ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN (SISTEMAS E INFORMÁTICA)

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS E INFORMÁTICA

TEMA: ANÁLISIS COMPARATIVO DE HERRAMIENTAS ORIENTADAS A COMPONENTES WEB VALIDADO CON UN CASO DE ESTUDIO.

AUTOR: MORENO OJEDA, MARCO ANTONIO

DIRECTOR: PHD. MARCILLO PARRA, DIEGO MIGUEL

SANGOLQUÍ

2017

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

CERTIFICACIÓN

Certifico que el trabajo de titulación, **"*Análisis comparativo de herramientas orientadas a componentes web validado con un caso de estudio*"** realizado por el señor ***MARCO ANTONIO MORENO OJEDA***, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar al señor ***MARCO ANTONIO MORENO OJEDA*** para que lo sustente públicamente.

Sangolquí, 30 de agosto del 2017

Atentamente,

Ing. Diego Marcillo PhD

DIRECTOR

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

AUTORÍA DE RESPONSABILIDAD

Yo, **Marco Antonio Moreno Ojeda**, con cédula de identidad No. 1719593244, declaro que este trabajo de titulación "**Análisis comparativo de herramientas orientadas a componentes web validado con un caso de estudio**" ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas. Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

Salgolquí, 30 de agosto del 2017

Marco Antonio Moreno Ojeda
C.C. 1719593244

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

AUTORIZACIÓN

Yo, **Marco Antonio Moreno Ojeda**, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca virtual de la institución el presente trabajo de titulación "**Análisis comparativo de herramientas orientadas a componentes web validado con un caso de estudio**" cuyo contenido, ideas y criterio son de mi autoría y responsabilidad.

Salgolquí, 30 de agosto del 2017

Marco Antonio Moreno Ojeda
C.C. 1719593244

DEDICATORIA

A mis padres Marco y Martha y hermanas Liz y Alex por estar continuamente apoyándome en cada una de las actividades que he realizado a lo largo de mi vida, por los consejos que me han brindado durante todo este periodo de formación de vida sosteniéndome siempre ante cualquier adversidad, por ser fuente de inspiración para continuar y siempre recordarme lo importante que es seguir de frente sin importar los obstáculos que se presentan y mostrándome que lo esencial no es esquivarlos sino superarlos.

A mis abuelos que al igual que mis padres me dieron la oportunidad de continuar con mis estudios y que siempre me han brindado su total y absoluta confianza y apoyo para cumplir cada una de mis metas y cuando he requerido su abrazo y su consuelo estuvieron presentes.

Marco Moreno

AGRADECIMIENTO

Agradezco especial e infinitamente a quien invirtió su tiempo durante mi formación como estudiante y en este proceso de titulación y quien a pesar de algunas circunstancias me brindó todo su apoyo y colaboración para conseguir esta nueva meta, mostrando ser una persona honorable y respetable, mi director de proyecto Ing. Diego Marcillo.

También agradezco a mis amigos con quienes estuvimos cursando todo este proceso académico y a aquellos que han sido amigos de siempre por el hecho de mostrar su cariño y afecto al no permitirme caer y más bien han extendido su mano en muchas ocasiones para seguir adelante.

Marco Moreno

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

Las tecnologías de desarrollo de software han evolucionado progresivamente lo cual ha permitido que algunos lenguajes como JavaScript tomen fuerza, particularmente en el desarrollo web. Debido a esto han surgido herramientas con orientación a componentes web con la finalidad de agilizar la programación de aplicaciones para dicho ambiente. Ante este antecedente se ha realizado un análisis comparativo que permita establecer que herramienta entre las más populares como Angular 2 y React JS es la más adecuada para realizar el proceso de desarrollo. Sin embargo, para el análisis se ha considerado normas de calidad para establecer un punto de inicio y que provea credibilidad y calidad en el estudio. La norma propuesta ha sido el estándar ISO/IEC 9126 el cual se ha integrado con metodologías que permitan construir un modelo adecuado de evaluación como es el caso de IQMC pasando por un previo estudio comparativo con similares para ser elegido. Durante el procedimiento de comparación se han identificado características y las mismas se han derivado en atributos más pequeños y medibles con la finalidad de obtener resultados eficientes y correctos. Al seleccionar la herramienta se realiza una aplicación y a continuación se validan los resultados obtenidos.

PALABRAS CLAVE:

- **ANGULAR 2**
- **ISO/IEC 9126**
- **REACT JS**
- **IQMC**

ABSTRACT

Software development technologies have evolved progressively which has allowed some languages like JavaScript to take force, particularly in web development. Because of this, tools have been developed with orientation to web components in order to expedite the programming of applications for that environment. Before this antecedent has been made a comparative analysis that allows to establish that tool among the most popular as Angular 2 and React JS is the most adequate to carry out the development process. However, for the analysis, quality standards have been considered to establish a starting point and provide credibility and quality in the study. The proposed standard has been the standard ISO/IEC 9126 which has been integrated with methodologies that allow to construct an adequate model of evaluation as is the case of IQMC going through a previous comparative study with similar ones to be chosen. During the comparison procedure characteristics have been identified and they have been derived in smaller and measurable attributes in order to obtain efficient and correct results. When selecting the tool an application is made and then the results obtained are validated.

KEYWORDS:

- **ANGULAR 2**
- **ISO/IEC 9126**
- **REACT JS**
- **IQMC**

CAPÍTULO I

INTRODUCCIÓN

1.1. Antecedentes

En la actualidad existen muchas herramientas de desarrollo que permiten disminuir tiempos de trabajo de un proyecto, brindan facilidades de implementación o facilitan incluso la programación por mantener una estructura óptima. Estos Frameworks y librerías se encuentran desarrollados en diferentes lenguajes de programación como PHP, Java, JavaScript, Python, entre otros. Sin embargo, muchos de ellos tienen una estructura preestablecida y también están enfocados para un solo ambiente y al querer realizar una migración de un ambiente a otro se vuelve complejo. Cabe mencionar que existen herramientas que están orientadas para desarrollar proyectos pequeños y otras que son más escalables, óptimos y tienen un enfoque para aplicaciones de índole empresarial.

Realizar proyectos con ayuda de un Framework puede deberse a varios motivos como puede ser que el Framework se adapta a una arquitectura establecida, brinda formas para mantener una codificación limpia, minimizan tiempo, entre otros; sin embargo, existen algunas herramientas web que han nacido con la finalidad de aportar algo más como por ejemplo la reutilización parcial o completa del código para diferentes ambientes, es decir que al obtener la aplicación web esta misma sirva para construir gran parte de una aplicación móvil y de esta manera minimizar tiempo de desarrollo lo cual significa una gran ventaja a nivel de costo y beneficio. Estas herramientas constituyen algunos Frameworks y/o librerías basados en componentes web como lo son Angular 2 y ReactJS.

En el mundo de desarrollo Web existe un debate de qué herramientas representan mayor aporte para un proyecto es por esta razón que se ve necesario realizar un estudio comparativo entre los principales Frameworks y librerías como los mencionados en el párrafo anterior, con la finalidad de brindar un mayor conocimiento en cuanto a las características y ventajas que

cada uno de estos puede ofrecer. Es de gran importancia determinar todos los aspectos más sobresalientes de cada una de las herramientas en cuestión ya que para un futuro se puede sustentar de manera adecuada la elección de un Framework u otra herramienta para desarrollar una aplicación web, móvil o incluso una aplicación híbrida.

1.2. Planteamiento del problema

Actualmente existen una serie de herramientas de desarrollo web como librerías, frameworks, plugins que tienen la finalidad de optimizar tiempo, mejorar la calidad, entre otras cosas. En esta evolución se incluye la aparición periódica de herramientas con diferentes enfoques, entre las que se encuentran aquellas orientadas a web components. La tendencia del desarrollo Front – End en la plataforma web en la actualidad, es el manejo de componentes (Azaustre, Web Components: presente y futuro en el desarrollo web, 2016). La dificultad de seleccionar herramientas orientadas a web components para el desarrollo de aplicaciones ha sido uno de los principales problemas que enfrentan los desarrolladores y arquitectos de soluciones. Algunas de las causas de este problema se detallan más adelante.

El tiempo es una de las principales causas del problema presentado. Para los desarrolladores es crucial el elegir la herramienta que mejor se adapte a sus necesidades, sin embargo, aprender la tecnología ya sea nueva o alguna actualización significativa conlleva dedicar tiempo (Mariano, 2017), por lo tanto la curva de aprendizaje es un factor clave en este problema, ya que muchos de los desarrolladores muestran resistencia a la hora de cambiar de herramientas por esta situación.

La falta de documentación basada en normativa es otra de las causas del problema, esto se debe a que con la gran cantidad de herramientas disponibles es complejo realizar investigaciones que las analicen en términos de desempeño, lo cual sería de gran ayuda en la selección. La calidad de una aplicación se complica al elegir la herramienta inadecuada (VECTOR ITC GROUP, 2015). Sí existen comparaciones entre herramientas, pero que no se fundamentan en alguna norma, buenas prácticas o documento técnico

estandarizado y esto provoca desconfianza en los interesados y mayor tiempo en el proceso de selección.

Otro de los motivos del problema mencionado previamente es la evolución constante de las herramientas. Este progreso tecnológico se ha convertido en un dolor de cabeza, ya que para elegir la herramienta, el factor decisivo más común se centra típicamente en la familiaridad del desarrollador con la misma, lo cual no es una base correcta, ya que tiende a ser subjetivo (Christodolou & Gizas, 2012). Es por esto que los desarrolladores tienden a dejar de lado factores importantes como el rendimiento, la escalabilidad y mejoras que ofrecen las diversas herramientas.

En este proyecto se realiza un estudio comparativo de herramientas orientadas a web components basándose en un estándar que brinde la mayor cantidad de información y que sea el soporte correcto para la selección de herramientas.

1.3. Justificación

En el desarrollo de un proyecto es importante tener fundamentos técnicos basados en estándares al elegir la tecnología con la que se trabaja, tomando en consideración aspectos como la escalabilidad, el rendimiento, la eficiencia, el consumo, entre otras características que permiten obtener un producto software de calidad.

La investigación se centra en herramientas orientadas a Web Components ya que esta es la nueva tendencia en el desarrollo de software a nivel de Front-End y los estudios realizados con este enfoque son escasos o utilizan características muy generales.

Se propone el uso de una norma estandarizada ISO/IEC que permita evaluar correctamente las herramientas, proporcione los parámetros necesarios para extraer la información más relevante y permita elegir la más apropiada. El propósito es brindar material base bien fundamentado para estudios similares o equivalentes al propuesto.

Otra razón para realizar la investigación y el análisis correspondiente de las herramientas es mostrar la importancia de selección con normativa, ya que

así se puede medir aspectos como por ejemplo la reutilización, que puede servir para determinar si la herramienta es apta para usarla en varios ambientes y a su vez si permite tener escalabilidad en un proyecto. De esta manera se ayuda a tomar decisiones reflexionando que la elección de una herramienta es crucial para el desarrollo de un proyecto.

Finalmente, para comprobar la validez de la herramienta orientada a componentes web seleccionada se realiza una aplicación que permita gestionar la disponibilidad de estacionamientos en una zona residencial, como un conjunto residencial o sus alrededores.

1.4. Objetivos

1.4.1. Objetivo general

Comparar herramientas orientadas a Web Components mediante el uso de una norma ISO/IEC 9126, para el desarrollo de aplicaciones, con un caso práctico.

1.4.2. Objetivos específicos

- Definir un modelo de evaluación de herramientas utilizando criterios tomados de la norma ISO/IEC 9126.
- Aplicar el modelo a las herramientas en estudio y seleccionar la mejor opción para la realización de un caso práctico.
- Desarrollar una aplicación utilizando la herramienta seleccionada para la gestión de disponibilidad de parqueaderos como caso de estudio.

1.5. Alcance

Para realizar la investigación mencionada es necesario en primer lugar revisar la normativa correspondiente que permita considerar los criterios necesarios para una adecuada evaluación y selección de un modelo correcto, en este caso se utilizará la Norma ISO/IEC 9126 de la cual se toma únicamente lo conveniente para obtener criterios específicos para las herramientas web pertinentes.

Otro de los aspectos importantes es conocer sobre las herramientas que son analizadas con el debido procedimiento, estas herramientas están basadas en la misma tecnología para este caso en “Web Components” y para ello se ha considerado los siguientes, Angular 2 y React JS.

Se realiza el análisis comparativo entre las herramientas mencionadas anteriormente y se desarrolla una aplicación con la tecnología elegida a través de la evaluación, esta aplicación es un Administrador de disponibilidad de estacionamientos para zonas residenciales, la misma que contendrá:

- Administración de Usuarios: Un administrador podrá crear, editar, dar de baja usuarios.
- Gestión de Estacionamientos: Un administrador podrá crear, editar, dar de baja un estacionamiento.
- Registro de Entrada y Salida: El usuario o administrador podrá ingresar a la aplicación y podrá registrar el parqueadero en el cual se ha estacionado el vehículo y la salida del mismo.
- Gestión de Vehículo: Se registra las características como marca, modelo, color, placa y su respectivo CRUD.
- Gestión de Propietario de Vehículo: Se debe registrar datos del propietario como nombre, apellido, cédula de identidad.
- Reportes como disponibilidad de estacionamiento, históricos de entrada y salida de vehículos, entre otros.
- Deberá constar de hora de llegada y hora de salida del vehículo.

CAPÍTULO II

MARCO TEÓRICO

2.1. Web Components

Los componentes web son un conjunto de API's de la plataforma web que permiten crear y utilizar elementos HTML personalizados, reutilizables y encapsulados, los mismos que se pueden usar en aplicaciones y sitios web. Cada web component tiene semántica, funcionalidad y lógica en cuanto a su presentación.

Esta especificación para el desarrollo web, se describe en algunos estándares web, a través de los cuales los desarrolladores consiguen extender a nuevos elementos HTML encapsulados y personalizados.

2.1.1. Estándares que utilizan web components

Como se mencionó anteriormente web components se basan en estándares pertenecientes a la plataforma web, estos son cuatro y se explican a continuación:

- **Custom Elements**

  Provee las maneras para construir elementos DOM propios de forma sencilla. Con esto, es posible crear elementos personalizados de manera inmediata y en colaboración de otras personas (Morales, 2014). Por ejemplo en el caso de reproductores basados en HTML5 o el mapa como se muestra (ver Figura 1).

```
<google-map latitude="37.77493" longitude="-122.41942"></google-map>
```

Figura 1 Ejemplo de etiqueta personalizada de mapa (Morales, 2014).

- **Shadow DOM**

  Brinda la forma de encapsular el contenido del componente con la finalidad de aislarlo del DOM original y renderizarlo. Esto quiere decir

que mantiene su estilo CSS y se ejecuta en otro ámbito diferente al DOM original del documento o página web, como se indica en la figura (ver Figura 2)

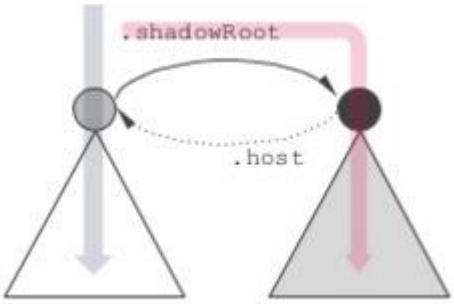

Figura 2 Shadow DOM (Vélez, 2015).

- **HTML Imports**

  Ofrece la posibilidad de importar archivos HTML a otros ficheros. Esto se logra por el modelo modular y empaquetado, tal como se expresa en la siguiente figura (ver Figura 3).

```
<head>
  <link rel="import" href="/path/to/imports/stuff.html">
  <!- Documentos en otro dominio necesitan CORS-enabled. -->
  <link rel="import" href="http://example.com/elements.html">
</head>
```

Figura 3 Importar un documento HTML (Paradigma Digital, 2014)

- **HTML Templates**

  Son pequeños pedazos de código HTML inertes que pueden ser utilizados cuando sea necesario. Se activan solamente cuando el componente es llamado para su respectiva renderización. La representación de lo mencionado se visualiza en la figura que continúa (ver Figura 4)

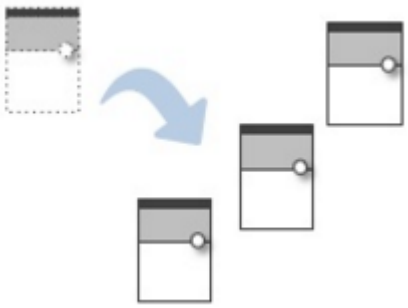

Figura 4 Plantillas de HTML (Vélez, 2015)

2.1.2. Compatibilidad con navegadores

La compatibilidad de los navegadores con componentes web se determina según el soporte que cada uno de ellos brinda a los estándares Custom Elements, Shadow DOM, HTML Imports, HTML Templates mencionados anteriormente. En caso de no ser compatibles, gracias a Polyfills se puede lograr esto ya que proporciona una porción de código que implementa características que los navegadores no soportan por sí mismos. Según la imagen (ver Figura 5), en la actualidad la compatibilidad es la siguiente:

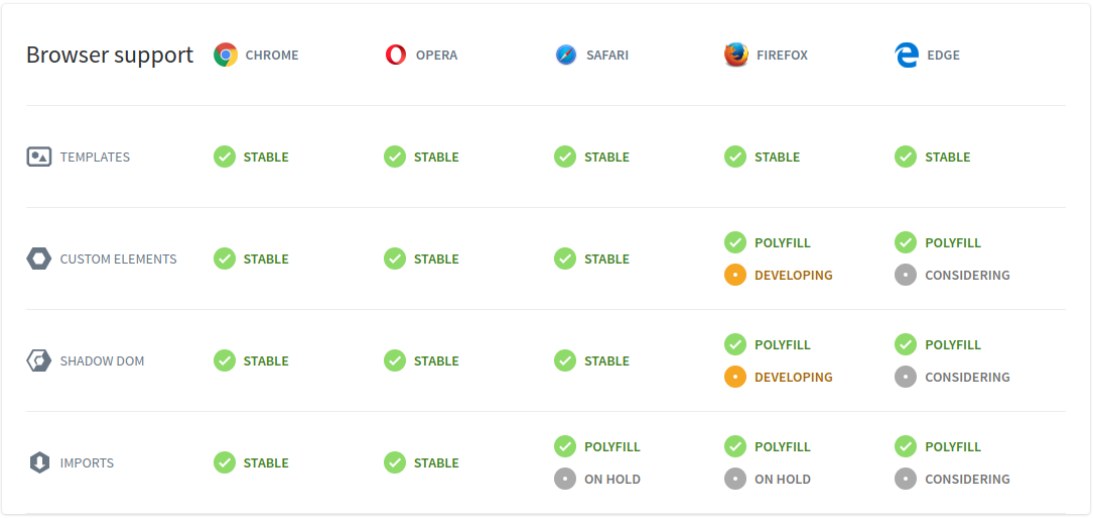

Figura 5 Compatibilidad de web components con browsers (Kuenzi, 2017)

Como se observa Chrome y Opera ya son los navegadores con mayor compatibilidad, mientras que Safari, Firefox, Edge todavía requieren de la

ayuda de Polyfill para relacionarse con esta nueva tecnología de desarrollo web.

2.1.3. Herramientas orientadas a web componentes

Dado que hoy en día se ha popularizado la programación con orientación a componentes web, se han desarrollado algunas herramientas como frameworks y librerías diseñadas para soportar este nuevo enfoque. Entre algunas de las herramientas que se proporcionan con este nuevo esquema tenemos a Polymer, Vue JS, Angular 2 y React JS. Siendo estos dos últimos muy utilizados por la comunidad debido a su gran capacidad de soporte e integración para el desarrollo de aplicaciones móviles de carácter híbrido.

2.2. Angular 2

2.2.1. Introducción

Angular 2 es el sucesor de AngularJS, ambos desarrollados por Google. No se trata de una actualización de Angular JS, ya que fue escrito desde cero, sin embargo, se ha tomado como referencia algunos conceptos como directivas, templates, data binding que se utilizaban en su predecesor. Este framework fue diseñado para construir aplicaciones cliente en HTML y JavaScript o lenguajes que compilan JavaScript como TypeSript, y también para renderizar desde el navegador y no desde el servidor. Además, que permite a los desarrolladores crear aplicaciones móviles bajo la integración de otras plataformas.

Con esta herramienta se puede escribir aplicaciones a través de plantillas HTML las mismas que se pueden manejar utilizando componentes, adicionar la lógica de la aplicación en servicios y tanto componentes como servicios encapsular en módulos.

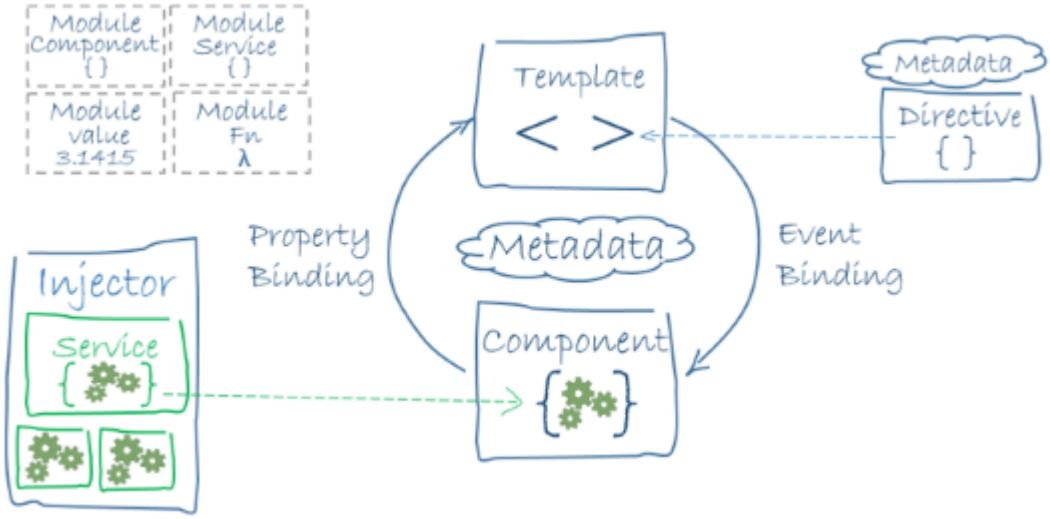

Figura 6 Bloques principales de Angular 2 (Google, 2017).

A más de los elementos mencionados anteriormente plantillas, componentes, servicios y módulos, se aprecian metadata, data binding, inyección de dependencias y directivas. Todos estos conforman los ocho bloques principales que contiene una aplicación realizada con esta herramienta. A continuación, se realiza una breve explicación de cada uno de estos bloques.

2.2.2. Módulos, Componentes, Templates y Metadatos

Módulos

Un módulo es un conjunto de código con la finalidad de proveer funcionalidades, flujos o capacidades específicas dentro de la aplicación. En Angular existe un módulo principal, por lo cual todas las aplicaciones desarrolladas con la herramienta mencionada mantendrán al menos un módulo.

Angular posee su propio sistema de módulos conocidos como Angular Modules o NgModules, estos facilitan la inyección de dependencias que requiera cada aplicación y se maneja por decoradores, en este caso @NgModule. Las propiedades más importantes son:

- Declarations. - Esta propiedad acepta un arreglo de las clases de vistas que corresponden al módulo.

