaludoController {
	@RequestMapping
	public String saluda() {
		return "saludo";
	}
}
```

</td></tr>
</table>

Tabla C.1 Ficha adicional de análisis del framework Spring.