- Exports. - Un conjunto de declaraciones a las cuales se tiene acceso desde otro módulo.
- Imports. - Todos los módulos que contienen funcionalidades necesarias para el módulo actual.
- Providers. - Es el conjunto de servicios que provee el módulo en la aplicación completa.
- Bootstrap. - Se utiliza únicamente en el módulo principal o raíz, en éste se define la vista principal.

```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
@NgModule({
  imports:      [ BrowserModule ],
  providers:    [ Logger ],
  declarations: [ AppComponent ],
  exports:      [ AppComponent ],
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

Figura 7 Ejemplo de Módulo (Google, 2017)

El empaquetamiento de Angular es como una colección de módulos de librerías JavaScript, las mismas que se distinguen por mantener un prefijo @angular. Entre estas se encuentran @angular/core, @angular/platform-browser, @angular/http, @angular/router, @angular/forms. Siendo las más frecuentes y de mayor uso.

Componentes

Un componente es el que controla un espacio de la pantalla, es decir la vista. Para crear un componente se define en una clase la lógica y funcionalidad que contiene, luego esta se encarga de interactuar con la vista por medio de métodos y propiedades (CloudExperto, 2017).

Es importante aclarar que un componente se crea y se destruye según la navegación en la aplicación y una de las razones es por el árbol de

componentes que se va formando durante el desarrollo de la aplicación ya que siempre existe un componente padre del cual se pueden obtener ramificaciones.

Templates

La vista de un componente se define por medio de una plantilla la misma que se basa en HTML y a su vez indica a la herramienta como debe mostrarse el componente, es decir como renderizarse. En esta plantilla a más de HTML se puede utilizar otros aspectos enmarcados en este framework de desarrollo como son directivas, data binding y componentes personalizados. Se puede apreciar en la figura (ver Figura 8).

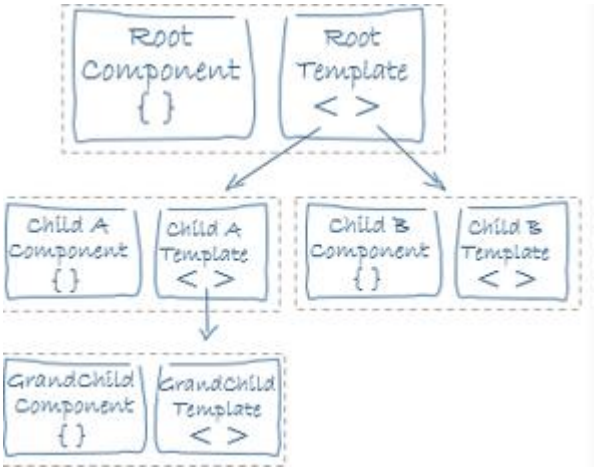

Figura 8 Composición de Template o Plantilla (Google, 2017)

Metadatos

Los metadatos proveen información a Angular de como procesar una clase y en conjunto con la plantilla y el componente describe la vista. Ante esto un componente no existe hasta que se lo bautice como tal, y los metadatos permiten realizar esta acción a través de la declaración de decoradores como @Component, @Injectable, @Input, @Output.

```
@Component({
  selector:    'hero-list',
  templateUrl: './hero-list.component.html',
  providers:  [ HeroService ]
})
export class HeroListComponent implements OnInit {
/* . . . */
}
```

Figura 9 Ejemplo de decorador Component (Google, 2017)

Como se observa (ver Figura 9) se envía parámetros para indicar a Angular que la clase será un componente y los principales son:

- Selector: Solicita a Angular que instancie el componente cada vez que encuentra un elemento con ese nombre respectivo.
- TemplateUrl: Es la dirección de la plantilla HTML relacionada con el componente.
- Providers: Si un componente necesita ejecutar algunos servicios, se identifica el conjunto de dependencias que contienen dichos servicios.
- StyleUrls: Un arreglo en el cual se pueden declarar todos los estilos que el componente requiere. Esto permite cumplir una de las características de web components como es el Shadow DOM.

2.2.3. Data binding, directivas, servicios e inyección de dependencias

Data Binding

El enlace de datos o data binding permite comunicar las partes de una plantilla con las partes de un componente y también los componentes padres con los hijos si es el caso. Angular permite vincular los datos de cuatro maneras que se indican a continuación (ver Figura 10) y se detallan en tabla que sigue (ver Tabla 1).

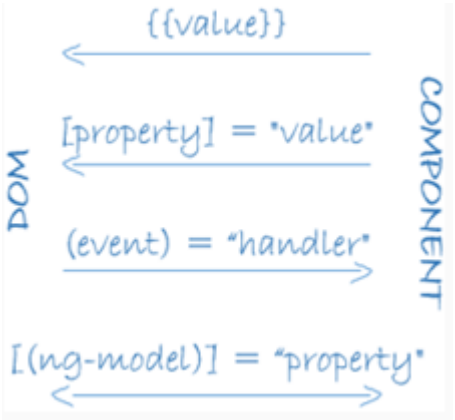

Figura 10 Representación sintaxis data binding (Google, 2017).

Tabla 1

Sintaxis de data binding

Nombre	Representación	Dirección
Interpolación	{{ valor }}	Unidireccional hacia el DOM
Property binding	[propiedad] = "valor"	Unidireccional hacia el DOM
Event binding	(event) = "handler"	Unidireccional desde el DOM
Two-way binding	[(ng-model)] = "propiedad"	Bidireccional desde y hacia el DOM

Fuente: (Google, 2017)

Directivas

Las directivas se representan con el decorador @Directive. Existen dos tipos de directivas las estructurales que son aquellas que permiten aumentar, quitar o reemplazar elementos en el DOM y las directivas de atributos las cuales alteran el comportamiento de un elemento (CloudExperto, 2017).

- Directivas estructurales: *ngFor, *ngIf, ngSwitch
- Directivas de atributo: [(ngModel)], ngStyle, ngClass

Las directivas mencionadas permiten realizar el control de los elementos, por lo tanto, pueden ser útiles para realizar el respectivo control de las funcionalidades de un elemento en un determinado momento.

Servicios

Los servicios son clases que contienen las funcionalidades que requiere la aplicación, básicamente la lógica del negocio desde el lado del cliente. Se encuentran definidos en muchas partes de la aplicación y puede ser cualquier función, estos son consumidos por los componentes. Estos servicios tienen como pieza clave al decorador @Injectable, el cual indica a Angular que se utilizarán otros servicios y la inyección de dependencias que básicamente es pasar parámetros en el constructor.

Como ya se mencionó un servicio puede esperar otro servicio y este puede ser un web service implementado. Para esto se utiliza el módulo HttpModule obtenido de @angular/http el cual provee los mecanismos adecuados basados en AJAX para realizar la petición correspondiente.

```
export class HeroService {
  private heroes: Hero[] = [];

  constructor(
    private backend: BackendService,
    private logger: Logger) { }

  getHeroes() {
    this.backend.getAll(Hero).then( (heroes: Hero[]) => {
      this.logger.log(`Fetched ${heroes.length} heroes.`);
      this.heroes.push(...heroes); // fill cache
    });
    return this.heroes;
  }
}
```

Figura 11 Ejemplo de servicio (Google, 2017)

Inyección de dependencias

Es una forma para proveer nuevas instancias de una clase con todas las dependencias que necesite. Generalmente estas dependencias suelen ser

servicios. Y son parámetros del constructor de la clase o del componente que lo requiere, de esta manera Angular solicita la inyección respectiva y reconoce los servicios correspondientes del componente únicamente con observarlo, una ventaja que brinda TypeScript (Oriol, 2017).

2.2.4. Instalación y Angular – Cli

Para iniciar con Angular 2 es importante tener instalado Node.js, este entorno popular con su gestor de paquetes facilita la instalación, actualización, recuperación de módulos y librerías que han sufrido algún daño de Angular e incluso permite que Angular 2 sea compatible en cualquier sistema operativo. También permite realizar la instalación de la herramienta Angular – CLI que el propio equipo de Angular ha entregado con la finalidad de agilizar el desarrollo de las aplicaciones.

Con el uso de Angular – CLI, esta interfaz de línea de comandos, es posible detectar errores antes y después de la ejecución de la aplicación, realizar pruebas, crear proyectos nuevos, configurar el entorno de desarrollo, generar código nuevo, entre otras opciones. Todo esto se puede conseguir de manera rápida y segura ya que este sistema se basa en las buenas prácticas que se exponen en la página oficial de Angular (Eizagirre, 2016).

2.2.5. Comunidad y compatibilidad con navegadores web

Comunidad

La comunidad de Angular actualmente en junio de 2017 consta con un grupo de sesenta y seis miembros principales de los cuales veinte y nueve son Google Developers Experts y los treinta y siete restantes son colaboradores inmediatos que ayudan a que esta plataforma, como la denominan, siga en crecimiento y tenga el soporte y la documentación efectiva y de calidad a disposición de todos.

También cabe mencionar que la comunidad de GitHub sigue en crecimiento para la misma fecha se tienen un total de 25000 seguidores sin considerar el número de seguidores de su antecesor AngularJS el mismo que actualmente tiene 56000 seguidores aproximadamente.

Adicionalmente cabe rescatar que existen otras comunidades relacionadas con Angular 2 como son las de Angular Material que se dedica a ofrecer un conjunto de elementos de interfaz de usuario para su uso en aplicaciones de una sola página. Otra comunidad es la AngularFire dedicada al soporte para aquellos interesados en el desarrollo Full – Stack con Firebase. Existen otras comunidades o colaboradores que han creado diferentes componentes y los tienen disponibles en la web.

Ante el trabajo de la comunidad de Angular hoy en día se cuenta con algunas versiones, las mismas que han ido mejorando y agregando varios aspectos de esta herramienta, existen ciento catorce versiones a la fecha considerando desde su versión alfa en 2015.

Compatibilidad con navegadores web

Las actuales versiones de Angular soportan los navegadores más recientes indicadas en la siguiente imagen oficial (ver Figura 12).

Chrome	Firefox	Edge	IE	Safari	iOS	Android	IE Mobile
latest	latest	14	11	10	10	Nougat (7.0) Marshmallow (6.0)	11
		13	10	9	9	Lollipop (5.0, 5.1)	
			9	8	8	KitKat (4.4)	
				7	7	Jelly Bean (4.1, 4.2, 4.3)	

Figura 12 Soporte a navegadores

2.3. React JS

2.3.1. Introducción

React JS es una librería JavaScript de Front – End que permite construir interfaces de usuario. Se caracteriza por ser declarativa, eficiente y flexible. Fue desarrollada por Facebook y al igual que Angular está diseñada para la

creación de aplicaciones web de una sola página. Para el uso de esta herramienta es recomendable utilizar JSX que tiene embebido HTML por lo cual este último es posible usarlo. Esta popular herramienta está encargada de la vista, es decir se encarga solo de esta tarea y para el desarrollo de aplicaciones completas se deben integrar otras librerías.

Dado las bondades que posee esta herramienta y tras el rendimiento que brinda en las aplicaciones desarrolladas ha sido considerado también para ser utilizado en el proyecto del framework para aplicaciones móviles conocido como React Native sin cambiar su funcionalidad (Facebook Inc, 2017).

2.3.2. Componentes, flujo de datos, DOM virtual y representación condicional

Componentes

Los componentes en React permiten dividir la interfaz en diferentes partes las mismas que son independientes y reutilizables, con esto se logra un mantenimiento aislado de cada una de ellos. Un componente se puede escribir como una función de JavaScript, esta puede aceptar entradas conocidas como props y retornar el elemento que se presentará en pantalla. A continuación se muestra un ejemplo (ver Figura 13).

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

function App() {
  return (
    <div>
      <Welcome name="Sara" />
      <Welcome name="Cahal" />
      <Welcome name="Edite" />
    </div>
  );
}

ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

Figura 13 Ejemplo de componente con React (Facebook Inc, 2017)

Uno de los elementos más importes de un componente son los “props” (ver Figura 13). Estos son objetos que almacenan información que utiliza el componente. También como se observa en la figura un componente puede utilizar otros componentes. Además de props, parte de un componente es el estado o “state” el que permite la interactividad con el componente, ya que este indica si guarda o no datos en su respectiva memoria.

Flujo de datos

La dirección de los datos es importante ya que permite establecer como se enlazarán los datos entre el componente y el DOM. En el caso de React JS esta vinculación se produce de manera unidireccional, es decir que los datos se transportan a través del state y props al componente. Bajo este esquema los componentes padres envían los datos a los hijos.

DOM Virtual

Con la finalidad de disminuir el coste en rendimiento por la constante actualización que se produce en el DOM y para mantener rendimiento alto en cuanto al renderizado, React implementa en su estructura un Document Object Model virtual. Este DOM virtual es una especie de memoria, en la cual se crea una copia del DOM original y al producirse un cambio se hace una comparación con el DOM original y se renderiza únicamente las partes afectadas. Este funcionamiento se visualiza en la ilustración (ver Figura 14).

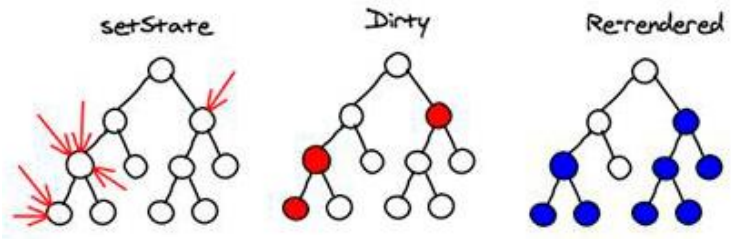

Figura 14 Virtual DOM (Baigorri, 2016)

Representación condicional

Es posible procesar algunos componentes según la necesidad, esto dependerá del estado de la aplicación. Para hacerlo se utiliza la representación condicional, las cuales son similares a los condicionales que se utilizan en JavaScript, es decir se deben utilizar operadores como if o el

operador condicional (condition ? expr1 : expr2) para crear los elementos que representen el estado actual y solicitar a React que actualice la interfaz (Facebook Inc, 2017).

```
function UserGreeting(props) {
  return <h1>Welcome back!</h1>;
}

function GuestGreeting(props) {
  return <h1>Please sign up.</h1>;
}

function Greeting(props) {
  const isLoggedIn = props.isLoggedIn;
  if (isLoggedIn) {
    return <UserGreeting />;
  }
  return <GuestGreeting />;
}

ReactDOM.render(
  // Try changing to isLoggedIn={true}:
  <Greeting isLoggedIn={false} />,
  document.getElementById('root')
);
```

Figura 15 Ejemplo de condicional

2.3.3. Conexión con web services y pruebas

Conexión con web services

React JS es compatible con JavaScript y por el estándar de este último conocido como ECMAScript 5 o ECMAScript 6 es posible realizar peticiones AJAX de forma nativa lo que permite a React JS consumir un servicio únicamente cambiando el estado del componente, como se muestra en el gráfico de ejemplo (ver Figura 16).

```
componentWillMount() {
    fetch('http://taller-angular.carlosazaustre.es/empleados')
      .then((response) => {
        return response.json()
      })
      .then((empleados) => {
        this.setState({ empleados: empleados })
      })
  }
```

Figura 16 Ejemplo de consumo de web service (Azaustre, Consumiendo un API REST desde React.js con ECMAScript6, 2015)

Pruebas

Si una aplicación realizada en React requiere de pruebas unitarias, es posible realizarlas ya que esta herramienta cuenta con las utilidades adecuadas. ReactTestUtils, permite probar los componentes en cualquier marco de pruebas, Facebook recomienda Jest. Entre las funciones que permite realizar esta utilidad se tiene la simulación de eventos en un nodo DOM, renderizar un elemento React en un nodo DOM separado en el documento, verificar si es un elemento React, entre otras.

2.3.4. Instalación y depuración

Para instalar React JS es necesario tener instalado previamente Node.js ya que al igual que con Angular, a través de este entorno se podrá recuperar, instalar y actualizar React y todas las dependencias que requiera el proyecto en cuestión. Luego de Node.js se instala la herramienta Create React App utilizando su respectivo comando de línea que se indica en la página oficial de React JS, a través de este se configuran y crean automáticamente los archivos básicos de la aplicación con las últimas funcionalidades y optimizando para la producción.

En cuanto a la depuración existe una herramienta para Chrome que permite depurar el código escrito en React brindando la oportunidad de visualizar y editar props y state del componente que se ha inspeccionado. Sin embargo existen estudios como el de (Munguía, 2016) y (Cleveroad, 2017), en el cual se menciona que el debugging de HTML es malo y de JavaScript es muy bueno.

2.3.5. Comunidad y navegadores web

Comunidad

El proyecto React fue creado y es mantenido por un equipo de Facebook, al liberarlo en 2013 se crearon varias comunidades y hoy en día es de las más populares. Dentro de las comunidades que se han creado se tiene grupos en Stack Overflow, Twitter, Instagram, GitHub, entre otras. Esta comunidad le ha dado gran seguimiento por lo cual el proyecto ha crecido siendo así que su versión actual es la 15.6, sin embargo, desde su nacimiento se cuenta con

aproximadamente cuarenta y nueve versiones y mantiene un número de seguidores aproximado a sesenta y siete mil usuarios en la plataforma GitHub estos datos para junio 2017.

También cabe rescatar que la comunidad ha desarrollado documentación que se encuentra en la página oficial, existen varios tutoriales, blogs y foros que hablan al respecto. El soporte que brindan es rápido siendo así que mantiene una cuenta de Chat en Discordapp, a través de la cual se puede mencionar cualquier duda. Y son accesibles incluso desde la página oficial de este proyecto.

Compatibilidad con navegadores

Según el estudio realizado por el noruego (Enge, 2014), en el cual se realizan pruebas utilizando TDD, se ha determinado que React JS es compatible con la mayoría de los navegadores, esto se representa en el gráfico mostrado a continuación (ver Figura 17):

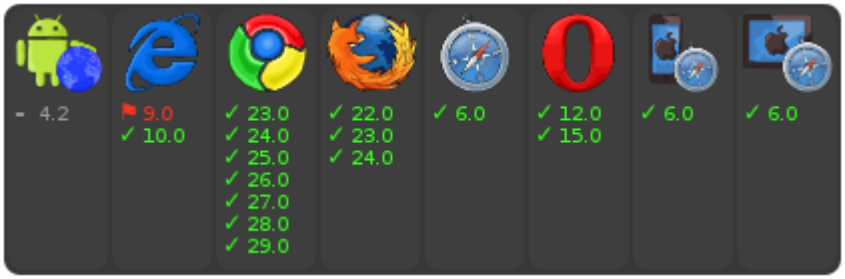

Figura 17 Soporte de React JS en navegadores (Enge, 2014)

2.4. Metodología UWE

La metodología UWE (UML-based Web Engineering) fue propuesta a inicios del año 2000 por la alemana (Koch, 2001), con el fin de facilitar el análisis y diseño de modelos para sistemas orientados a la web, manteniendo todo este proceso durante el ciclo de vida de estas aplicaciones. Cabe resaltar que el proceso unificado de desarrollo y UML son base fundamental de esta metodología, por lo cual tiende a ser iterativa e incremental.

UWE se caracteriza por mantener una notación estándar gracias a la base que tiene de UML, provee métodos para la construcción de cada modelo que

propone la metodología y permite incrementar precisión de los modelos debido a la especificación de restricciones.

Esta metodología presenta un proceso que consta de la fase de captura de requerimientos, análisis y diseño e implementación. Durante el desarrollo de estas etapas se presentan los diferentes modelos que UWE sugiere, estos modelos son de casos de uso, conceptual, de navegación, de presentación, de procesos.

2.4.1. Etapas de desarrollo de UWE

Ante la relación que mantiene UWE con el proceso unificado de desarrollo el proceso en esta metodología es similar ya que se han unificado algunas de las fases de UP. UWE considera únicamente tres etapas las cuales se explican a continuación.

2.4.1.1. Captura de requerimientos

Es el proceso de determinar que aplicación se va a construir. Estos requerimientos pueden ser funcionales o no funcionales se debe considerar que un requerimiento es una condición o capacidad a la que una aplicación debe ajustarse. El principal producto entregable durante esta etapa es el modelo de casos de uso compuesto por actores y casos de uso (Koch, 2001). También se explican otros artefactos como perfiles de usuario que es la descripción de grupos de usuarios y sus tareas.

2.4.1.2. Análisis y diseño

Durante esta etapa se realiza el análisis de los requerimientos funcionales obtenidos en la fase anterior y en el diseño se adapta los resultados obtenidos a las condiciones de los requisitos no funcionales. Todo este análisis y diseño se ve reflejado en los productos entregables que se generan en esta fase, estos son, los modelos conceptual, de navegación, de presentación y proceso como se puede observar en la siguiente figura (ver Figura 18).

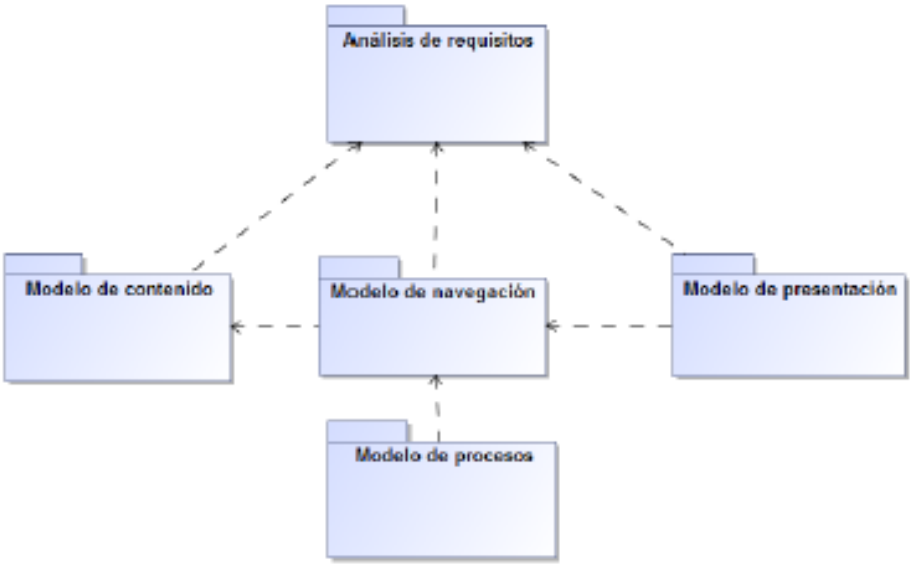

Figura 18 Modelos de UWE (Nieves, Ucán, & Menéndez, 2014)

2.4.1.3. Implementación

Esta etapa consiste en la transformación de los artefactos conseguidos en las fases anteriores a un producto tangible constituido en un sistema, es decir obtener componentes como scripts, ejecutables, código fuente, etc.

2.4.2. Modelos de UWE

Como se identificó en el inciso anterior en las etapas de UWE en cada una de ellas se generan algunos productos entre estos los identificados como modelos los que se detallan a continuación.

2.4.2.1. Modelo de casos de uso

Representa las funciones que corresponden al sistema que se desea, también muestra el entorno y sirve como un documento informativo para los involucrados en el desarrollo de la aplicación. Está compuesto por actores, casos de uso y relaciones o asociaciones. Se identifica por el diagrama de casos de uso (ver Figura 19).

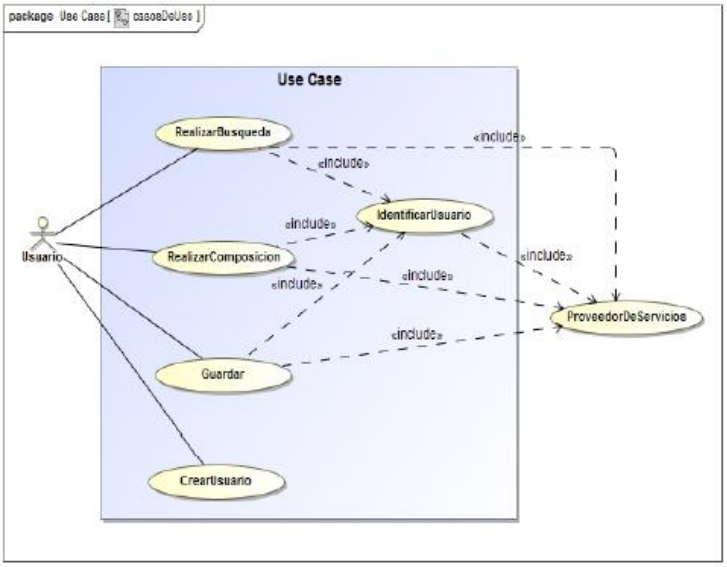

Figura 19 Ejemplo de casos de uso (Nieves, Ucán, & Menéndez, 2014)

2.4.2.2. Modelo conceptual

Se construye de acuerdo a los casos de uso y está representado por un diagrama de clases. Tiene como finalidad expresar visualmente la información del dominio del problema para la aplicación web. Este diagrama está constituido por clases con sus respectivos atributos y métodos, y por las relaciones de clases que pueden ser de agregación, asociación, herencia o uso.

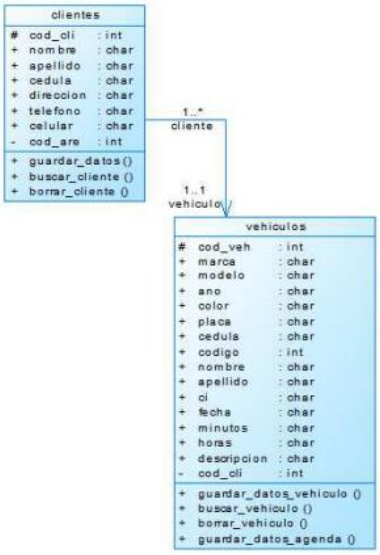

Figura 20 Ejemplo de diagrama de clases (Nolivos, Coronel, Salvador, & Campaña, 2013)

2.4.2.3. Modelo de navegación

El objetivo de este modelo es mostrar la interacción de la aplicación web entre sus diferentes páginas. Está constituido por unidades de navegación y llamadas a otras unidades, conocidos como nodos y enlaces respectivamente. Para establecer el diagrama correspondiente se debe utilizar estereotipos como se muestra (ver Figura 21).

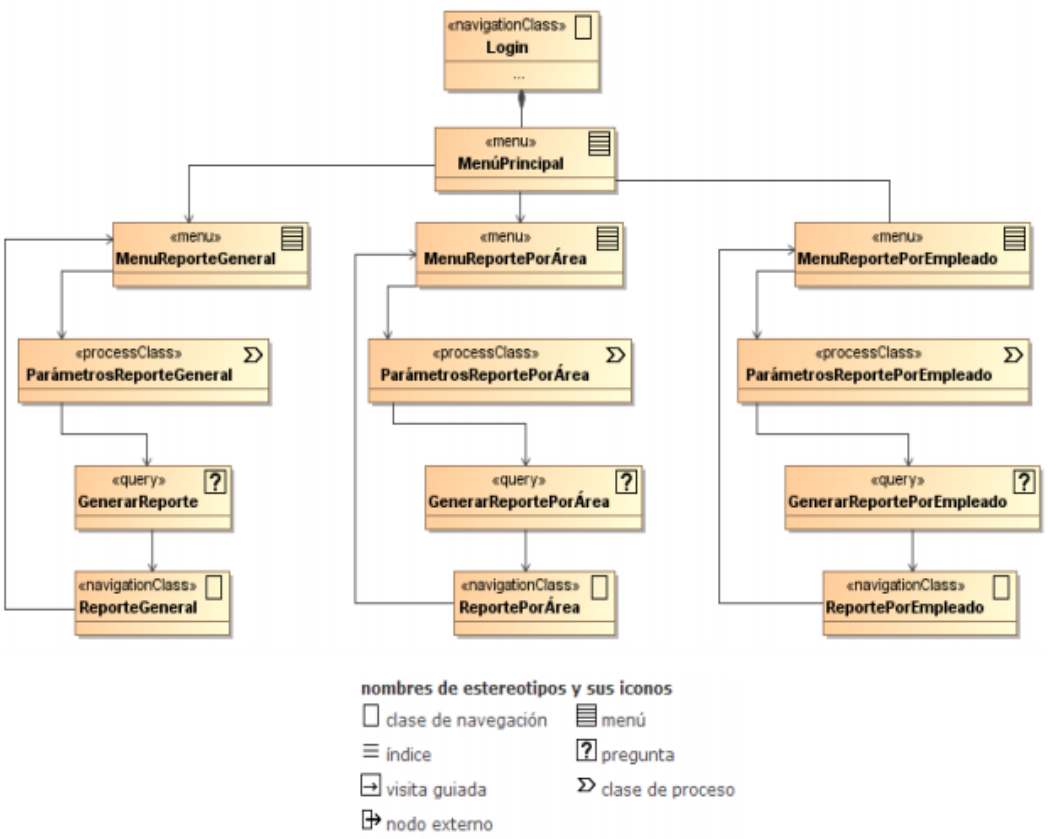

Figura 21 Modelo de navegación y estereotipos de modelo de navegación (Nolivos, Coronel, Salvador, & Campaña, 2013).

2.4.2.4. Modelo de presentación

Básicamente este diagrama es la representación de cómo se muestran los objetos de navegación al usuario. También requiere estereotipos para los respectivos modelos y se visualiza como se indica en el ejemplo ilustrado en la siguiente figura (ver Figura 22).

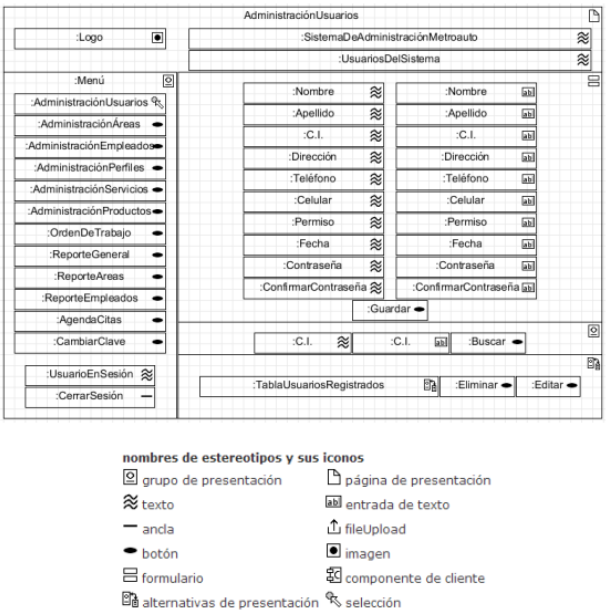

Figura 22 Diagrama de presentación y estereotipos (Nolivos, Coronel, Salvador, & Campaña, 2013).

2.4.2.5. Modelo de procesos

La finalidad de este modelo es integrar los procesos del negocio con el modelo de navegación y definir el comportamiento del proceso. En este modelo se detalla características, flujos de información (Quingaluisa, 2016). Se divide en dos modelos explicados a continuación:

- **Modelo de estructura de proceso:** Se deriva del modelo de contenido y al igual que este se representa por un diagrama de clases, la misma que describe la relación entre todas las clases que lo constituyen.

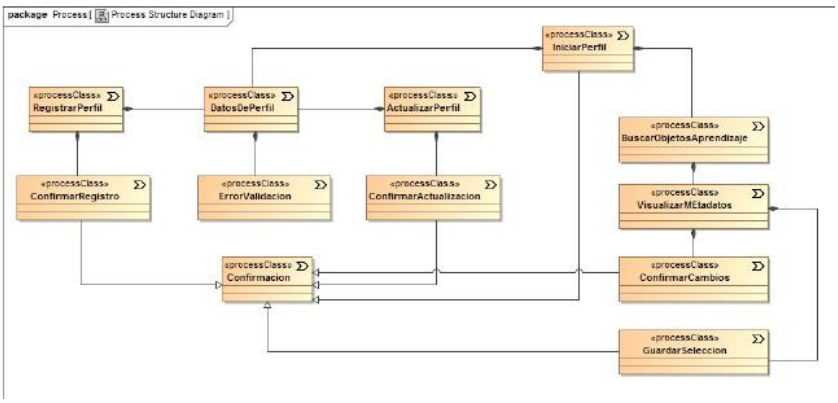

Figura 23 Ejemplo de modelo de estructura de procesos (Nieves, Ucán, & Menéndez, 2014).

- **Modelo de flujo de proceso:** Representado por un diagrama de actividad, especifica todas las actividades relacionadas con las clases de proceso o processClass definidas en el modelo de estructura del proceso.

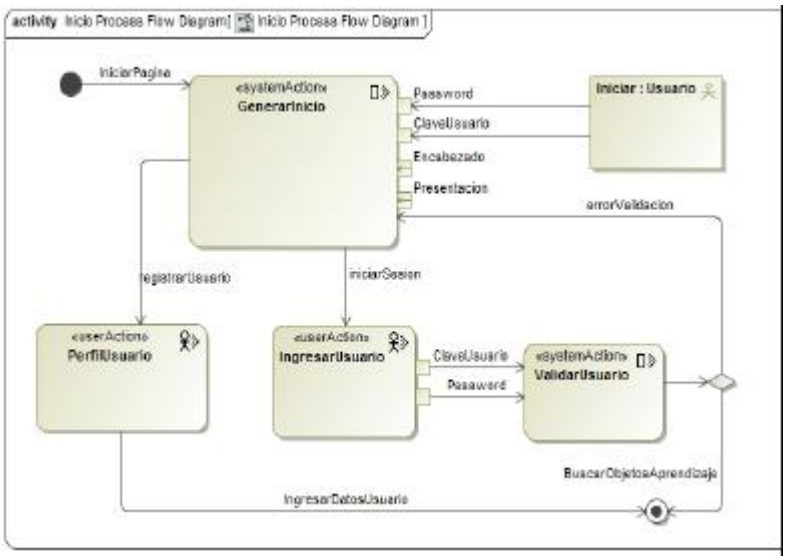

Figura 24 Ejemplo de flujo de proceso (Nieves, Ucán, & Menéndez, 2014)

2.5. Calidad de producto software

Para definir calidad de software es necesario saber la definición de calidad, para esto se considera definiciones establecidas formalmente. La calidad de productos y servicios es la capacidad de satisfacer a clientes tanto en el impacto previsto y no previsto en las partes interesadas (ISO, 2015).

Calidad de software según (Pressman, 2010), “es el proceso eficaz de software que se aplica de manera que crea un producto útil que proporciona valor medible a quienes lo producen y a quienes lo utilizan”. Es decir, la calidad establece brindar apoyo con fines de minimizar esfuerzos; satisfacer y beneficiar al usuario en contenido, funcionalidad y características.

Determinar la calidad de un producto software no es fácil, para ello es necesario cuestionar que propiedades se requieren para evaluar, como estructura un modelo, y como aplicar y evaluar. Así es como aparecen los modelos de calidad.

2.6. Modelos de calidad de producto software

Para medir la calidad de un producto software este se debe analizar desde varios enfoques, sin embargo los principales puntos de vista a ser considerados son el aspecto funcional y el no funcional (Carvallo, 2013). Es decir, la calidad se basa en los servicios que el producto debe proporcionar, su comportamiento en algunas situaciones, restricciones que debe proveer, entre otros. Estos aspectos son muy generales, lo cual lleva descomponerlos y a su vez estudiar los modelos de calidad.

Un modelo de calidad es la integración de buenas prácticas con propuesta de mejora en administración, procesos y avances de calidad. Orientan a una correcta evaluación de calidad y a relacionar todas las características base de requerimientos. Generalmente un modelo de calidad tiene una estructura jerárquica, donde se descomponen los factores de mayor nivel a básicos.

2.6.1. Clasificación de modelos de calidad de producto software

Existen tres tipos de modelos de calidad, estos son los fijos, a medida y mixto.

- Fijos. - Se basan en un catálogo inicial desde el que se inicia la evaluación, construye una jerarquía multinivel, es reusable pero muy poco flexible a cambios, mantiene siempre los mismos factores, al mantener una estructura es fácil de comparar. Por ejemplo, McCall, Boehm, Keller.

- A medida. - Muestra flexibilidad por no depender de ningún catálogo inicial de factores, está orientado a objetivos según el contexto, se descomponen los objetivos a factores que puedan medirse, es adaptable pero su coste es alto en cuanto a su construcción y su reutilización es baja de un proyecto a otro. Por ejemplo: Goal Question Metric (GQM), IEEE 1061.

- Mixtos. - Es una combinación de las ventajas del modelo fijo y a medida, permite la existencia de factores básicos, reusable en varios proyectos, es flexible. Por ejemplo modelo de Gilb, ISO/IEC 9126-1.

2.7. Norma ISO/IEC 9126

La norma ISO/IEC 9126 es un estándar de índole internacional que tiene como objetivo principal la especificación y evaluación de la calidad de software (Solano, 2014). Originalmente esta norma fue concebida en base a los modelos de McCall (ver Figura 25) y Boehm (ver Figura 26).

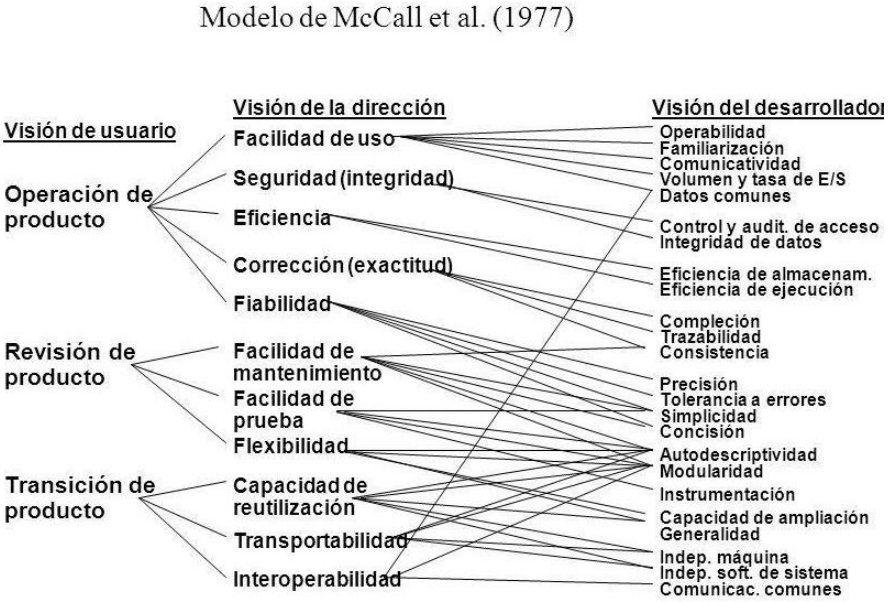

Figura 25 Modelo de McCall. (Gaytan, 2014)

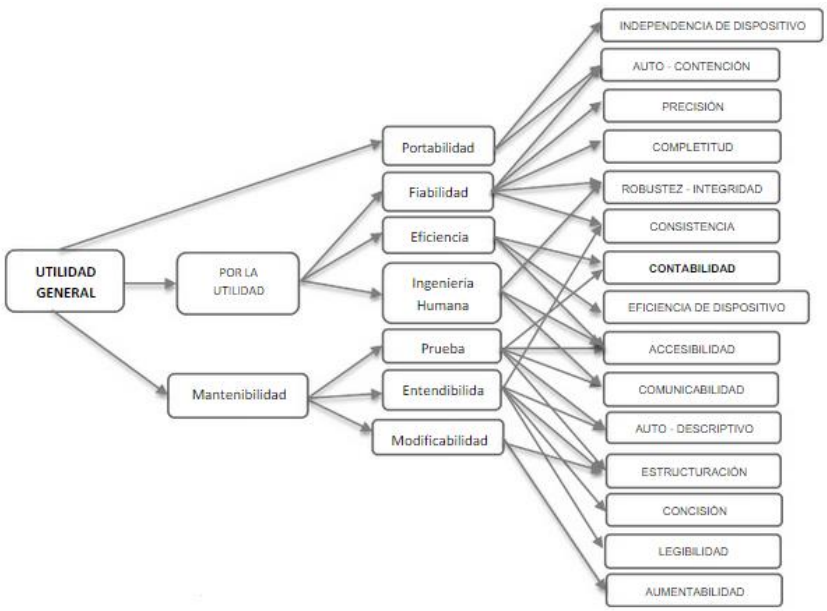

Figura 26 Modelo de Boehm (Murillo)

Anteriormente estaba constituido como un estándar único, pero en la actualidad esta norma está dividida en varias partes:

- ISO/IEC 9126-1: Information technology – Software quality characteristics and metrics – Part 1: Quality model: Esta parte del estándar es la única publicada y en ella se recomienda el modelo de calidad, el mismo que brinda características que a su vez se derivan en subcaracterísticas y pueden refinarse a atributos de calidad internos y externos.
- ISO/IEC 9126-2: Information technology – Software quality characteristics and metrics – Part 2: External metrics: Esta parte provee posibles métricas de calidad externas para medir las características de calidad en una etapa posterior de desarrollo y después del proceso de operación.
- ISO/IEC 9126-3: Information technology – Software quality characteristics and metrics – Part 3: Internal metrics: Esta parte proporciona posibles métricas de calidad internas para medir las características de calidad de software aplicables a un producto no ejecutable durante el diseño y la codificación (Djouab & Bari, 2016).
- ISO/IEC 9126-4: Information technology – Software quality characteristics and metrics – Part 4: Quality in use metrics: Sugiere métricas para medir las características de calidad de software aplicables después de que el producto entre en proceso de operación.

Para una mejor comprensión a continuación se procede a explicar el modelo de calidad, es decir la norma ISO/IEC 9126 – 1.

2.7.1. Modelo de calidad: ISO/IEC 9126 – 1

Como se mencionó anteriormente el modelo de calidad en esta norma se basa en calidad interna y calidad externa, para ello se categorizan los atributos de calidad en seis características (ver Figura 3) y a su vez estas se subdividen y pueden ser medidas por métricas internas o externas.

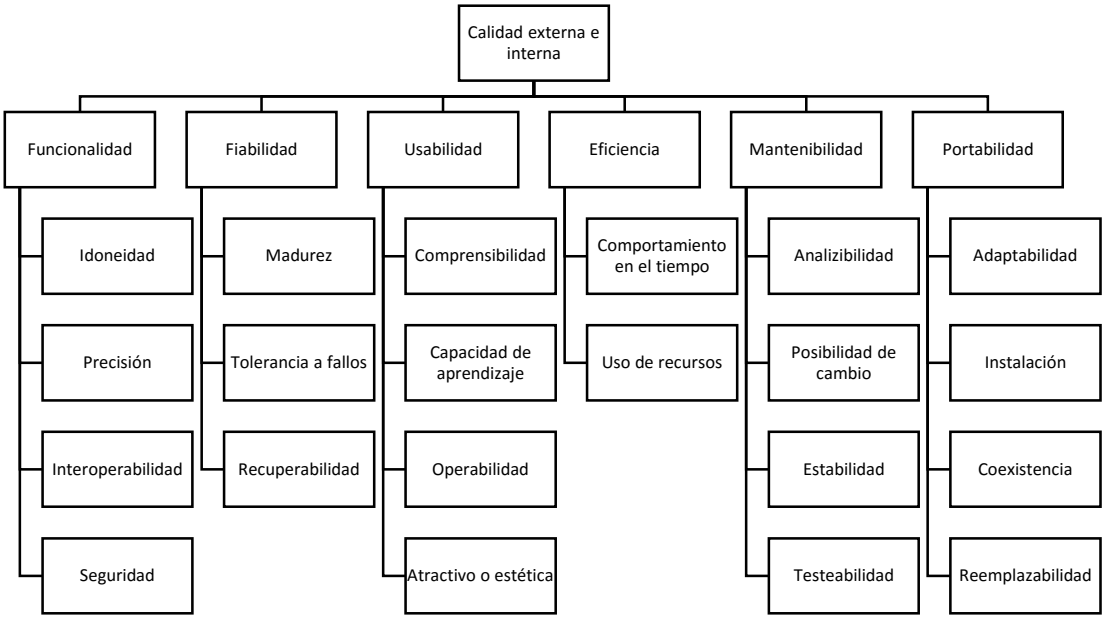

Figura 27 Modelo de calidad interna y externa (International Organization for Standarization, 2000)

2.7.1.1. Calidad interna

Es la calidad del software observada desde un punto de vista interno. Se mide y evalúa en función de los requisitos de calidad internos. Se obtiene mediante revisiones de documentos de especificación, modelos de comprobación o análisis estático del código fuente (Zeiss, Vega, Schieferddecker, Neukirchen, & Grabowski, 2007).

2.7.1.2. Calidad externa

Para la (International Organization for Standarization, 2000), la calidad externa es la totalidad de las características del productos software desde un enfoque externo. Corresponde a la calidad cuando se ejecuta el software, se mide y evalúa con métricas externas mientras se prueba un entorno simulado.

2.7.2. Características y subcaracterísticas

Una vez revisado el modelo de calidad, calidad interna y calidad externa, en esta sección se explica brevemente cada una de las características y

subcaracterísticas establecidas por el estándar, teniendo lo siguiente (ver Tabla 2):

Tabla 2

Características y subcaracterísticas del modelo de calidad ISO/IEC 9126-1 (International Organization for Standarization, 2000)

Características - Subcaracterísticas	Descripción
Funcionalidad	Facultad del software de proveer funcionalidades que cumplan con requisitos bajo alguna condición específica.
Idoneidad	Permite identificar si la herramienta software tiene la capacidad de facilitar funciones para realizar tareas específicas y conseguir objetivos requeridos por el usuario
Precisión	Se refiere a la exactitud que provee el producto software para la entrega de resultados
Interoperabilidad	Interacción de la herramienta software con otros sistemas.
Seguridad	Permite medir la facultad que una herramienta software tiene para la protección de datos como credenciales de acceso, permiso de escritura y lectura, diferenciando autorizaciones y denegaciones.
Fiabilidad	Capacidad de mantener un nivel de rendimiento específico bajo ciertas condiciones.
Madurez	Grado en el que un producto software evita fallas dado un defecto en el mismo.
Tolerancia a fallos	Facultad de la herramienta para conservar un adecuado rendimiento en ocasiones donde existan fallas o vulnerabilidades de interfaz.

Continúa

Recuperabilidad	Se refiere a que el producto software recupere un nivel de rendimiento y a su vez los datos afectados ante una falla.
Usabilidad	Posibilidad que brinda el producto al usuario bajo ciertas condiciones para ser estudiado con facilidad, utilizado y estético.
Comprensibilidad	Es la facultad de la herramienta software con la cual el usuario puede saber si es idóneo. Además brinda la información necesaria para la ejecución de tareas específicas.
Capacidad de aprendizaje	Cualidad del producto que permite aprenderlo.
Operabilidad	Permite identificar si el producto software es administrable y controlable.
Atractivo o Estética	Cualidad del producto para brindar estética al usuario.
Eficiencia	Capacidad de proveer el rendimiento apropiado en relación a los recursos utilizados.
Comportamiento en el tiempo	Permite determinar si el software proporciona de alguna manera tiempos de respuesta, proceso y rendimiento adecuados bajo ciertos parámetros.
Uso de recursos	Facultad para utilizar apropiadamente los recursos ante funciones que la herramienta software realice.
Mantenibilidad	Capacidad a ser modificado, esto incluye correcciones, mejoras, adaptaciones debido a cambios en el entorno y requerimientos funcionales.
Analizibilidad	Permite establecer si el software en cuestión provee la manera de ser diagnosticado ante algún fallo.

Continúa

Posibilidad de cambio	Facultad para permitir una modificación específica a ser incluida.
Estabilidad	Capacidad para evadir efectos imprevistos por cambios en el software.
Testeabilidad	Cualidad del producto software de validación de las modificaciones.
Portabilidad	Capacidad del producto software para transportarlo de un ambiente a otro.
Adaptabilidad	Capacidad de adaptarse a diferentes ambientes específicos, sin necesidad de aplicar acciones o medios distintos de los previstos para este fin
Instalación	Capacidad de ser instalado en un medio.
Coexistencia	Capacidad de una herramienta software de entenderse con otro, en un mismo entorno y compartiendo recursos.
Reemplazabilidad	Capacidad de la herramienta software que provee la facilidad de reemplazarlo por otro producto para el mismo fin y bajo las mismas condiciones.

2.8. Metodologías de construcción de modelos.

2.8.1. Web Quality Evaluation Methodology (WebQEM)

La metodología WebQEM se basa en un modelo de calidad jerárquico, en este caso el que ofrece ISO/IEC 9126. Fue desarrollado por Luis Olsina con la finalidad de evaluar la calidad de productos software orientados a la Web, se adapta a características netamente de la web. Ha sido utilizada en algunos casos de estudio como por ejemplo en el estudio de evaluación a sitios web de museos (Olsina, Web-site Quality Evaluation Method: a Case Study on Museums, 1999), (Kumar, Kumar, & Mathur, 2014), (Covella, 2005) y entre otros.

En el estudio comparativo de modelos de calidad realizado por (Gonzáles, André, & Hernández, 2015) se lo clasifica como metodología para establecer

modelos mixtos, permite evaluar calidad externa y tiene un nivel de jerarquización de capa tres.

Como se mencionó anteriormente esta metodología tiene como marco conceptual la ISO/IEC 9126 y es por eso que plantea cuatro características como son facilidad de uso, funcionalidad, confiabilidad, eficiencia (Scalone, 2006). La metodología sugiere descomponer dichas características a un nivel más abstracto hasta que puedan ser medidas. Esto se comprende mejor explicando el proceso de WebQEM (ver Figura 28) que a continuación se muestra.

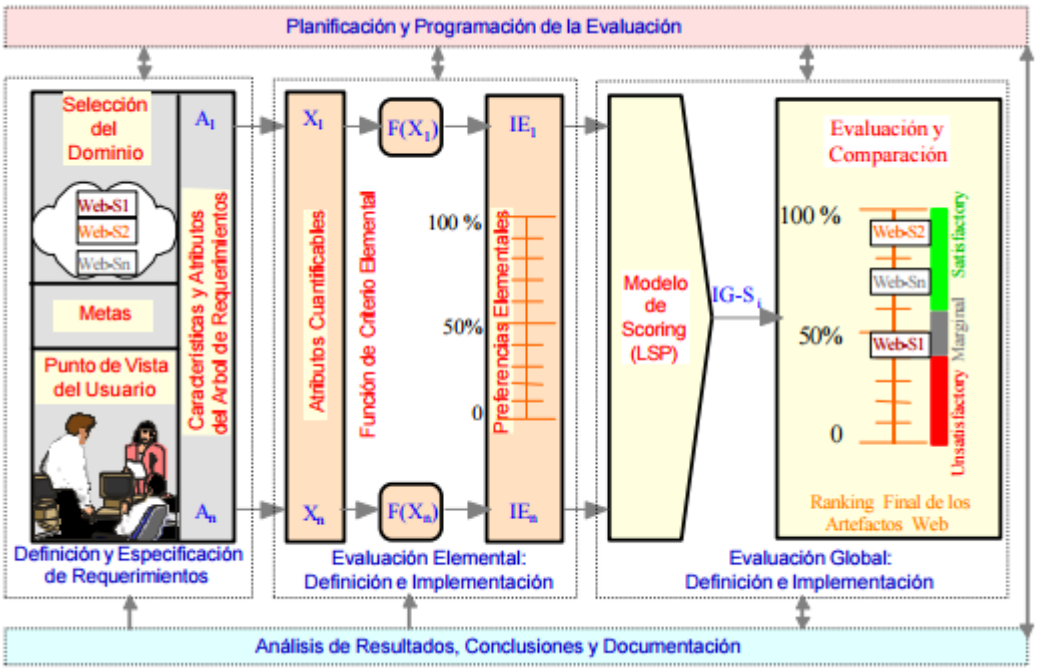

Figura 28 Proceso de WebQEM (Olsina, Metodología Cuantitativa para la Evaluación y Comparación de la Calidad de Sitios Web, 1999)

Los pasos a seguir en esta metodología son los siguientes:

- Definición del dominio para la evaluación de calidad. - Para evaluar se debe conocer con exactitud el alcance de dominio de la aplicación a evaluar, así al realizar un estudio comparativo estos deben estar dentro del mismo dominio.

- Definición de las metas de evaluación y selección del perfil de usuario. - Se deben establecer las metas de evaluación para tener

claro el panorama y se debe considerar que estas metas y la selección de las características y atributos a evaluar cambian según el perfil de la audiencia.

- Definición y especificación de los requerimientos de calidad. - En esta etapa se genera la jerarquía correspondiente a las características, subcaracterísticas y atributos derivados. Esto debe ser escrito en un árbol de requerimientos.
- Definición e implementación de la evaluación elemental. - Se establecen criterios que permitan evaluar elementalmente y los encargados deben medirlos y conseguir valores de indicadores elementales apropiados. Para esto existen las métricas que tienen un rango de valores aceptables y están designadas a cada atributo, adicionalmente sirven de ayuda para obtener indicadores elementales.
- Definición e implementación de la evaluación global. - Utilizando modelos de agregación y sus respectivos cálculos se obtienen indicadores de calidad, parciales y globales. Esto puede hacerse mediante agrupamiento de los indicadores elementales.
- Análisis de resultados, conclusiones y recomendaciones. - Se compara y evalúa los resultados anteriormente obtenidos, es decir parciales y globales, para esto se requiere las metas y el perfil del usuario establecidas previamente. Se procede a documentar presentando conclusiones y recomendaciones.

2.8.2. Software Quality Assurance Exercises (SQAE)

SQAE, es un método desarrollado por una corporación llamada MITRE, cuya finalidad es permitir producir sistemas con resultados fiables a lo largo de todo el ciclo de vida del mismo. Se basa en los modelos de McCall, Boehm y Dromey, por lo cual establece una jerarquía compuesta de atributos tangibles y medibles.

La jerarquía que tiene esta metodología parte de cuatro áreas de calidad las mismas que se relacionan con sus respectivos factores de calidad y de

estos se derivan atributos los que pueden ser cuantificados a través de métricas y medidas. Entre el factor y el área de calidad se utilizan porcentajes, esto permite definir un mapeo entre ellos (ver Figura 29).

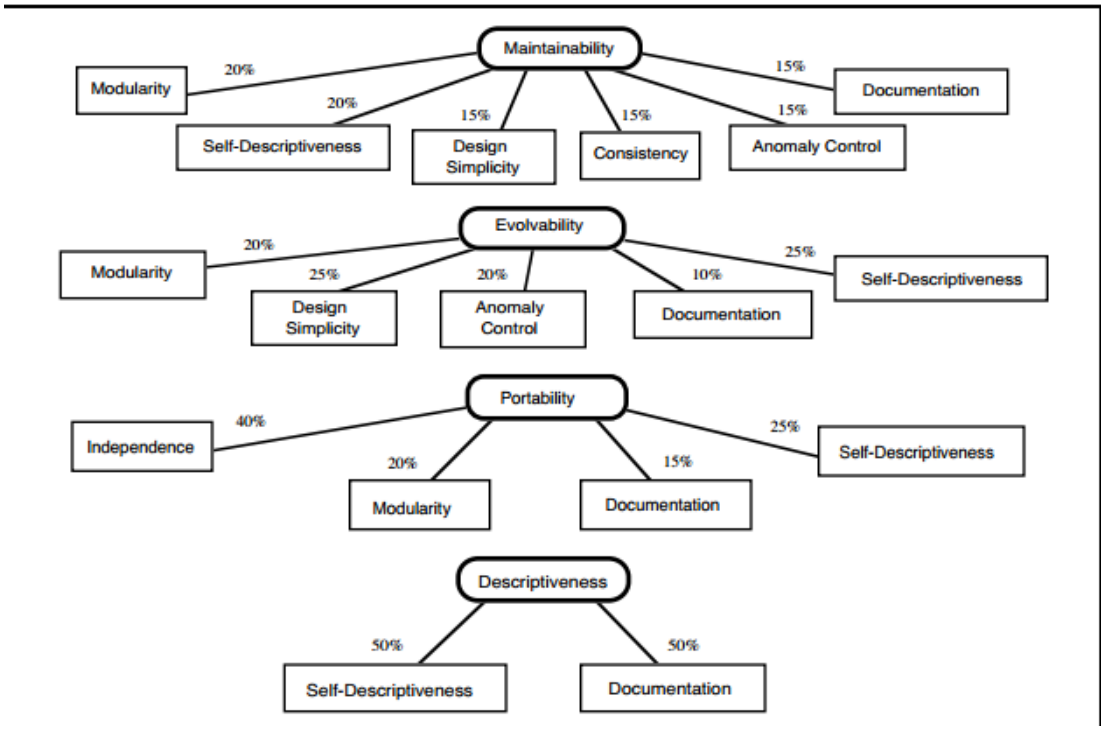

Figura 29 Mapeo de SQAE (Martin & Shafer, 1996)

El proceso que debe seguir esta metodología se aplica a cada atributo y es el siguiente:

- Alcance de evaluación
- Dato del atributo
- Criterio de evaluación
- Contexto de evaluación

En la evaluación de (Gonzáles, André, & Hernández, 2015) indica el tipo de calidad que evalúa, siendo este calidad interna. Además, clasifica este método como uno que permite construir modelos de calidad mixtos y menciona que el modelo presenta tres capas.

2.8.3. Individual Quality Model Construction (IQMC)

Esta metodología fue diseñada tomando como referencia la norma ISO/IEC 9126-1. Proporciona una guía para determinar los factores correctos con los que se construye modelos de calidad mixtos. Este método propone algunos pasos (ver Figura 30) que pueden ser simultaneados o iterados. El primer paso consiste en el estudio del ámbito de calidad y los restantes son para refinamiento de características hasta atributos, lo que permite la construcción del modelo. El procedimiento se indica:

- Estudio del ámbito del software. - Estudiar completamente el entorno del software y de cada componente cuya calidad se desea evaluar. En caso poseer los conocimientos, este paso puede omitirse.
- Determinación de características de calidad. - Parte del catálogo que ofrece el estándar ISO/IEC 9126-1, iniciando con las seis características que provee. Estas pueden reformularse, eliminarse o incluso añadir otras de ser necesario.
- Refinamiento de la jerarquía de características. - Se forma la jerarquía, ya que las características definidas anteriormente se descomponen obteniendo un nivel de abstracción más detallado.
- Refinamiento de subcaracterísticas en atributos. - Descomponer las subcaracterísticas en atributos más pequeños capaces de ser medidos directa o indirectamente con otros factores básicos.
- Refinamiento de atributos derivados en básicos. - Desglosar los atributos hasta conseguir atributos básicos, es decir que sean completamente medibles.
- Establecimiento de relaciones entre factores de calidad. - Se determina la relación que existe entre cada uno de los factores, para determinar la dependencia entre ellos.
- Determinación de métricas para los atributos. Se determinan métricas para medir cada uno de los atributos que han sido reconocidos.

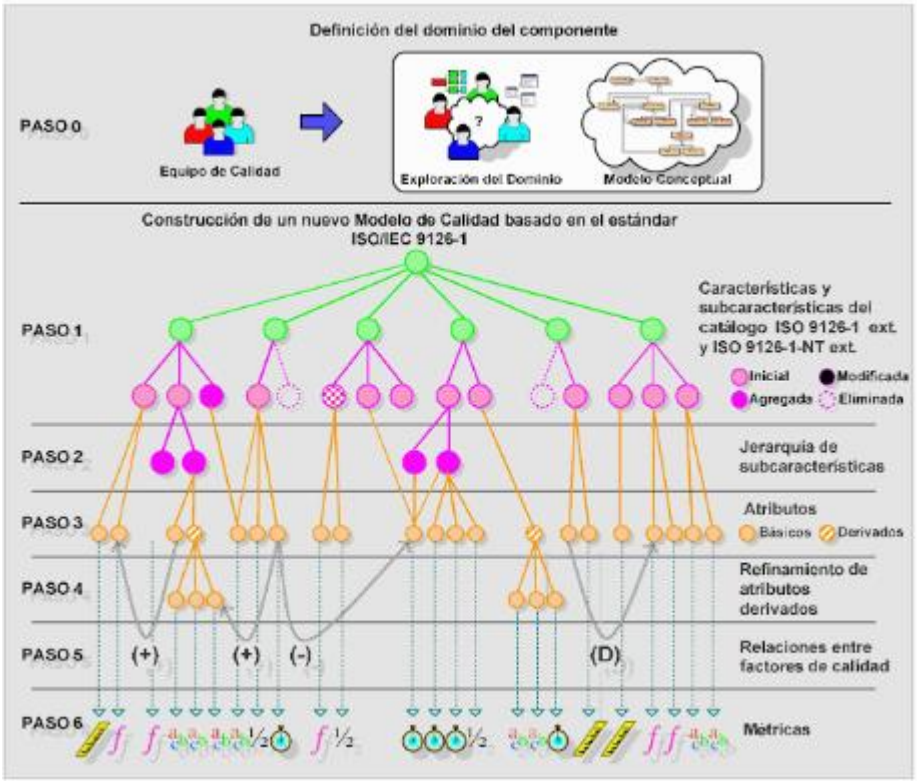

Figura 30 Pasos de IQMC (Calero, Moraga, & Piattini, 2010)

CAPÍTULO III

ANÁLISIS, EVALUACIÓN Y CASO DE ESTUDIO

3.1. Introducción

El presente capítulo se divide en tres partes fundamentales comparación de las metodologías para la construcción del modelo; definición, construcción y aplicación del modelo de evaluación; y desarrollo del caso de estudio. En la primera parte se realizó un breve análisis comparativo de las metodologías para la construcción de modelos, revisando algunas características en común, la relación de cada una con la norma ISO/IEC 9126 establecida en los objetivos de este proyecto (ver sección 1.4) y finalmente se determinaron criterios y métricas que permitieron comparar y seleccionar la metodología más apropiada. En la segunda parte se definió y construyó el modelo de evaluación considerando las características fundamentadas en el estándar ISO/IEC 9126 y siguiendo el procedimiento de la metodología seleccionada en la fase anterior, además se valoró el modelo aplicándolo a herramientas orientadas a componentes web definidas en el alcance del proyecto (ver sección 1.5), esto permitió cuantificar el porcentaje de la herramienta Angular 2 y establecerla como la más destacada. Cabe mencionar que este proceso fue indispensable para la fase final, donde se desarrolló el caso de estudio utilizando la metodología de desarrollo web UWE y la herramienta que se destacó en la parte anterior.

En base a los resultados obtenidos en este apartado y la aplicación desarrollada, en el capítulo 4 se realiza la validación respectiva y en el mismo se detalla la metodología utilizada y resultados correspondientes al estudio de validación.

3.2. Análisis de metodologías de construcción de modelos de evaluación

El análisis de las metodologías WebQEM, SQAE e IQMC realizado en este proyecto, se basa en el modelo de calidad que provee el estándar ISO/IEC 9126 y en las características principales de las metodologías mencionadas.

Respectivamente en las secciones 3.2.1 y 3.2.2 se repasa lo más importante. Después en la sección 3.2.3 se analiza y finalmente en la sección 3.2.4 se detallan los resultados.

3.2.1. Modelo de calidad de producto software

El modelo de calidad de la norma ISO/IEC 9126 se detalló en la sección 2.7.1 del capítulo 2, en resumen, este modelo provee características y subcaracterísticas ordenadas de forma jerárquica, las mismas que permiten medir la calidad interna y externa de un producto software a través del uso de métricas adecuadas.

En el análisis de las metodologías se consideraron las características principales del modelo de calidad expuestas en el capítulo 2 en la sección 2.7.2. Estas características son funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad.

Otro de los aspectos fundamentales a tener en cuenta en este comparativo es la clasificación del modelo de calidad que provean las metodologías que se evaluaron y para ello se debe tener en claro el tipo de modelo al que corresponde ISO/IEC 9126-1 el mismo que es evaluado como un modelo mixto según (Gonzáles, André, & Hernández, 2015)

3.2.2. Estudio de metodologías

Para (Calero, Moraga, & Piattini, 2010), es complicado construir un modelo de calidad debido a algunos factores como la inexperiencia del equipo encargado de la construcción, la incongruencia de terminología del dominio con el modelo, desconocimiento de descomposición de niveles del modelo.

Por las dificultades indicadas anteriormente se realizó un estudio de las metodologías WebQEM, SQAE e IQMC, en este se resumieron características relevantes obtenidas del fundamento teórico de las mismas proporcionado en la sección 2.8 del capítulo 2. Las ideas planteadas a continuación (ver Tabla 3) sirvieron fueron base para realizar el análisis realizado en la sección 3.2.3.

Tabla 3

Características principales de WebQEM, SQAE e IQMC

Característica	WebQEM	SQAE	IQMC
Nombre completo	Web Quality Evaluation Model	Software Quality Assessment Exercises	Individual Quality Model Construction
Se basa en	ISO/IEC 9126	Boehm, McCall, Dromey	ISO/IEC 9126
Nivel de conocimiento	Experto	Experto	Experto
Requerimientos que acepta	Funcionales y no funcionales	Funcionales y no funcionales	Funcionales y no funcionales
Características principales	Usabilidad, funcionalidad, confiabilidad, eficiencia	Mantenibilidad, evolutividad, portabilidad, descriptividad	Todas las dadas en el catálogo del estándar ISO/IEC 9126
Tipo de modelo	Mixto	Mixto	Mixto
# de capas	3	3	3
Tipo de calidad que evalúa	Externa	Interna	Interna y Externa
Trabajos relacionados al 29 de mayo 2017	7 millones.	585 mil	330 millones
Fases del modelo	- Planear y programar la evaluación de calidad. • Metas de la evaluación • Perfil de la audiencia	- Alcance de la evaluación. - Dato del atributo. - Criterio para la evaluación. - Contexto de la evaluación.	- Estudio del ámbito del software. - Determinar subcaracterísticas de calidad. **Continúa**

	- Definir y especificar requerimientos de calidad. - Diseñar y ejecutar la evaluación elemental • Proceso de medición • Obtención de indicadores - Diseñar y ejecutar la evaluación global. • Modelos de agregación de indicadores • Cálculos - Analizar resultados.	- Refinar jerarquía de subcaracterísticas. - Refinar subcaracterísticas en atributos. - Refinar atributos derivados en básicos. - Establecer relaciones entre factores de calidad. - Determinar métricas para los atributos.

3.2.3. Análisis comparativo de metodologías

Previamente se ha mostrado información de modelo de calidad y de cada una de las metodologías que se comparan posteriormente. Con dicha información se establece en esta sección cuadros comparativos basándose en cada dato obtenido.

3.2.3.1. Relación de características ISO/IEC 9126 con las metodologías.

Para iniciar se muestra en una tabla las características principales que la norma ISO/IEC 9126-1 indica en su modelo de calidad, después se procede a seleccionar aquellas que correspondan a cada una de las metodologías en cuestión, finalmente se determina con cuantas características cumple cada metodología y se suma para obtener el total correspondiente.

Tabla 4.

Relación de características ISO/IEC 9126-1 con metodologías.

Características ISO/IEC 9126 - 1	WebQEM	SQAE	IQMC
Funcionalidad	X		X
Fiabilidad	X		X
Usabilidad	X		X
Eficiencia	X		X
Mantenibilidad		X	X
Portabilidad		X	X
Total	**4**	**2**	**6**

3.2.3.2. Criterios de evaluación de metodologías.

Así como se relacionó las metodologías con el modelo de calidad a través de las principales características de éste, es importante establecer otro tipo de parámetros que permitan evaluar los métodos. En este apartado se los compara utilizando factores obtenidos del estudio del estudio previo realizado:

- Compatibilidad con ISO/IEC 9126-1: Se basa en el número de características y el tipo de modelo de evaluación que la metodología permite construir (ver Tabla 3).
- Nivel de abstracción: Corresponde al número de capas que la metodología provee (ver Tabla 3). En (Calero, Moraga, & Piattini, 2010) menciona que mientras mayor sea el nivel de descomposición, es más preciso el detalle del componente a evaluar.
- Grado de experticia del evaluador: Se refiere al nivel de experticia que debe tener el evaluador para aplicar la metodología respectiva (ver Tabla 3).

- Aplicación a requerimientos funcionales y no funcionales: Tiene relación con el tipo de requerimientos compatibles con la metodología (ver Tabla 3).
- Reutilizable: Se mide según el tipo de modelo que las técnicas en estudio permiten desarrollar. La teoría correspondiente de tipos de modelo de calidad dice que los modelos que presentan mayor reutilización son los fijos y mixtos.
- Documentación: Este criterio se lo utiliza ya que es un factor importante de selección como sugiere (Vega, Gasca, & Echeverry, 2012) ya que ayuda a verificar la aplicabilidad y actualización de la técnica, modelo, método, etc. en cuestión. Se basa en los resultados de búsquedas en internet.

Para medir cada una de los factores presentados se ha utilizado la ponderación indicada en la tabla siguiente (ver Tabla 5).

Tabla 5

Métricas de evaluación de metodologías.

Criterio	Alto/Si	Medio	Bajo/No
Compatibilidad ISO/IEC 9126-1	3 si x>4 características	2 si x=4 características	1 si x<4 características
Nivel de abstracción	3 si x>2 capas	2 si x=2 capas	1 si x<2 capas
Grado de experticia del evaluador	1	2	3
Aplicación a requerimientos funcionales y no funcionales	2 si cumple los dos requerimientos	No aplica	1 si cumple solo a un requerimientos
Reutilizable	3 si el modelo es mixto	2 si el modelo es fijo	1 si el modelo es a medida

Continúa

Documentación	3 Si x>=150 M	2 Si x<150 M; x > 10 M	1 Si x <= 10M

3.2.4. Evaluación y resultados

3.2.4.1. Evaluación de metodologías de construcción de modelos de evaluación

Una vez definidos los criterios (ver tabla 5) para comparar y evaluar los métodos presentados en esta sección se procede a calificar cada uno de ellos.

Tabla 6

Evaluación de metodologías por criterios

Criterios	WebQEM	SQAE	IQMC
Compatibilidad con ISO/IEC 9126-1	2	1	3
Nivel de abstracción que permite	3	3	3
Grado de experticia de evaluador	2	3	2
Aplicación a requerimientos funcionales y no funcionales	2	2	2
Reutilizable	3	3	3
Documentación	2	1	3
Total	**15**	**14**	**18**

Adicionalmente se muestra un análisis que considera una comparación de las fases principales de cada una de las metodologías, con la finalidad de determinar aquellas similares y las complementarias de ser el caso. Todo esto se fundamenta en el marco teórico de cada una de las etapas de estas técnicas estudiadas.

Tabla 7

Cumplimiento de fases por metodología

Fases	WebQEM	SQAE	IQMC
Conocimiento del dominio o ámbito.	X		X
Primer nivel de características	X	X	X
Segundo nivel de características	X	X	X
Encontrar atributos	X		X
Derivar atributos en medibles	X		X
	Continúa		

Uso de métricas para atributos derivados	X		X
Definir objetivos o metas de evaluación	X	X	
Determinar un perfil de evaluación	X		
Relacionar factores de calidad	X		X
Definir métricas	X	X	X
Análisis de Resultados	X	X	X
Total	**11**	**5**	**9**

3.2.4.2. Resultados

Finalmente, reunida la información en la sección previa, para determinar que metodología se adapta mejor para construir el modelo evaluador de este proyecto, se globaliza los resultados obtenidos. En esta ocasión se muestra un resumen de los resultados de las tablas 3.3 y 3.4.

Tabla 8

Resumen de comparación

	Tabla 6	Tabla 7c	Total
WebQEM	15	11	26
SQAE	14	5	17
IQMC	18	9	27

En la tabla resumen (ver Tabla 8) se observa que IQMC y WebQEM suman resultados similares, 27 y 26 puntos respectivamente. Esto indica que cualquiera de las dos metodologías mencionadas podría aceptarse para la construcción del modelo requerido. Sin embargo, al realizar el análisis del estudio en la sección 3.2.4.1, se refleja que IQMC con 18 puntos supera a los 15 de WebQEM y se determina que, IQMC tiene mejor compatibilidad con el estándar ISO/IEC 9126-1, lo cual representa definir un modelo más detallado y preciso para la evaluación de las herramientas web correspondientes. Adicionalmente, se precisa que WebQEM tiene menor documentación en la web.

Por otro lado, la relación con las fases presentadas en las metodologías, WebQEM presenta dos adicionales con respecto a IQMC, los cuales gracias a la flexibilidad de IQMC podrían ser consideradas como sub-etapas del primer paso de éste “Conocimiento del ámbito del software”, pero dado que ya existe un perfil determinado que es el de desarrollador y el objetivo de la

evaluación de las herramientas web es claro, tanto “Definir objetivos o metas de evaluación” y “Determinar un perfil de evaluación” no son requeridos para este proyecto.

Ante lo mencionado, se elige a Individual Quality Model Construction (IQMC) como la metodología a seguir para la elaboración del modelo que permita evaluar y a su vez comparar las herramientas orientadas a web components establecidas.

3.3. Construcción de modelo de evaluación de herramientas con IQMC.

Para la construcción del modelo evaluador se siguió el procedimiento de la metodología Individual Quality Model Construction (IQMC), establecida como la más apropiada en la sección 3.2. Cada uno de los siete pasos sugeridos en la metodología, se desarrollaron en el presente apartado.

3.3.1. Estudio del ámbito del software

Las herramientas a evaluar son frameworks orientados a web components, los cuales están escritos en JavaScript y son herramientas open source. Estos frameworks están definidos en el alcance del proyecto y son Angular 2 y ReactJS.

Por concepto de web components se sabe que cualquier herramienta con este enfoque permite crear todo tipo de componentes y que estos sean reutilizables en una aplicación web o por otros usuarios de ser el caso (Álvarez, 2015). Este estándar además presenta especificaciones como Custom Elements, Templates, HTML Imports, Shadow DOM. La primera se refiere a crear etiquetas personalizadas, la segunda crear plantillas para enseñar datos, la tercera importar archivos que contengan los componentes creados y por último encapsular el componente para guardar integridad de los mismos.

Un aspecto muy importante de las herramientas web actuales es la compatibilidad y adaptación que poseen con los diferentes navegadores como Internet Explorer, Google Chrome, Mozilla Firefox, entre otros. Ya que esto ayuda a determinar la portabilidad que tendrán las aplicaciones desarrolladas con estos frameworks.

Por otro lado, dada la naturaleza de estas herramientas a evaluar, Angular 2 y ReactJS, es fundamental mencionar los objetivos que un framework generalmente debe brindar al desarrollador. El primer objetivo es agilizar el proceso de desarrollo, el segundo objetivo se enfoca a la reutilización que este provea y el tercer objetivo, promover buenas practicas con el uso de patrones (Sousa, 2014). Estos objetivos llevan a que sea sea claro, preciso y fácil de aprender, brinden relación o comunicación con otros lenguajes, estén actualizados y que su código sea útil y fácil de mantener.

En (Bermeo, 2014) se considera fundamental el soporte y documentación que un framework posee, ya que la comunidad responsable de estos proyectos se encargan del mantenimiento, actualizaciones, correcciones y documentación de los mismos. Esto determina un factor de calidad importante en cuanto a que sin un equipo que realice y asegure estas actividades, el producto final no tendrá garantías de su correcto funcionamiento y perdería recomendaciones en el mercado.

3.3.2. Determinación de las características de calidad

Las características principales están establecidas en el estándar ISO/IEC 9126 y se detallan la sección 2.7.2 del capítulo 2. Las características consideradas son: funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad.

3.3.3. Refinamiento de la jerarquía de subcaracterísticas

La norma utilizada en este trabajo de titulación provee características y a su vez sugiere subcaracterísticas de éstas, las cuales se explican en la sección 2.7.2 (ver Tabla 2). En resumen, se lista las características con sus respectivas derivaciones.

- **Funcionalidad:** idoneidad, exactitud, interoperabilidad, seguridad.
- **Fiabilidad:** madurez, tolerancia a fallos, recuperabilidad.
- **Usabilidad:** comprensibilidad, capacidad de aprendizaje, operabilidad, atractivo o estética.
- **Eficiencia:** comportamiento en el tiempo, uso de recursos.

- **Mantenibilidad:** analizibilidad, posibilidad de cambio, estabilidad, testeabilidad.
- **Portabilidad:** adaptabilidad, instalación, coexistencia, reemplazabilidad.

3.3.4. Refinamiento de subcaracterísticas en atributos

Como se observa en la sección 3.3.3 las características se descomponen en características más pequeñas, en esta sección se realiza lo mismo con el objetivo de encontrar atributos que sean menos complejos y en lo posible que sean cuantificables.

Tabla 9

Atributos de Idoneidad

Funcionalidad	
Idoneidad	
Atributo	**Descripción**
Manipulación de DOM	Brinda funcionalidades para el manejo adecuado de elementos HTML5
Comunicación cliente – servidor	Tiene funcionalidades para comunicarse con el servidor.
Validación de formularios	Posee las funcionalidades necesarias para realizar la validación correspondiente de formularios
Enlace de datos bidireccional	Es la manera de obtener y actualizar la información del DOM.

Tabla 10

Atributos de Precisión

Funcionalidad	
Precisión	
Atributo	**Descripción**

Continúa

Exactitud al manipular el DOM	Provee los elementos necesarios para manejar el DOM preciso y adecuadamente
Efectividad en las validaciones	Las funciones de validación que posee son efectivas.

Tabla 11

Atributos de Interoperabilidad

Funcionalidad	
Interoperabilidad	
Atributo	**Descripción**
Intercambio de información con web services.	Tiene la capacidad de comunicarse directamente con algún servicio web disponible.

Tabla 12

Atributos de Seguridad

Funcionalidad	
Seguridad	
Atributo	**Descripción**
Confidencialidad	Aporta funcionalidades para proteger la información en contra de accesos no autorizados.
Integridad	Previene accesos o modificaciones a funcionalidades sin autorización.

Tabla 13

Atributo de Madurez

Fiabilidad	
Madurez	
Atributo	**Descripción**

Continúa

Tiempo de vida	Tiempo que tiene la herramienta web desde su versión liberada en los repositorios GitHub hasta la actualidad.
Actualizaciones disponibles	Disposición de actualizaciones desde su lanzamiento y periódicas.
Popularidad en repositorios reconocidos	Determina el número de usuarios que siguen a la herramienta.

Tabla 14

Atributos de Tolerancia a fallos

Fiabilidad	
Tolerancia a Fallos	
Atributo	**Descripción**
Permite conocer el fallo con información adecuada	Tiene las funcionalidades y controles correspondientes para determinar una falla.

Tabla 15

Atributos de Recuperabilidad

Fiabilidad	
Recuperabilidad	
Atributo	**Descripción**
Tiempo de recuperación de un módulo o librería dañada	Brinda la facilidad de recuperar o recuperación automática de componentes averiados.

Tabla 16

Atributos de Comprensibilidad

Usabilidad	
Comprensibilidad	
Atributo	**Descripción**
Definición de principios	Muestra definiciones e información clara y precisa en el sitio web de la herramienta.
Contenido actualizado	El contenido del sitio web correspondiente se mantiene actualizado

Tabla 17

Atributos de Capacidad de aprendizaje

Usabilidad	
Capacidad de aprendizaje	
Atributo	**Descripción**
Tamaño de la curva de aprendizaje	Existe facilidad de aprendizaje de la herramienta.
Efectividad de la documentación	Se proporciona toda la documentación de funcionalidades, componentes y demás adecuadamente.

Tabla 18

Atributos de Operabilidad

Usabilidad	
Operabilidad	
Atributo	**Descripción**
Basado en clases	Está basada en programación orientada a objetos para el desarrollo de aplicaciones.
Plantillas	Provee los mecanismos necesarios para generar plantillas.

Continúa

Soporte	Tiene el soporte correspondiente para poder ser usado y operado.

Tabla 19

Atributos de Atractivo-Estética

Usabilidad	
Atractivo	
Atributo	**Descripción**
Sintaxis	Provee sintaxis que permite desarrollar código no ambiguo y completo.
Semántica	Permite asegurar que las implementaciones se comporten igual según las funcionalidades.

Tabla 20

Atributos de Comportamiento en el tiempo

Eficiencia	
Comportamiento en el tiempo	
Atributo	**Descripción**
Renderización	Capacidad de una herramienta web en cargar en un navegador.

Tabla 21

Atributos de Uso de recursos

Eficiencia	
Uso de recursos	
Atributo	**Descripción**
Número de librerías	Cantidad de librerías necesarias que necesita una herramientas web para satisfacer requerimientos de un desarrollador.

Tabla 22

Atributos de Analizibilidad

Mantenibilidad	
Analizibilidad	
Atributo	**Descripción**
Identificación de errores en código	Permite reconocer errores de codificación de manera fácil y óptima.
Uso de patrones	Revisa si la herramienta brinda un patrón de diseño o se adapta con facilidad.
Depuración de HTML	Habilidad de identificar y corregir errores en HTML.
Depuración de JavaScript	Habilidad de identificar y corregir errores en JavaScript.

Tabla 23

Atributos de Posibilidad de cambio

Mantenibilidad	
Posibilidad de cambio	
Atributo	**Descripción**
Código abierto	Código fuente puede ser modificado según las necesidades.
Actualizaciones	La herramienta se puede extender con nuevas funcionalidades a través de actualizaciones.

Tabla 24

Atributos de Estabilidad

Mantenibilidad	
Estabilidad	
Atributo	**Descripción**
N/A	N/A

Tabla 25

Atributos de Testeabilidad

Mantenibilidad	
Testeabilidad	
Atributo	**Descripción**
Librerías que permitan realizar pruebas.	Proporciona métodos, librerías específicas para realizar pruebas unitarias.

Tabla 26

Atributos de Adaptabilidad

Portabilidad	
Adaptabilidad	
Atributo	**Descripción**
Se ejecuta en diferentes navegadores	Capacidad de que la herramienta web sea compatible y se ejecute la aplicación en cualquier navegador.
Se acopla a web móvil	Capacidad de adaptarse con facilidad aplicaciones web para dispositivos móviles.

Tabla 27

Atributos de Instalación

Portabilidad	
Instalación	
Atributo	**Descripción**
Manuales	Existen manuales de instalación.
Soporte	Tiene diversas formas de soporte en línea como blogs, multimedia para su instalación.
Tiempo	Cantidad de tiempo que se demora para instalar en su esencia básica.
Compatibilidad	Posibilidad de instalar en cualquier sistema operativo.

Tabla 28

Atributos de Coexistencia

Portabilidad	
Coexistencia	
Atributo	**Descripción**
Coexistencia con JavaScript	Capacidad de la herramienta en funcionar en conjunto con JavaScript.
Coexistencia con CSS	Capacidad de la herramienta en funcionar en conjunto con CSS.
Coexistencia con HTML5	Capacidad de la herramienta en funcionar en conjunto con HTML5.
Coexistencia con TypeScript	Capacidad de la herramienta en funcionar en conjunto con TypeScript.

Tabla 29

Atributos de Reemplazabilidad

Portabilidad	
Reemplazabilidad	
Atributo	**Descripción**
Actualización de versión	Capacidad de reemplazar por una versión reciente sin que afecte al funcionamiento.
Cambio de Herramienta	Facilidad de reemplazo por otra herramienta diferente.

3.3.5. Refinamiento de atributos derivados en básicos

Ya descompuestos los atributos, se realiza el mismo procedimiento a éstos, con la finalidad de obtener atributos más básicos para poder medirlos de algún modo.

Tabla 30

Atributos básicos de Manipulación de DOM

Funcionalidad	
Idoneidad	
Manipulación de DOM	
Atributo derivado	**Descripción**
Creación de componentes	Tiene las funcionalidades apropiadas para crear nuevos componentes.
Reutilización de componentes	Provee mecanismos para utilizar componentes nativos como personalizados.

Tabla 31

Atributos básicos de Comunicación cliente servidor

Funcionalidad	
Idoneidad	
Comunicación cliente servidor	
Atributo derivado	**Descripción**
Método Ajax	Provee las funcionalidades que permiten la recarga parcial y provee los métodos correspondientes para comunicarse con el servidor. Sin necesidad de acudir a librerías externas.

Tabla 32

Atributos básicos de Exactitud al manipular al DOM

Funcionalidad	
Precisión	
Exactitud al manipular el DOM	
Atributo derivado	**Descripción**
Componentes cumplen el objetivo	Los componentes entregan funcionalidad, métodos por el cual fueron diseñados.

Tabla 33

Atributos básicos de Confidencialidad

Funcionalidad	
Seguridad	
Confidencialidad	
Atributo derivado	**Descripción**
Transferencia de datos segura	La herramienta proporciona algún método de transferencia segura de datos.
Datos encriptados	La herramienta permite utilizar métodos de encriptación de datos.
Trato de contraseñas	Permite manejar contraseñas ocultar y visualizar las mismas.

Tabla 34

Atributos básicos de Integridad

Funcionalidad	
Seguridad	
Integridad	
Atributo derivado	**Descripción**
Bloqueo de funcionalidades	Se puede bloquear funcionalidades en una aplicación creada.
Ocultar funcionalidades	Probabilidad de ocultar funcionalidades de acuerdo a la aplicación creada.

Tabla 35

Atributos básicos de popularidad en repositorios reconocidos

Fiabilidad	
Madurez	
Popularidad en repositorios reconocidos	
Atributo derivado	**Descripción**

Continúa

GitHub	Número de seguidores de la herramienta en el repositorio donde se encuentra el código fuente.

Tabla 36

Atributos básicos de Efectividad de documentación

Usabilidad	
Capacidad de aprendizaje	
Efectividad de documentación	
Atributo derivado	**Descripción**
Manuales	Existe disponibilidad de manuales.
Tutoriales	Variedad de tutoriales para desarrollo.
Guías	Se dispone de guías de desarrollo correspondiente a la herramienta.
Artículos	Existe variedad de artículos relacionados a la herramienta.

Tabla 37

Atributos básicos de Sintaxis

Usabilidad	
Atractivo	
Sintaxis	
Atributo derivado	**Descripción**
Simple	Las funcionalidades que provee la herramienta son simples de usar.
Claro	La sintaxis requerida de la herramienta es simple para usar.

Tabla 38

Atributos básicos de Renderización

Eficiencia	
Comportamiento en el tiempo	
Renderización	
Atributo derivado	**Descripción**
Tiempo de renderizado	Tiempo de carga que ofrece la aplicación utilizando la herramienta web.

Tabla 39

Atributos básicos de Ejecución en diferentes navegadores

Portabilidad	
Adaptabilidad	
Se ejecuta en diferentes navegadores	
Atributo derivado	**Descripción**
Internet Explorer	Se adapta y ejecuta en Internet Explorer
Google Chrome	Se adapta y ejecuta en Google Chrome
Mozilla Firefox	Se adapta y ejecuta en Mozilla Firefox
Safari	Se adapta y ejecuta en Safari
Opera	Se adapta y ejecuta en Opera

3.3.6. Establecer relaciones entre factores de calidad

Existen tres tipos de relaciones entre los factores de calidad, estos son solapamiento, transversalidad, dependencia. Para este caso no se requiere de dependencia, pero si presenta las otras dos clases de relación las mismas que tienen el siguiente significado:

- Solapamiento: donde un factor de calidad está involucrado en la descomposición de varias características de nivel superior y las métricas son diferentes.
- Transversalidad: es una relación de solapamiento, sin embargo, también cambia la definición del factor.

3.3.7. Determinación de métricas para los atributos

Las métricas que se utilizan son las que se listan en las tablas siguientes (ver tabla 40), (ver tabla 41) y (ver tabla 42):

Tabla 40

Métricas de cumplimiento

Cumplimiento	**Si**	**No**
Valor	1	0

Tabla 41

Rango de métricas de evaluación

Valoración	Excelente	Bueno	Regular	Malo
Rango	4	3	2	1

Tabla 42

Criterios de métricas de evaluación

Criterio / Métrica	Excelente	Bueno	Regular	Malo
Tiempo de carga (ms)	<100	200	300	>400
# Recursos	<1	4	8	>10
Tamaño aprendizaje (alto, medio, bajo, ninguno)	4	3	2	1
Tiempo de vida (años)	<4	3	2	>1
# actualización	<15	10	5	0
Tiempo recuperación (h)	<1	2	3	>4
Popularidad (#estrellas)	>=100000	<100000 y >=50000	<50000 y >=25000	<25000

3.3.8. Diseño de modelo de evaluación

Tabla 43

Modelo de evaluación

	Subcaracterísticas/Atributos/Derivados	MÉTRICAS
FUNCIONALIDAD	**IDONEIDAD**	
	Manipulación de DOM	
	Creación de componentes	Si=1; No=0
	Reutilización de componentes	Si=1; No=0
	Comunicación cliente – servidor	
	Método Ajax	Si=1; No=0
	Validación de formularios	Si=1; No=0
	Enlace de datos bidireccional	Si=1; No=0
	PRECISIÓN	
	Exactitud al manipular el DOM	
	Componentes cumplen el objetivo	Si=1; No=0
	Eficacia en las validaciones	Si=1; No=0
	INTEROPERABILIDAD	
	Intercambio de información con web services	Si=1; No=0
	SEGURIDAD	
	Confidencialidad	
	Transferencia de datos segura	Si=1; No=0
	Datos encriptados	Si=1; No=0
	Trato de contraseñas	Si=1; No=0
	Integridad	
	Bloqueo de funcionalidades	Si=1; No=0
	Ocultar funcionalidades	Si=1; No=0
FIABILIDAD	**MADUREZ**	
	Tiempo de vida	Años = [4;3;2;1]
	Actualizaciones disponibles	#act = [4;3;2;1]
	Popularidad en repositorios reconocidos.	
	GitHub	#estrella=[4;3;2;1]
	TOLERANCIA A FALLOS	
	Permite conocer el fallo con información adecuada	Si=1; No=0
	RECUPERABILIDAD	
	Tiempo de recuperación de un módulo o librería dañado	t(h) = [4;3;2;1]
USABILIDAD	**COMPRENSIBILIDAD**	
	Definición de principios	Si=1; No=0
	Contenido actualizado	Si=1; No=0
	CAPACIDAD DE APRENDIZAJE	
	Continúa	

	Tamaño de la curva de aprendizaje	(alto, medio, bajo, ninguno) = [4;3;2;1]
	Efectividad de documentación	
	Manuales	calidad (excelente, bueno, bajo, malo) = [4;3;2;1]
	Tutoriales	
	Guías	
	Artículos	
	OPERABILIDAD	
	Basado en clases	Si=1; No=0
	Plantillas	Si=1; No=0
	Soporte	Si=1; No=0
	ATRACTIVO	
	Sintaxis	
	Simple	Si=1; No=0
	Claro	Si=1; No=0
	Semántica	Si=1; No=0
EFICIENCIA	**COMPORTAMIENTO EN EL TIEMPO**	
	Renderización	
	Tiempo de renderización	tiempo (ms) = [4;3;2;1]
	USO DE RECURSOS	
	Número de librerías	# cant = [4;3;2;1]
MANTENIBILIDAD	**ANALIZIBILIDAD**	
	Identificación de errores en código	Si=1; No=0
	Depuración de JavaScript	#(alto, medio, bajo, ninguno) = [4;3;2;1]
	Depuración de HTML	#(alto, medio, bajo, ninguno) = [4;3;2;1]
	Manejo de patrones de diseño	Si=0; No=0
	POSIBILIDAD DE CAMBIO	
	Código abierto	Si=1; No=0
	Actualizaciones	Si=1; No=0
	TESTEABILIDAD	
	Librerías que permitan realizar pruebas	Si=1; No=0
PORTABILID	**ADAPTABILIDAD**	
	Se ejecuta en diferentes navegadores	
	Internet Explorer	Si=1; No=0
	Google Chrome	Si=1; No=0

Continúa 

	Mozilla Firefox	Si=1; No=0
	Safari	Si=1; No=0
	Opera	Si=1; No=0
	Se acopla a web móvil	Si=1; No=0
	INSTALACION	
	Manuales	Si=1; No=0
	Soporte	Si=1; No=0
	Tiempo	Si=1; No=0
	Compatibilidad	Si=1; No=0
	COEXISTENCIA	
	Coexistencia con JavaScript	Si=1; No=0
	Coexistencia con CSS3	Si=1; No=0
	Coexistencia con HTML5	Si=1; No=0
	Coexistencia con TypeScript	Si=1; No=0
	REEMPLAZABILIDAD	
	Actualización de versiones	Si=1; No=0
	Cambio de herramienta	Si=1; No=0

3.4. Aplicación de modelo de evaluación

Tabla 44

Ejecución de modelo evaluador a herramientas orientadas a web components

	Subcaracterísticas/ Atributos/Derivados	**MÉTRICAS**	**ANGULAR 2**	**REACT JS**
FUNCIONALIDAD	**IDONEIDAD**			
	Manipulación de DOM			
	Creación de componentes	Si=1; No=0	1	1
	Reutilización de componentes	Si=1; No=0	1	1
	Comunicación cliente - servidor			
	Método Ajax	Si=1; No=0	1	1
	Validación de formularios	Si=1; No=0	1	1
	Enlace de datos bidireccional	Si=1; N0=0	1	0
	PRECISIÓN			
	Exactitud al manipular el DOM			
	Componentes cumplen el objetivo	Si=1; No=0	1	1
	Eficacia en las validaciones	Si=1; No=0	1	1
	INTEROPERABILIDAD			
	Continúa			

	Intercambio de información con web services	Si=1; No=0	1	1
	SEGURIDAD			
	Confidencialidad			
	Transferencia de datos segura	Si=1; No=0	0	0
	Datos encriptados	Si=1; No=0	0	0
	Trato de contraseñas	Si=1; No=0	1	1
	Integridad			
	Bloqueo de funcionalidades	Si=1; No=0	1	1
	Ocultar funcionalidades	Si=1; No=0	1	1
FIABILIDAD	**MADUREZ**			
	Tiempo de vida	Años = [4;3;2;1]	2	3
	Actualizaciones disponibles	#act = [4;3;2;1]	4	4
	Popularidad en repositorios reconocidos.			
	GitHub	#estrella= [4;3;2;1]	2	3
	TOLERANCIA A FALLOS			
	Permite conocer el fallo con información adecuada	Si=1; No=0	1	1
	RECUPERABILIDAD			
	Tiempo de recuperación de un módulo o librería dañado	t(h) = [4;3;2;1]	4	4
USABILIDAD	**COMPRENSIBILIDAD**			
	Definición de principios	Si=1; No=0	1	1
	Contenido actualizado	Si=1; No=0	1	1
	CAPACIDAD DE APRENDIZAJE			
	Tamaño de la curva de aprendizaje	(alto, medio, bajo, ninguno) = [4;3;2;1]	3	2
	Efectividad de documentación			
	Manuales	Si=1; No=0	1	1
	Tutoriales		1	1
	Guías		1	1
	Artículos		1	1
	OPERABILIDAD			
	Basado en clases	Si=1; No=0	1	0
	Plantillas	Si=1; No=0	1	1
	Soporte	Si=1; No=0	1	1
	ATRACTIVO			

Continúa

	Sintaxis			
	Simple	Si=1; No=0	1	1
	Claro	Si=1; No=0	1	1
	Semantica	Si=1; No=0	1	1
EFICIENCIA	**COMPORTAMIENTO EN EL TIEMPO**			
	Renderización			
	Tiempo de renderización	tiempo (ms) = [4;3;2;1]	3	4
	USO DE RECURSOS			
	Número de librerías	# cant = [4;3;2;1]	3	4
MANTENIBILIDAD	**ANALIZIBILIDAD**			
	Identificación de errores en código	Si=1; No=0	1	1
	Depuración de JavaScript	(alto,medio, bajo,ninguno)= [4;3;2;1]	4	4
	Depuración de HTML	(alto,medio, bajo,ninguno)= [4;3;2;1]	4	3
	Uso de patrones de diseño	Si=1; No=0	1	0
	POSIBILIDAD DE CAMBIO			
	Código abierto	Si=1; No=0	1	1
	Actualizaciones	Si=1; No=0	1	1
	TESTEABILIDAD			
	Librerías que permitan realizar pruebas	Si=1; No=0	1	1
PORTABILIDAD	**ADAPTABILIDAD**			
	Se ejecuta en diferentes navegadores			
	Internet Explorer	Si=1; No=0	1	1
	Google Chrome	Si=1; No=0	1	1
	Mozilla Firefox	Si=1; No=0	1	1
	Safari	Si=1; No=0	1	1
	Opera	Si=1; No=0	1	1
	Se acopla a web móvil	Si=1; No=0	1	1
	INSTALACION			
	Manuales	Si=1; No=0	1	1
	Soporte	Si=1; No=0	1	1
	Tiempo bajo de instalación	Si=1; No=0	1	1
	Compatibilidad	Si=1; No=0	1	1
	COEXISTENCIA			
	Coexistencia con JavaScript	Si=1; No=0	1	1
	Coexistencia con CSS3	Si=1; No=0	1	1
		Continúa		

	Coexistencia con HTML5	Si=1; No=0	1	1
	Coexistencia con TypeScript	Si=1; No=0	1	0
	REEMPLAZABILIDAD			
	Actualización de versiones	Si=1; No=0	1	1
	Cambio de herramienta	Si=1; No=0	0	0

3.5. Resultados finales de evaluación.

Finalmente, como se aprecia en el resumen de resultados (ver Tabla 45), en base al estudio realizado la herramienta con mejor proyección es Angular 2, teniendo una ventaja mayor en la usabilidad y mantenibilidad, seguida de la portabilidad y funcionalidad. En otras características como Fiabilidad, Eficiencia tiene desventaja con relación a React JS. Ante estos resultados Angular 2 es la herramienta orientada a web components que satisface características tanto de componentes web como de herramientas de desarrollo web.

Tabla 45

Resumen de evaluación de herramientas orientadas a web components

Característica	Máximo		Angular 2		React JS	
	%	**Puntaje**	**%**	**Puntaje**	**%**	**Puntaje**
Funcionalidad	15,66	13	13,25	11	12,05	10
Fiabilidad	20,48	17	15,66	13	18,07	15
Usabilidad	19,28	16	18,07	15	15,66	13
Eficiencia	9,64	8	7,23	6	9,64	8
Mantenibilidad	15,66	13	15,66	13	13,25	11
Portabilidad	19,28	16	18,07	15	16,87	14
Total	100	83	87,95	73	85,54	71

El comportamiento de los resultados porcentuales obtenidos anteriormente se aprecia en la siguiente gráfica (ver Figura 31), de esta se determinó la existencia de una relación directa entre Angular 2 y los valores máximos de portabilidad, mantenibilidad, funcionalidad y usabilidad, y una relación inversa

en fiabilidad y eficiencia siendo las menores calificaciones proporcionando ventaja a React JS únicamente en estas características.

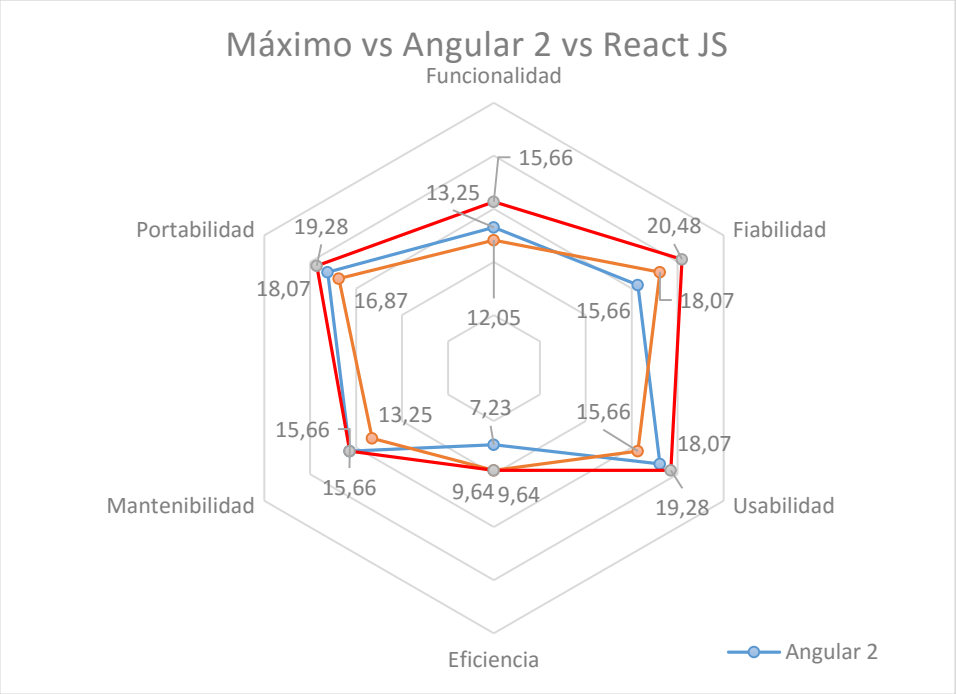

Figura 31 Comportamiento de resultados de la evaluación

3.6. Desarrollo de caso de estudio

Una vez realizado el análisis y determinado la mejor opción entre las herramientas orientadas a web components establecidas, se procede a desarrollar la aplicación con el Framework Angular 2. Para el desarrollo de esta aplicación se ha utilizado la metodología de desarrollo web UWE, ingeniería web basada en UML y ante esto se ha dividido este apartado en tres partes cada una contiene los productos entregables respectivos como se explicó en el capítulo 2 en la sección 2.4.

3.6.1. Captura de requerimientos

En esta fase se describen los diferentes perfiles de usuario con la finalidad de conocer las tareas que deben llevar a cabo cada usuario. El producto más importante es el modelo de casos de uso en el cual se representa prácticamente los requisitos funcionales del sistema.

3.6.1.1. Descripción perfiles de usuario

- Administrador: Es el usuario encargado de llevar la gestión de todos los procesos del negocio, es decir que tiene acceso a las diferentes tareas designadas a otros actores. Las principales tareas de este usuario son las siguientes:
    - Gestionar usuarios: crear, modificar, eliminar y ver toda la información correspondiente.
    - Gestionar estacionamientos: crea, modifica, elimina y observa toda la información pertinente. Esta tarea es indispensable para la funcionalidad del sistema.
    - Eliminación de vehículos y personas.
    - Reportes: Acceder a los reportes dispuestos para su respectiva toma de decisiones.
- Supervisor: Se encarga de revisar diariamente el uso de los estacionamientos dentro de una zona determinada, las tareas designadas para este perfil son:
    - Registrar personas.
    - Modificar información de personas.
    - Registrar vehículos.
    - Modificar la información de vehículos.
    - Registrar la entrada de vehículos.
    - Registrar la salida de vehículos.

3.6.1.2. Casos de Uso

El presente diagrama (ver Figura 32) se muestra la interacción de los usuarios del sistema con el mismo y también refleja los requerimientos del sistema.

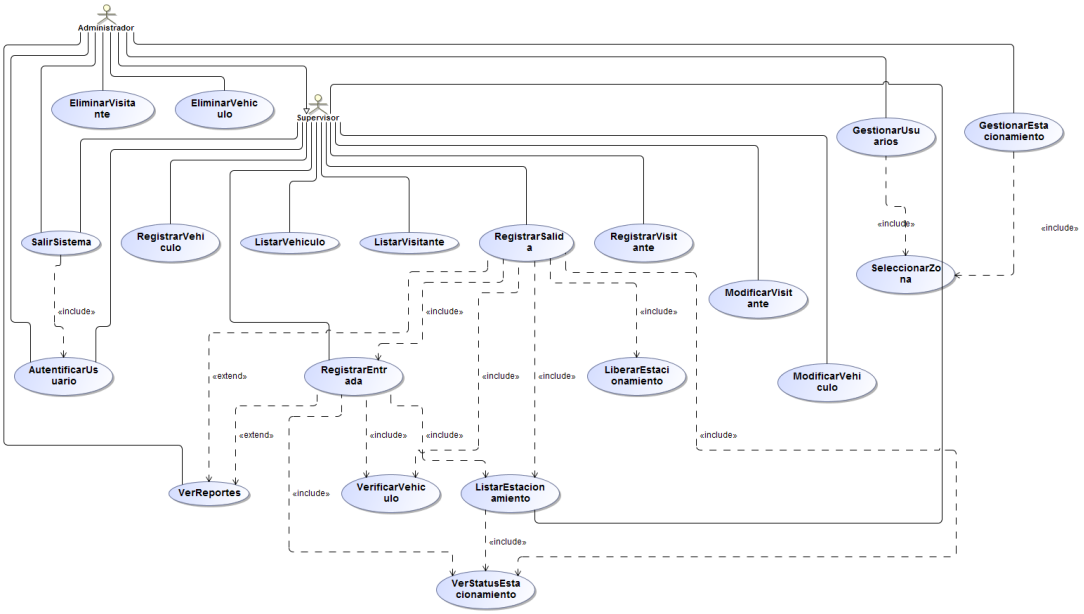

Figura 32 Diagrama de casos de uso del sistema.

3.6.1.3. Descripción de casos de uso.

En esta sección se ha realizado la descripción de cada uno de los casos de uso mostrados en el diagrama anterior (ver Figura 32).

Tabla 46

Caso de uso autenticar usuario

Nombre del C.U.	**AutenticarUsuario**
Actores	Administrador – Supervisor
Descripción	El administrador o supervisor para tener acceso al sistema deben ingresar sus respectivas credenciales.
	Flujos
Flujo básico	1. El usuario ingresa su nombre de usuario y contraseña. 2. El usuario hace clic en “Login”
Flujo alternativo	Ninguno
Precondiciones	El usuario debe haber sido registrado previamente.

Continúa

Post – condiciones	El usuario ingresa a la página principal con el menú correspondiente.
Excepciones	El sistema muestra un mensaje de error si las condiciones no son correctas.

Tabla 47

Caso de uso salir del sistema

Nombre del C.U.	**SalirSistema**
Actores	Administrador – Supervisor
Descripción	El usuario saldrá del sistema cuando ya no requiera utilizarlo.
	Flujos
Flujo básico	1. El usuario hace clic en “Salir”
Flujo alternativo	Ninguno
Precondiciones	Haberse autentificado previamente.
Post – condiciones	El sistema muestra la página para ingresar al sistema.
Excepciones	Ninguna

Tabla 48

Caso de uso salir del sistema

Nombre del C.U.	**EliminarVisitante**
Actores	Administrador
Descripción	El usuario podrá eliminar un visitante registrado.
	Flujos
Flujo básico	1. Seleccionar la opción Visitante. 2. Hacer clic en el botón “eliminar”. 3. Confirmar eliminación del visitante correspondiente.
Flujo alternativo	Ninguno
Precondiciones	Deben existir visitantes registrados.

Continúa 

Post – condiciones	Recargar listado de visitantes.
Excepciones	El sistema muestra un mensaje de error si no se ha podido eliminar.

Tabla 49

Caso de uso eliminar vehículo

Nombre del C.U.	**EliminarVehiculo**
Actores	Administrador
Descripción	El usuario podrá eliminar un vehículo registrado a una persona.
	Flujos
Flujo básico	1. Seleccionar la opción Visitante. 2. Presionar el botón ver del visitante respectivo. 3. Hacer clic en el botón “eliminar” del vehículo. 4. Confirmar eliminación del vehículo correspondiente.
Flujo alternativo	Ninguno
Precondiciones	Deben existir vehículos registrados.
Post – condiciones	Actualizar listado de vehículos.
Excepciones	El sistema muestra un mensaje de error si no se ha podido eliminar.

Tabla 50

Caso de uso gestionar usuarios

Nombre del C.U.	**GestionarUsuarios**
Actores	Administrador

Continúa

Descripción	El administrador encargado puede registrar, editar, listar y eliminar usuarios registrados.
	Flujos
Flujo básico	1. Debe seleccionar la opción “Usuarios” en el menú. 2. Se listan todos los usuarios registrados. 3. Para registrar a. El usuario debe llenar los campos nombre, cédula, teléfono, email, nombre usuario, contraseña. b. Seleccionar la zona a la que se le asignará. c. Seleccionar el perfil que se asignará. d. Guardar la información ingresada. 4. Para modificar a. Se debe seleccionar el usuario haciendo clic en el botón editar. b. Cambiar la información cargada nombre, cédula, teléfono, email, nombre usuario, contraseña, zona, perfil. c. Guardar la información. 5. Para eliminar se debe seleccionar el usuario haciendo clic en el botón eliminar y confirmar la misma.
Flujo alternativo	Ninguno
Precondiciones	Para registrar debe haber ingresado al sistema. Para modificar y eliminar tener registrados usuarios.
Post – condiciones	Actualiza el listado
Excepciones	El sistema muestra un mensaje de error si no se han podido llevar a cabo las acciones determinadas.

Tabla 51

Caso de uso gestionar estacionamientos

Nombre del C.U.	**GestionarEstacionamientos**
Actores	Administrador
Descripción	El administrador encargado puede registrar, editar, listar y eliminar estacionamientos registrados.
	Flujos
Flujo básico	1. Debe seleccionar la opción “Estacionamientos” en el menú. 2. Se listan todos los estacionamientos registrados. 3. Para registrar a. El usuario debe llenar los campos número de parqueadero. b. Seleccionar la zona a la que se le asignará. c. Guardar la información 4. Para modificar a. Se debe seleccionar el estacionamiento haciendo clic en el botón editar correspondiente. b. Cambiar la información cargada número de parqueadero, zona. c. Guardar la información. 5. Para eliminar se debe seleccionar el estacionamiento haciendo clic en el botón eliminar y confirmar la misma.
Flujo alternativo	Ninguno
Precondiciones	Para registrar debe haber ingresado al sistema. Para modificar y eliminar tener registrados estacionamientos.
Post – condiciones	Actualiza el listado

Continúa

Excepciones	El sistema muestra un mensaje de error si no se han podido llevar a cabo las acciones determinadas.

Tabla 52

Caso de uso seleccionar zonas

Nombre del C.U.	**SeleccionarZonas**
Actores	Administrador
Descripción	El sistema listará las zonas designadas y el administrador deberá seleccionar mientras registra usuario o estacionamiento.
	Flujos
Flujo básico	1. Registrar o modificar un usuario o estacionamiento. 2. Seleccionar una zona de la lista cargada.
Flujo alternativo	Ninguno
Precondiciones	El usuario debe haber sido registrado previamente. Ingresar a registrar o modificar de usuario. Ingresar a registrar o modificar de estacionamiento.
Post – condiciones	Guardar el usuario o estacionamiento.
Excepciones	Ninguna

Tabla 53

Caso de uso ver reportes

Nombre del C.U.	**VerReportes**
Actores	Administrador
Descripción	El usuario visualizará información correspondiente al proceso de entradas y salidas que detallan y otros.

Continúa

Flujos	
Flujo básico	1. El usuario debe seleccionar la opción reportes. 2. Seleccionar el reporte que desea visualizar.
Flujo alternativo	Ninguno
Precondiciones	El usuario debe haber ingresa al sistema
Post – condiciones	Ninguna
Excepciones	Si se ha producido algún error durante la generación del reporte se mostrará un mensaje de error.

Tabla 54

Caso de uso registrar vehículo

Nombre del C.U.	**RegistrarVehiculo**
Actores	Supervisor
Descripción	El supervisor podrá ingresar la información correspondiente de un vehículo.
Flujos	
Flujo básico	1. Seleccionar la opción “Visitante” en el menú. 2. Seleccionar con el botón “Ver” del visitante correspondiente. 3. Llenar los datos correspondientes placa, marca, modelo, color. 4. Guardar los datos
Flujo alternativo	Ninguno
Precondiciones	Debe haber ingresado al sistema. Tener registrado visitantes.
Post – condiciones	Actualización de listado de vehículos.
Excepciones	Se muestra mensaje de error si se produce algún problema.

Tabla 55

Caso de uso listar vehículo

Nombre del C.U.	**ListarVehiculo**
Actores	Supervisor
Descripción	El administrador puede ver todos los vehículos correspondientes a un visitante.
	Flujos
Flujo básico	1. Seleccionar la opción “Visitante”. 2. Seleccionar el visitante corresponiente presionando el boton “Ver”.
Flujo alternativo	Ninguno
Precondiciones	Tener registrados visitantes. Haber ingresado al sistema.
Post – condiciones	Ninguna
Excepciones	Se muestra un mensaje de error si existen problemas al mostrar el listado.

Tabla 56

Caso de uso modificar vehículo

Nombre del C.U.	**ModificarVehiculo**
Actores	Supervisor
Descripción	El supervisor podrá modificar la información correspondiente de un vehículo.
	Flujos
Flujo básico	1. Seleccionar la opción “Visitante” en el menú. 2. Seleccionar al visitante con el botón “Ver” correspondiente. 3. Presionar el botón “Editar” respectivo.

Continúa

	4. Modificar los datos correspondientes placa, marca, modelo, color. 5. Guardar los datos
Flujo alternativo	Ninguno
Precondiciones	Debe haber ingresado al sistema. Tener registrado visitantes. Tener vehículos registrados.
Post – condiciones	Actualización de listado de vehículos.
Excepciones	Se muestra mensaje de error si se produce algún problema.

Tabla 57

Caso de uso listar visitante

Nombre del C.U.	**ListarVisitante**
Actores	Supervisor
Descripción	El administrador puede listar y ver información de todos los visitantes registrados.
	Flujos
Flujo básico	1. Seleccionar la opción “Visitante” del menú.
Flujo alternativo	Ninguno
Precondiciones	Haber ingresado al sistema.
Post – condiciones	Ninguna
Excepciones	Se muestra un mensaje de error si existen problemas al mostrar el listado.

Tabla 58

Caso de uso registrar visitante

Nombre del C.U.	**RegistrarVisitante**
Actores	Supervisor

Continúa

Descripción	El supervisor podrá ingresar visitantes al sistema.
	Flujos
Flujo básico	1. Seleccionar la opción “Visitantes” del menú. 2. Presionar el botón “Agregar”. 3. Llenar los datos del formulario nombre, cédula, teléfono, email. 4. Guardar la información.
Flujo alternativo	Ninguno
Precondiciones	El usuario debe haber ingresado al sistema.
Post – condiciones	Actualizar el listado de visitantes.
Excepciones	El sistema muestra un mensaje de error si existe algún problema al guardar.

Tabla 59

Caso de uso registrar visitante

Nombre del C.U.	**ModificarVisitante**
Actores	Supervisor
Descripción	El supervisor podrá modificar información de visitantes.
	Flujos
Flujo básico	1. Seleccionar la opción “Visitantes” del menú. 2. Presionar el botón “Editar” correspondiente al visitante que se desea modificar. 3. Cambiar los datos cargados en el formulario nombre, cédula, teléfono, email. 4. Guardar la información.
Flujo alternativo	Ninguno
Precondiciones	El usuario debe haber ingresado al sistema. Tener registrados visitantes.
Post – condiciones	Actualizar el listado de visitantes.

Continúa

Excepciones	El sistema muestra un mensaje de error si existe algún problema al guardar.

Tabla 60

Caso de uso verificar vehículo

Nombre del C.U.	**VerificarVehiculo**
Actores	Supervisor
Descripción	El supervisor tendrá que verificar si el vehículo ya se encuentra en algún estacionamiento.
	Flujos
Flujo básico	1. Seleccionar la opción “Entrada y Salida” del menú. 2. Del listado que se muestra presionar el botón entrada correspondiente al estacionamiento que se va a asignar. 3. En la caja de texto Placa el usuario ingresará la placa y el sistema automáticamente verificará el vehículo.
Flujo alternativo	Si el vehículo está asignado a otro estacionamiento, se muestra un mensaje indicando que ya se encuentra en otro estacionamiento.
Precondiciones	El usuario debe haber ingresado al sistema. El estacionamiento debe estar libre.
Post – condiciones	Llenar el formulario con los datos respectivos al vehículo.
Excepciones	Si se produce algún error al buscar se muestra en un mensaje de error.

Tabla 61

Caso de uso ver estado de estacionamiento

Nombre del C.U.	**VerStatusEstacionamiento**
Actores	Supervisor
Descripción	El usuario puede acceder a un listado de estacionamientos en el cual se puede observar su respectivo estado, es decir si está libre u ocupado.
	Flujos
Flujo básico	1. Seleccionar la opción “Entrada y Salida” del menú. 2. Observar el estado del estacionamiento, este ya viene cargado previamente.
Flujo alternativo	Ninguno
Precondiciones	El usuario debe haber ingresado al sistema.
Post – condiciones	Ninguna
Excepciones	Ninguna

Tabla 62

Caso de uso registrar entrada

Nombre del C.U.	**RegistrarEntrada**
Actores	Supervisor
Descripción	El usuario podrá registrar la entrada de un vehículo a algún estacionamiento.
	Flujos
Flujo básico	1. Seleccionar la opción “Entrada y Salida” del menú. **Continúa** 

	2. Del listado que se muestra presionar el botón entrada correspondiente al estacionamiento que se va a asignar al vehículo. 3. Llenar los datos correspondientes como placa, observación. 4. Guardar la información.
Flujo alternativo	Ninguno
Precondiciones	El usuario debe haber ingresado al sistema. Verificar vehículo.
Post – condiciones	Actualización del estado correspondiente en el listado de estacionamientos.
Excepciones	El sistema muestra un mensaje de error si se produce un problema al guardar la información.

Tabla 63

Caso de uso listar estacionamiento

Nombre del C.U.	**ListarEstacionamiento**
Actores	Supervisor
Descripción	El usuario podrá únicamente ver un listado de los estacionamientos existentes.
	Flujos
Flujo básico	1. Seleccionar la opción “Entrada y Salida” del menú. 2. Se muestra un listado con todos los estacionamientos activos.
Flujo alternativo	Ninguno
Precondiciones	El usuario debe haber ingresado al sistema. Tener registrado al menos un estacionamiento.
Post – condiciones	Ninguno
Excepciones	Ninguno

Tabla 64

Caso de uso registrar salida

Nombre del C.U.	**RegistrarSalida**
Actores	Supervisor
Descripción	El usuario podrá registrar la salida de un vehículo que ha ingresado
	Flujos
Flujo básico	1. Seleccionar la opción “Entrada y Salida” del menú. 2. Se muestra un listado con el estado de todos los estacionamientos. 3. Presionar el botón salida correspondiente al estacionamiento. 4. Confirmar registro de salida.
Flujo alternativo	Ninguno
Precondiciones	El usuario debe haber ingresado al sistema. Haber registrado entrada para ese estacionamiento.
Post – condiciones	Actualizar estado en el listado de estacionamientos.
Excepciones	Si no se ha podido registrar salida notificar con un mensaje de error.

3.6.2. Análisis y diseño

En esta fase se refleja todo lo realizado durante el proceso de captura de requerimientos y se muestran en los diferentes modelos que la metodología sugiere.

3.6.2.1. Modelo conceptual

Este modelo se representa por un diagrama de clases, para este proyecto el diagrama correspondiente se presenta a continuación (ver Figura 33).

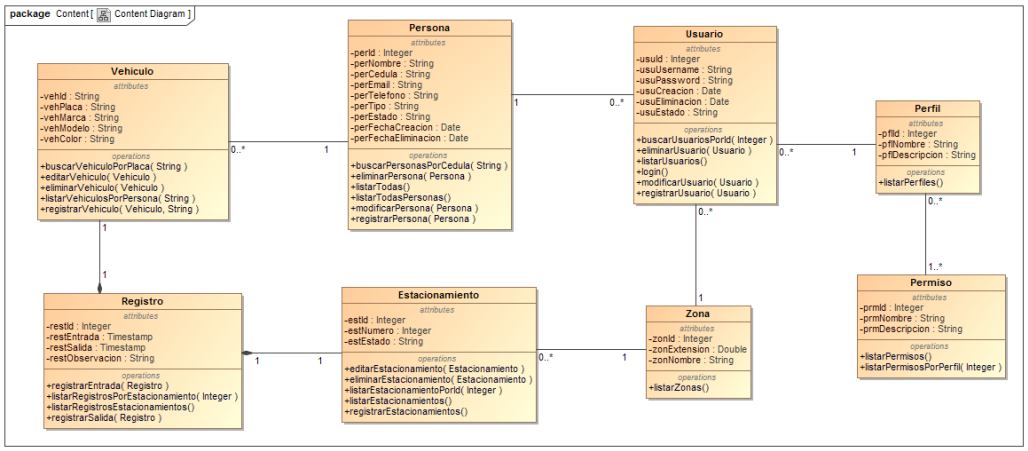

Figura 33 Diagrama de modelo conceptual del sistema

3.6.2.2. Modelo de navegación

El diagrama de navegación representa la navegación que el usuario tendrá en el sistema y se representa de la siguiente manera (ver Figura 34).

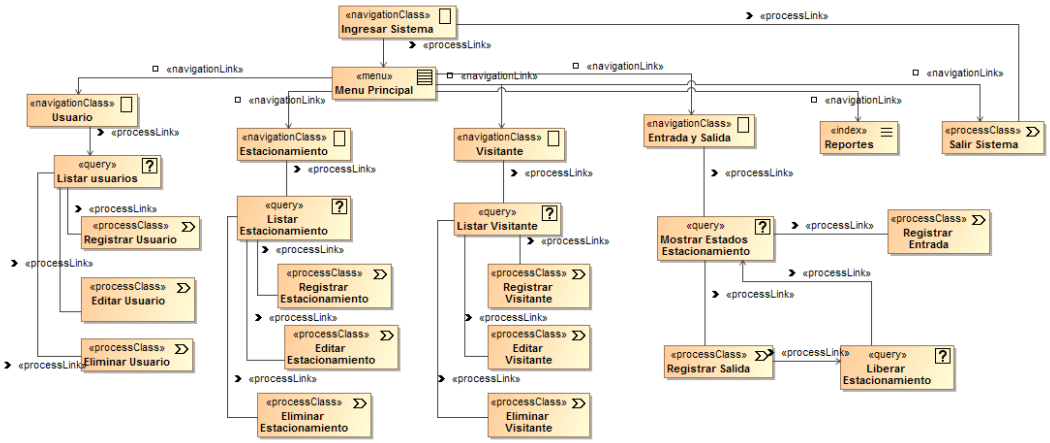

Figura 34 Diagrama de modelo de navegación del sistema.

3.6.2.3. Modelo de presentación

El modelo de presentación se encarga de mostrar el diseño de cada pantalla que se aprecia en la aplicación y se han modelado en diferentes etapas que se indican en las siguientes figuras. En primer lugar se indica la el diseño de la pantalla correspondiente al inicio de sesión en la aplicación y se muestra en el siguiente modelo (ver Figura 35).

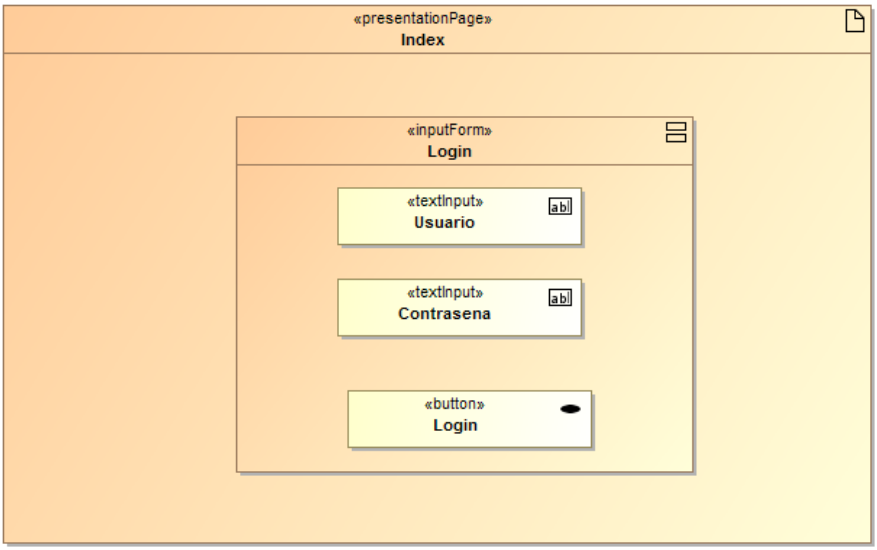

Figura 35 Modelo de presentación para Login

La siguiente figura (ver Figura 36) muestra la página principal después de iniciar la sesión, en esta se aprecia el menú principal de manera general.

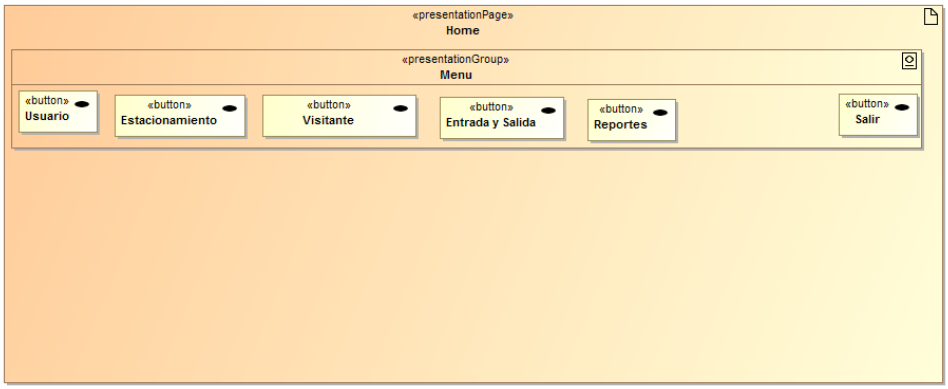

Figura 36 Modelo de presentación para página principal y menú

En el diseño correspondiente a la gestión de usuarios (ver Figura 37) se incluye el listado de usuarios y la interacción con diferentes acciones como registrar, editar y eliminar.

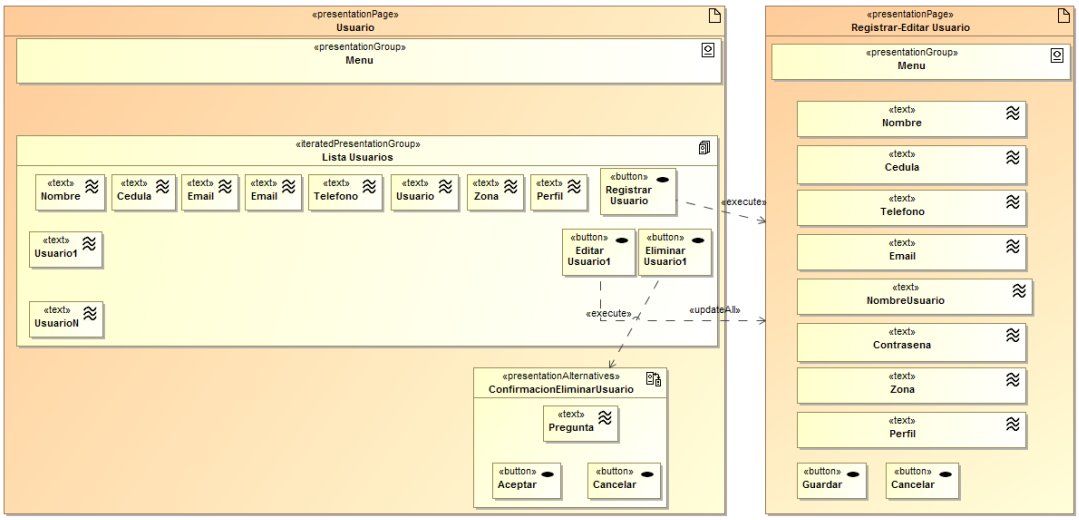

Figura 37 Modelo de presentación para usuarios

La gestión de estacionamientos fue representada en el siguiente diagrama (ver Figura 38) en el que se detalla listado, registro, edición y eliminación.

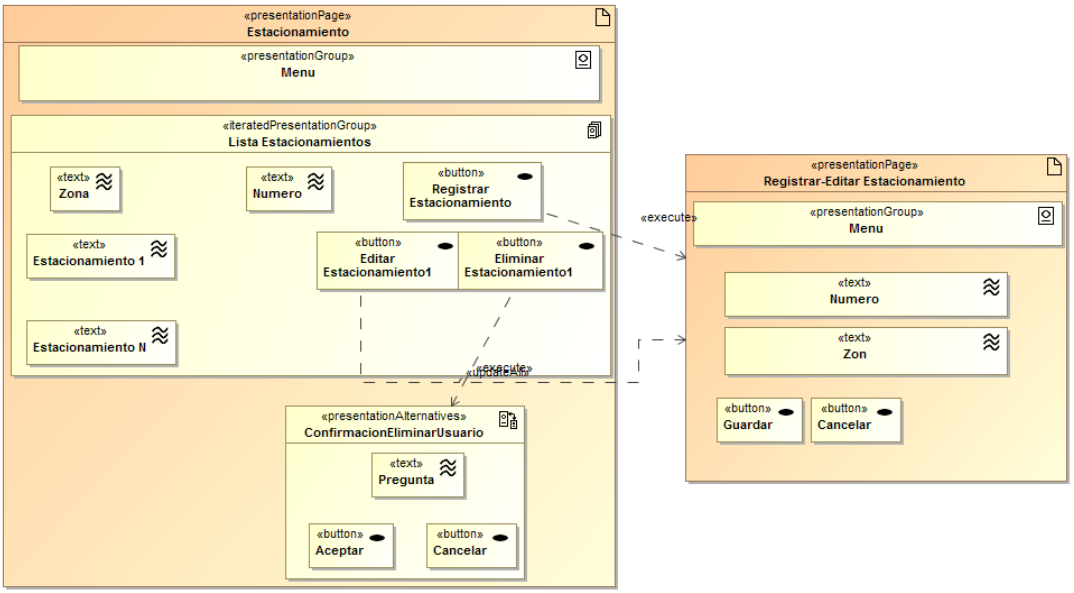

Figura 38 Modelo de presentación para estacionamientos

Las pantallas correspondientes al registro, edición, eliminación y muestra de información de visitantes se ilustran a continuación (ver Figura 39).

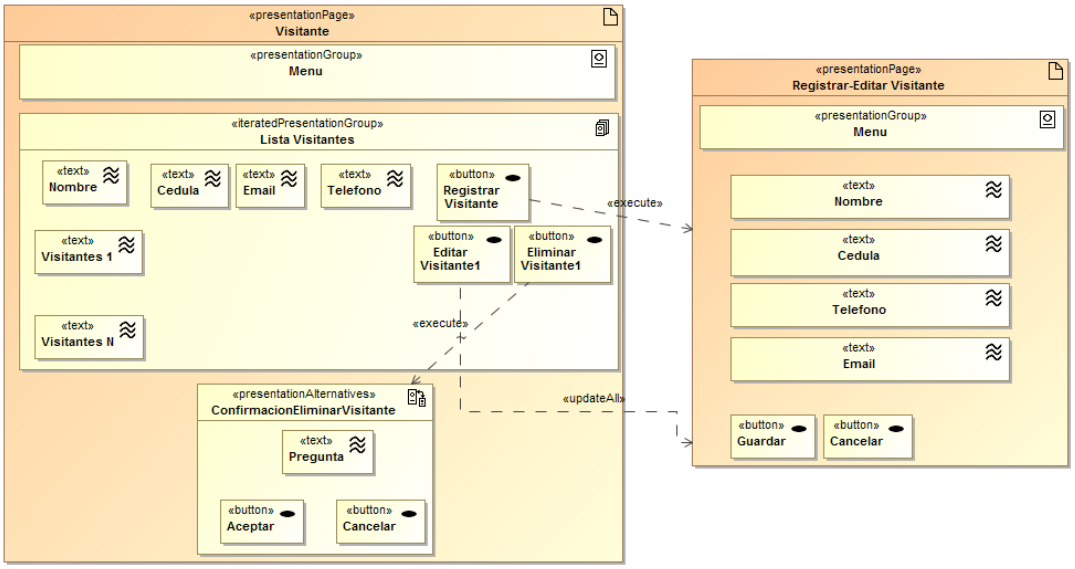

Figura 39 Modelo de presentación para visitantes

Finalmente, la vista para el registro de la entrada de un vehículo se representa en la siguiente figura (ver Figura 40), en esta se detalla la información del estacionamiento con su respectivo estado, la vista del registro de la entrada y la interacción al registrar la salida respectiva.

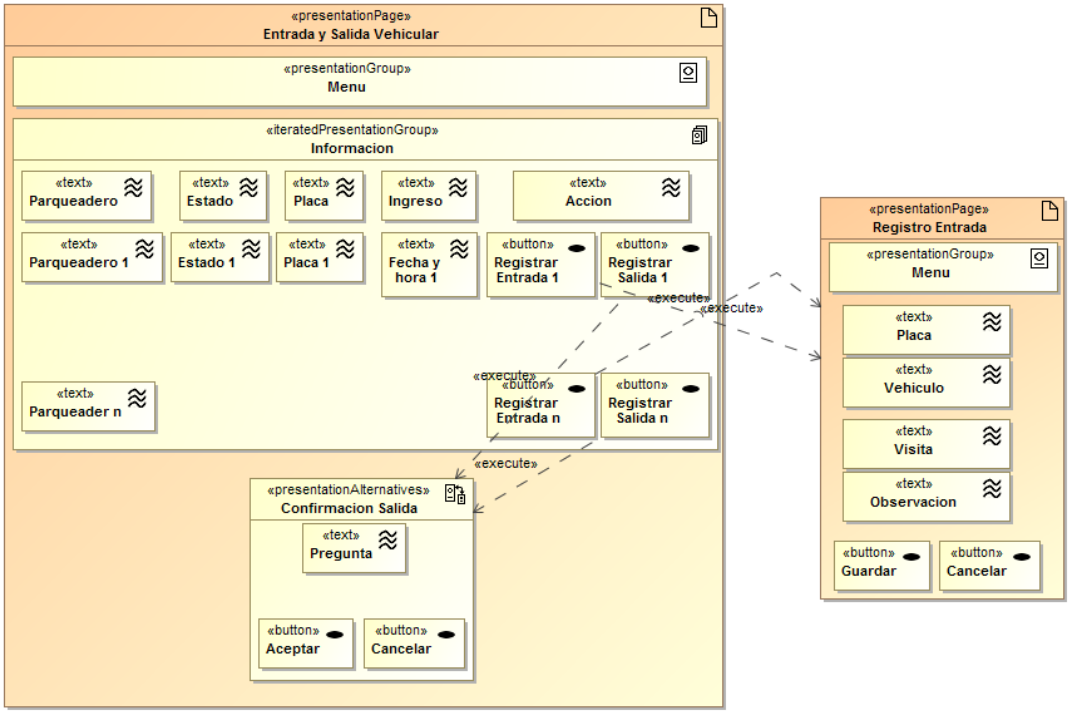

Figura 40 Modelo de presentación para entrada y salida de vehículos

3.6.2.4. Modelo de proceso

El modelo de proceso está dividido en dos partes el modelo de estructura de proceso en el cual se detalla operaciones básicas y el proceso principal (ver Figura 40) y los modelos de flujos de procesos los cuales especifican a mayor profundidad los procesos de estructura, estos modelos se observan en varias figuras.

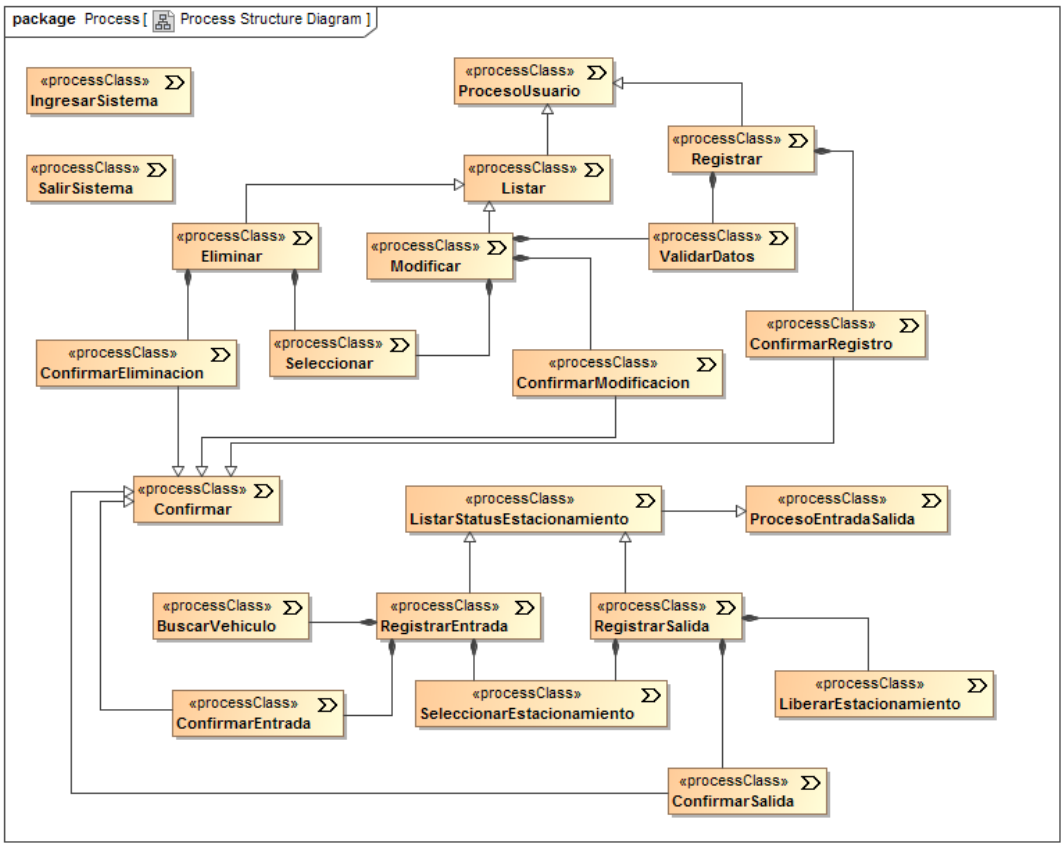

Figura 41 Modelo de estructura de proceso genérico del sistema

Los modelos de flujos de procesos se representan a partir de la siguiente figura (ver Figura 42), en esta se explica el proceso de Login o ingreso al sistema desde las acciones que se llevan a cabo hasta la página principal con su respectivo menú.

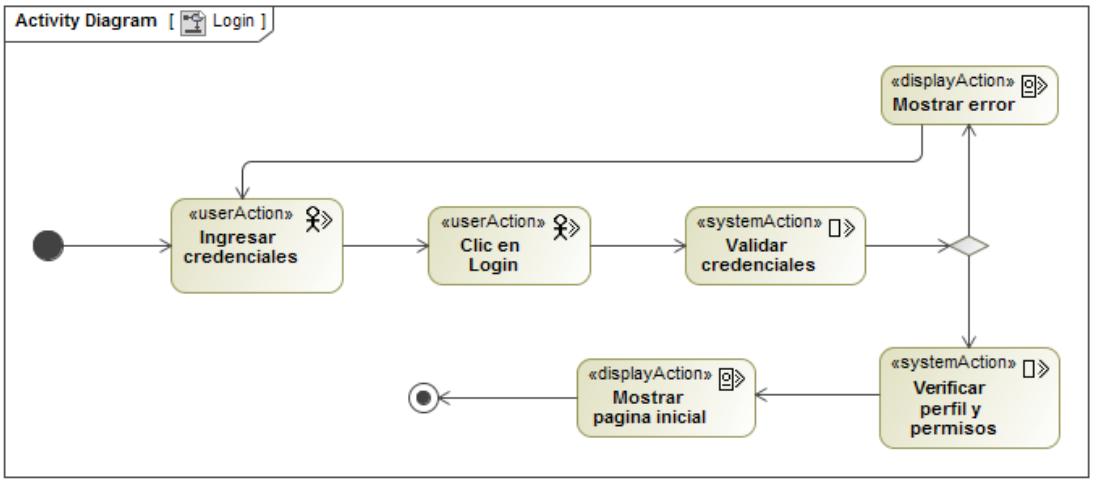

Figura 42 Modelo de flujo de proceso de Login

El siguiente flujo (ver Figura 43) detalla el proceso de registro de visitantes, usuarios y estacionamientos. Este proceso es similar en los casos mencionados y se ha considerado para su representación a usuarios.

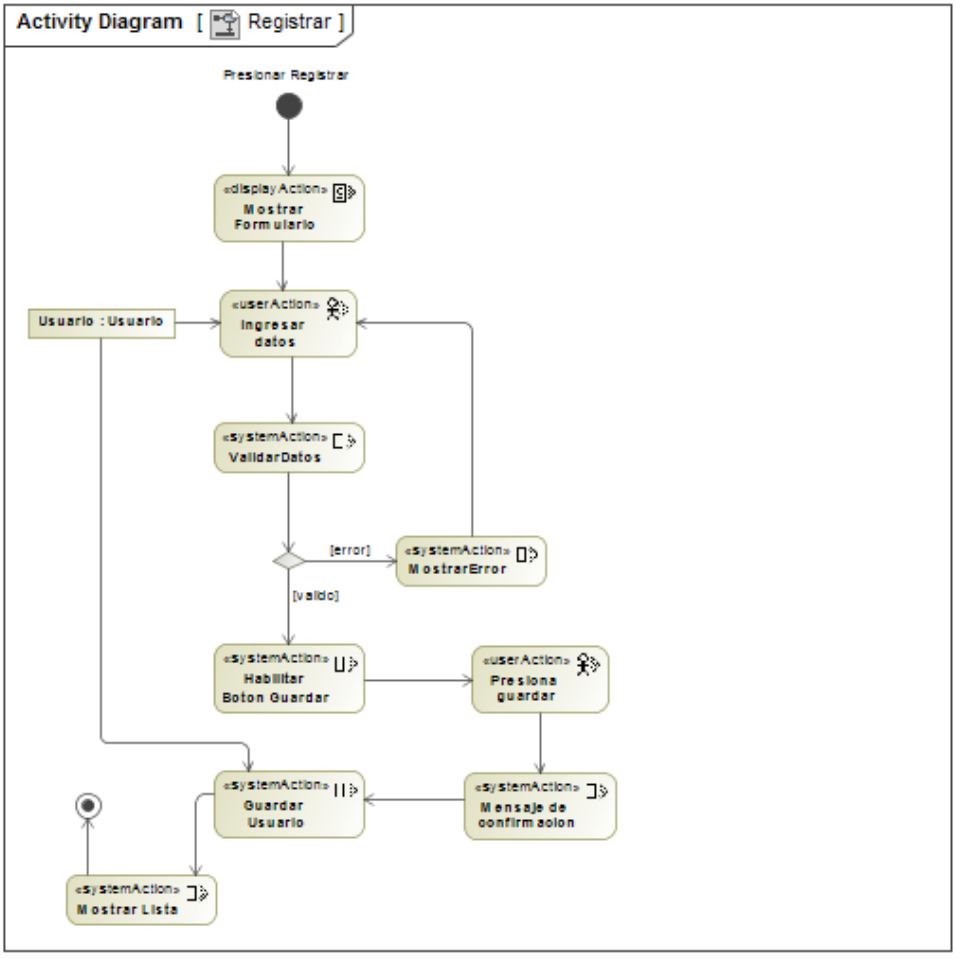

Figura 43 Modelo de flujo de proceso para registrar datos.

De igual manera que el flujo de proceso de registro el de edición está relacionado con la modificación de usuarios, visitantes y estacionamiento y se aprecia a continuación (ver Figura 44). En este diagrama se explica cómo realizar la edición desde la visualización de los datos correspondientes hasta la confirmación y guardado de los datos modificados.

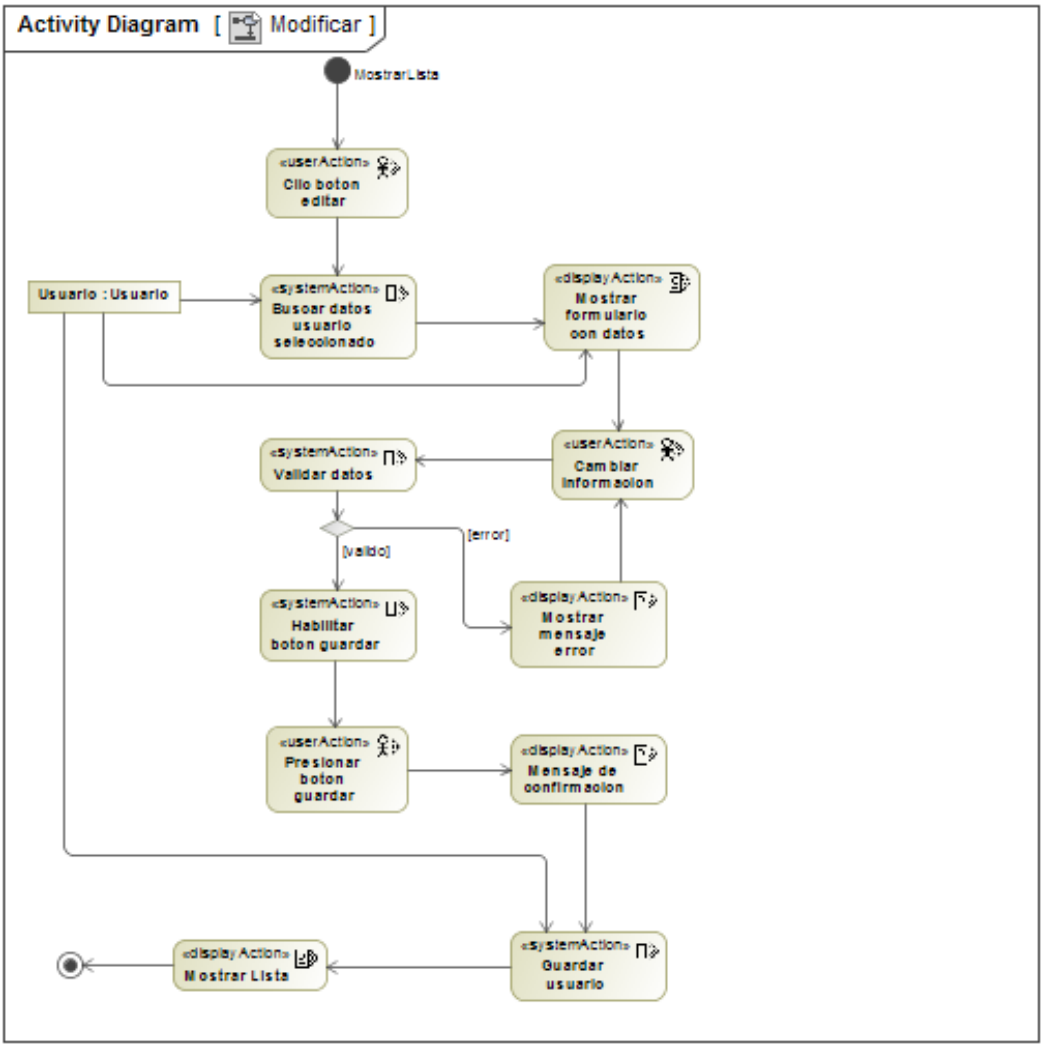

Figura 44 Modelo de flujo de proceso para modificar datos

Todo proceso “eliminar” trabaja de la misma manera y esta se refleja en el diagrama adecuado (ver Figura 45). Este flujo inicia en el listado de los datos y finaliza en aceptación y eliminación de los mismos.

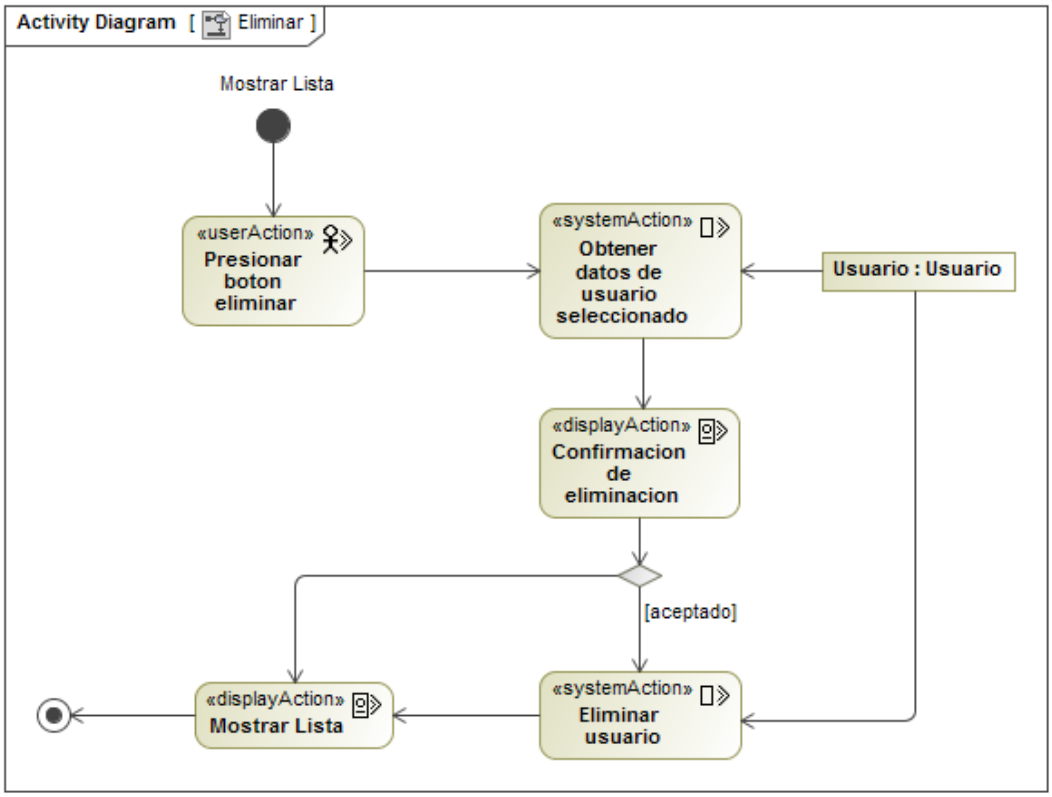

Figura 45 Modelo de flujo de proceso para eliminar datos

En el registro de una entrada vehicular se realiza la verificación de la placa correspondiente hasta la confirmación del ingreso realizado, todo este proceso se detalla en el flujo presentado (ver Figura 46).

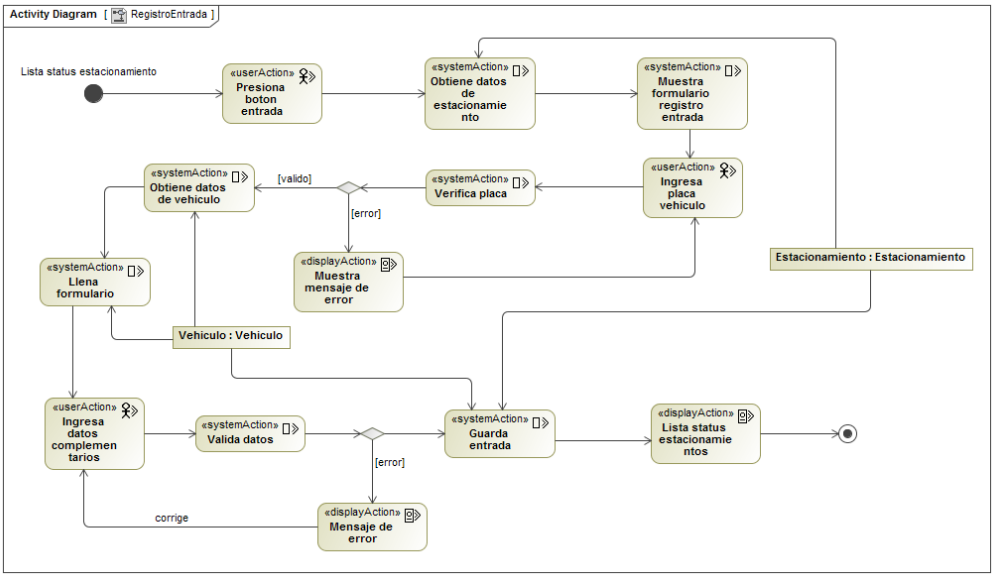

Figura 46 Modelo de flujo de proceso para registrar entrada de vehículos

Finalmente se muestra el flujo del proceso del registro de la salida de un vehículo (ver Figura 47), este diagrama indica las actividades realizadas para liberar un estacionamiento.

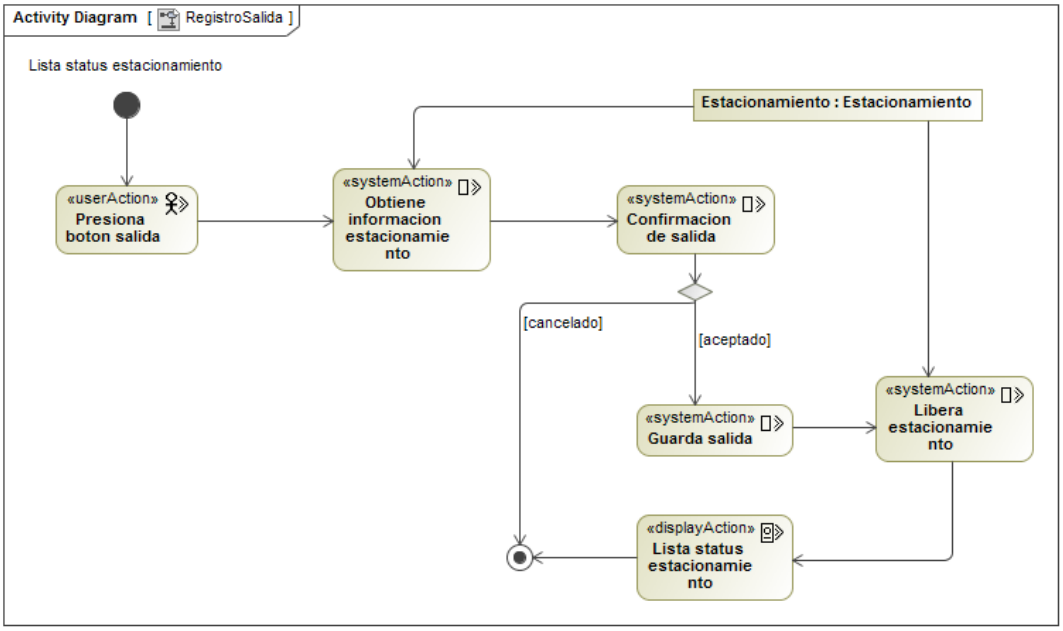

Figura 47 Modelo de flujo de proceso para registrar salida de vehículo

3.6.3. Implementación

La metodología utilizada sugiere que en los entregables de esta fase sean componentes del producto final por lo cual en este apartado se muestra herramientas que se han utilizado para llevar a cabo la implementación del caso de estudio considerado, el modelo físico de la base de datos construida y otros anexos.

3.6.3.1. Herramientas utilizadas

- MySQL: Se ha utilizado este motor de base de datos de código abierto para la gestión correspondiente.
- Netbeans: Este IDE de desarrollo se utilizó para codificar lo relacionado con Back-End ya que este se lo hizo en lenguaje Java.
- Visual Studio Code: Dada su buena compatibilidad con Angular 2, TypeScript y JavaScript se ha utilizado para implementar el Front-End de la aplicación.

- Glassfish: Como servidor de aplicaciones por su alta compatibilidad con el estándar empresarial que ofrece Java, dado que se implementó servicios REST que son consumidos por la parte Front-End.

- Node.js: Es un entorno de ejecución JavaScript el cual permite la instalación de todas las dependencias y herramientas de desarrollo requeridas para un proyecto realizado con Angular.

3.6.3.2. Modelo físico de base de datos

El modelo de base de datos utilizado para el desarrollo de la aplicación se lo visualiza a continuación (ver Figura 48), el script respectivo se encuentra en el documento adjunto en el Anexo D.

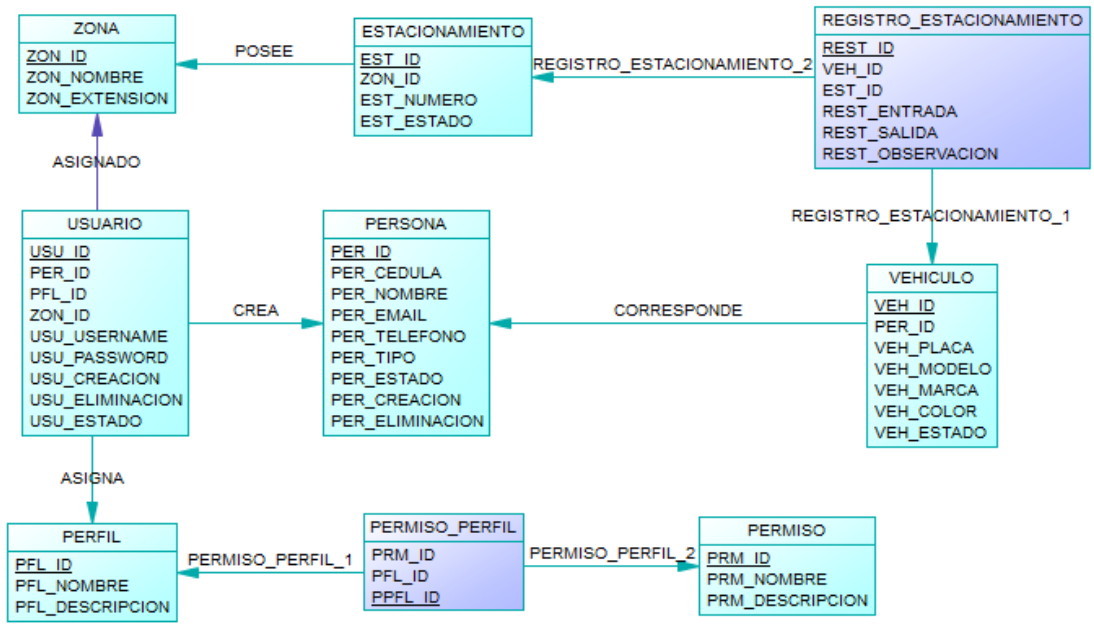

Figura 48 Modelo físico de base de datos del sistema

CAPÍTULO IV

VALIDACIÓN DE RESULTADOS COMPARATIVOS

4.1. Metodología de encuesta

Para realizar la validación se requiere de la opinión de diversos usuarios que posean la información correspondiente, es decir que tengan los conocimientos en un nivel intermedio a superior de la herramienta que se ha utilizado para el desarrollo de la aplicación, en este caso de Angular 2. La técnica utilizada para este procedimiento es la encuesta y la metodología que se ha aplicado para su desarrollo se explica a continuación.

En primer lugar, se utilizó como referencia los criterios tomados en cuenta en el análisis comparativo de las herramientas (ver sección 3.4.) para formular las preguntas, ya que la finalidad de la encuesta es validar que Angular 2 cumpla con los mismos. Cada pregunta corresponde a uno o más criterios si es el caso y un conjunto de hasta cinco preguntas evalúan las características principales cuyo análisis se muestra posteriormente y su detalle se encuentra adjunto en el Anexo C.

En cuanto a las valoraciones para cada una de las preguntas se ha considerado conveniente utilizar una escala de 5 niveles, con los siguientes valores iniciando desde la más alta con Excelente, Bueno, Regular, Malo y Ninguno como la más baja.

- **Excelente:** Los indicadores evaluados en la pregunta cumplen satisfactoriamente las expectativas del encuestado.
- **Bueno:** Indica que lo considerado en la pregunta cumple normalmente las expectativas.
- **Regular:** Este nivel especifica que la característica si se cumple pero no satisface la expectativa requerida.
- **Malo:** No cumple con el indicador evaluado a pesar que pueda existir indicios.

- **Ninguno:** Absolutamente no cumple este indicador sin existir ningún tipo de indicio o simplemente el participante no observado nada al respecto.

Se ha elegido esta escala para facilitar la evaluación y determinar valores que permitan realizar un análisis entre los resultados que proveen los participantes a través de la valoración de las 20 preguntas y los resultados obtenidos en el análisis anteriormente realizado (ver sección 3.5.).

La encuesta es aplicada una vez que se han seleccionado los participantes y se ha expuesto la aplicación la cual sirve como uno de los medios de presentación de varios de los criterios, siendo el otro medio complementario la experiencia que tiene el desarrollador participe en cuanto al manejo de la herramienta orientada a componentes web, Angular 2.

4.1.1. Determinar el número de participantes

La muestra se ha determinado utilizando como referencia estudios similares, relacionados con estudios comparativos de software y que han utilizado como técnica la encuesta y observación. En estos trabajos se ha establecido como participantes a aquellos que tienen algún conocimiento del desarrollo software.

Los estudios revisados son los siguientes (Jácome, 2016), (Jaramillo, 2013), (Molina, Loja, Zea, & Loaiza, 2016) los cuales indican un número de participantes de 20, 5 y 5 respectivamente. Bajo estos antecedentes el cálculo de la muestra utilizada en este proyecto se ha obtenido del promedio de estos valores referenciales el mismo que es equivalente a 10 individuos.

4.1.2. Resultados obtenidos

Primero se debe aclarar que el análisis realizado considera las características principales de los criterios evaluados, las cuales son funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad, recordando que estas características son procedentes de la norma ISO/IEC 9126 que se utilizó en el respectivo análisis de herramientas orientadas a componentes web.

Con esta aclaración, primero se procedió a calcular valores porcentuales mínimos y máximos en los cuales el rango que proveen determina la zona en la cual las características principales deben situarse para obtener una validación positiva. Para obtener este rango y valoración se consideró los puntajes máximos y obtenidos (ver Tabla 45) de las herramientas estudiadas teniendo como resultado los siguiente (ver Tabla 65).

Tabla 65

Porcentajes mínimos y máximos

	Máximo		Mínimo	
Característica	**Porcentaje [%]**	**Puntaje**	**Porcentaje [%]**	**Puntaje**
Funcionalidad	100,00	13	84,62	11
Fiabilidad	100,00	17	76,47	13
Usabilidad	100,00	16	93,75	15
Eficiencia	100,00	8	75,00	6
Mantenibilidad	100,00	13	100,00	13
Portabilidad	100,00	16	93,75	15

Cabe mencionar que el máximo puntaje que se puede obtener en cada característica es del ciento por ciento ya que se analizará el comportamiento por característica más no de manera global como se lo realizó en el capítulo 3.

Una vez obtenidos los resultados de las encuestas se procedió a agrupar las preguntas por característica principal; después se realizó la suma de los porcentajes que se encuentran valorados como Excelente y Bueno en cada pregunta; el siguiente paso es promediar dichas sumas en cada grupo, este resultado es el correspondiente a la característica y es el analizado posteriormente. Dados estos porcentajes y después del procedimiento correspondiente al análisis de datos de las encuestas los cuales se encuentran detallados en el Anexo C, los resultados finales se encuentran resumidos en la siguiente tabla (ver Tabla 66) y figura (ver Figura 48).

Tabla 66

Resumen de resultados

Característica	Encuesta [%]	Mínimo [%]	Diferencia [%]	Validación
Funcionalidad	85,0	84,62	0,38	Cumple
Fiabilidad	80,0	76,47	3,53	Cumple
Usabilidad	96,7	93,75	2,92	Cumple
Eficiencia	75,0	75,00	0,00	Cumple
Mantenibilidad	100,0	100,00	0,00	Cumple
Portabilidad	94,0	93,75	0,25	Cumple

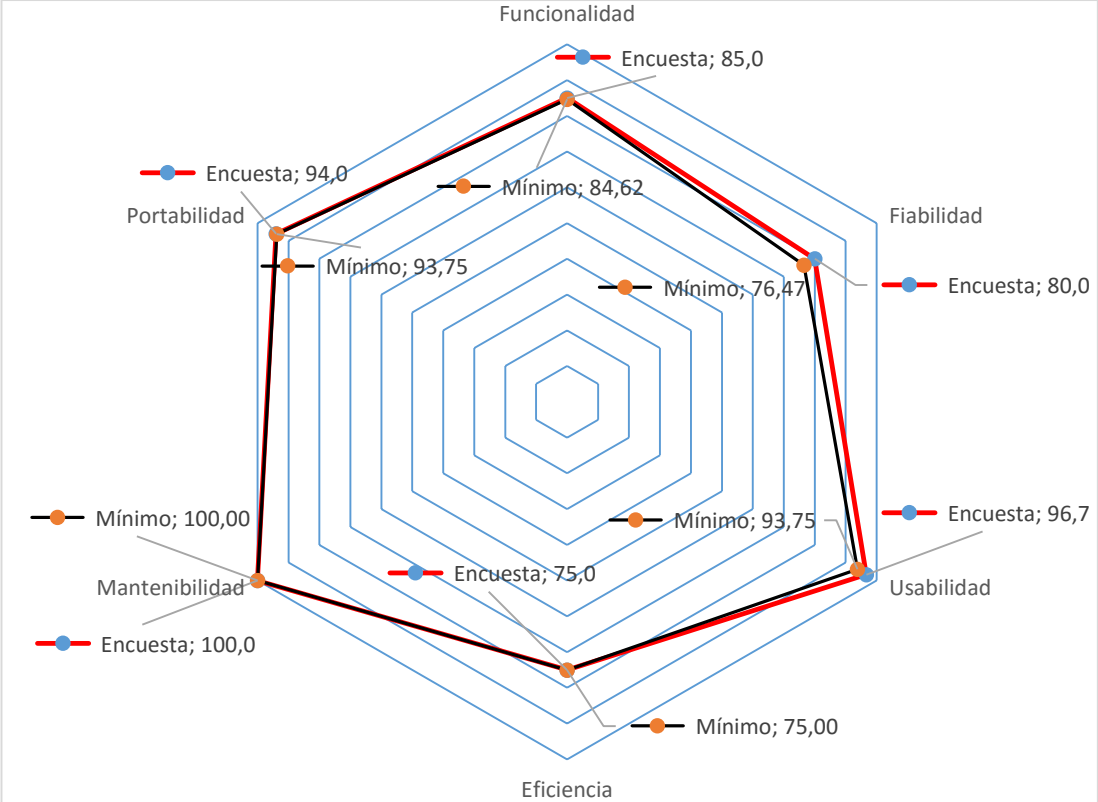

Figura 49 Encuesta vs Análisis

Como se puede apreciar las 6 características fundamentales evaluadas a través de los indicadores en cada pregunta son validadas en la encuesta planteada. Teniendo porcentajes que se encuentran dentro de los límites dados anteriormente, lo que permitió concluir que Angular 2 efectivamente cumple con los criterios considerados para realizar la evaluación y comparación.

Existen características como mantenibilidad y eficiencia cuyos porcentajes son iguales en ambos casos, de lo cual se concluyó basándose en la primera cualidad mencionada que Angular 2 es un Framework orientado a web components que brinda mantenibilidad a las aplicaciones desarrolladas con éste. En el segundo caso al observar que se alcanza el 75% se determinó que si cumple con los criterios de eficiencia sin embargo el nivel de cumplimiento es mínimo y esto corrobora lo obtenido en el análisis donde se obtuvo a React JS con mayor valoración en este aspecto. Las demás características se encuentran sobre el mínimo valor establecido esto indica que las expectativas de la herramienta son buenas en cuanto a funcionalidad, fiabilidad, usabilidad y portabilidad, además que valida que el estudio realizado y la herramienta seleccionada han sido los adecuados.

CAPÍTULO V

CONCLUSIONES Y LÍNEA DE TRABAJO FUTURO

5.1. Conclusiones

- Se definió un modelo de evaluación utilizando criterios de la norma ISO/IEC 9126. La inclusión del estándar como parte de la metodología para la evaluación de las herramientas fue indispensable, debido a que aportó con características y subcaracterísticas generales como punto de partida. Esto permitió tener una base y definiciones fundamentadas y comprensibles para el desarrollo del modelo.
- Realizar el análisis de las metodologías que permitan construir modelos de calidad fue fundamental ya que de esta manera se definió el tipo de modelo de evaluación, la compatibilidad con la norma establecida y el procedimiento que se debe seguir para la respectiva construcción.
- Con el estudio realizado a las metodologías para la construcción de modelos, se obtuvo como mejor opción IQMC. Este permitió identificar atributos de las subcaracterísticas dadas en la norma y métricas correspondientes. Esto conllevó elaborar un modelo con mayor precisión y flexibilidad para plasmarlo en una matriz.
- Se aplicó el modelo a las herramientas en estudio donde se obtuvo un puntaje muy cercano entre Angular 2 y React JS, sin embargo, la primera consiguió superar en cuatro de las seis características evaluadas obteniendo una ventaja de casi dos puntos.
- Angular 2 obtiene ventaja en usabilidad a su similar, debido a que supera en la capacidad de aprendizaje y en la facilidad de operación que este ofrece. Además, supera en otras características como el enlace de datos, la coexistencia con otros lenguajes, lo cual acumula puntos a su favor.
- El uso de la metodología de desarrollo web UWE proporcionó el procedimiento para modelar la aplicación brindando la información necesaria de análisis, diseño e implementación para mantener un proceso

ágil y eficaz lo cual complementado con el uso de la herramienta con orientación a componentes web permitió optimizar de mejor manera el tiempo de desarrollo.

- Se desarrolló la aplicación con Angular 2 como la herramienta seleccionada, donde se determinó que el uso de herramientas orientadas a componentes web agilita la implementación de aplicaciones, ya que provee facilidad en cuanto a la reutilización de algunos componentes tanto propios como creados por la comunidad correspondiente a dicha herramienta.

5.2. Línea de trabajo futuro

- Con la finalidad de complementar y profundizar la situación de cumplimiento de algunas características dadas en este proyecto, se propone realizar un benchmarking completamente práctico aplicando las herramientas Angular 2 y React JS a un caso práctico. Esto como una validación adicional del presente proyecto.
- Dado que se seleccionó Angular 2 como la mejor opción, se plantea elaborar un estudio adicional en el cual se utilice la arquitectura web conocida como MEAN Stack, que se refiere al uso de MongoDB, ExpressJS, Angular y NodeJS. Esto con la finalidad de medir su eficiencia en cuanto al desarrollo e implementación de sistema.

REFERENCIAS BIBLIOGRÁFICAS

Álvarez, M. Á. (02 de 11 de 2015). *Web Components*. Obtenido de Desarrolloweb.com: https://desarrolloweb.com/articulos/que-son-web-components.html

Azaustre, C. (25 de 06 de 2015). *Consumiendo un API REST desde React.js con ECMAScript6*. Obtenido de https://carlosazaustre.es/blog/consumiendo-un-api-rest-desde-react-js-con-ecmascript6/

Azaustre, C. (15 de 01 de 2016). *Web Components: presente y futuro en el desarrollo web*. Obtenido de bbvaopen4u.com: https://bbvaopen4u.com/es/actualidad/web-components-presente-y-futuro-en-el-desarrollo-web

Baigorri, A. (31 de 05 de 2016). *ReactJS y ¿sus víctimas?* Obtenido de itblogsogeti.com: https://itblogsogeti.com/2016/05/31/reactjs-y-sus-victimas/

Bermeo, I. (08 de 2014). Análisis comparativo de frameworks JavaScript: JQuery y MooTools, para la implementación de aplicaciones web en la empresa SOFYA. Aplicación a un caso de estudio. Quito, Ecuador.

Calero, C., Moraga, M. A., & Piattini, M. G. (2010). *Calidad del producto y proceso software.* Madrid: RA-MA.

Carvallo, J. P. (28 de 07 de 2013). *Introducción a los modelos de calidad del software.* Obtenido de es.slideshare.net: https://es.slideshare.net/JuanPabloCarvallo/4-introduccion-a-los-modelos-de-calidad

Christodolou, S., & Gizas, A. (2012). *Performance evaluation framework of all classes of selectors fro JavaScript libraries.* Patras, Grecia.

Cleveroad. (17 de 02 de 2017). *React vs Angular: Two sides of JavaScript*. Obtenido de cleveroad.com: https://www.cleveroad.com/blog/react-vs-angular-ultimate-performance-research-2017

CloudExperto. (18 de 05 de 2017). *Capítulo 3: Arquitectura de una aplicación Angular*. Obtenido de cloudexperto.com: https://www.cloudexperto.com/academia/angular2/capitulo-3-arquitectura-de-una-aplicacion-angular/

Covella, G. (11 de 2005). Medición y Evaluación de Calidad en Uso de Aplicaciones Web. La Plata, Argentina.

Djouab, R., & Bari, M. (5 de 05 de 2016). An ISO 9126 Based Quality Model for the e-Learning. Montreal, Canadá.

Eizagirre, M. (08 de 12 de 2016). *Introducción a Angular-CLI*. Obtenido de blog.irontec.com: https://blog.irontec.com/introduccion-a-angular-cli/

Enge, A. (19 de 06 de 2014). *Testing React Components*. Obtenido de www.asbjornenge.com: http://www.asbjornenge.com/wwc/testing_react_components.html

Facebook Inc. (2017). *React: A JavaScript library for building user interfaces*. Obtenido de facebook.github.io: https://facebook.github.io/react/

Gaytan, V. (2014). *Calidad de producto.* Obtenido de slideplayer.es: http://slideplayer.es/slide/320773/

Gonzáles, A., André, M., & Hernández, A. (12 de 2015). Análisis comparativo de modelos y estándares para evaluar la calidad del producto software. La Habana, Cuba.

Google. (2017). *Angular: Architecture Overview*. Obtenido de angular.io: https://angular.io/guide/architecture

International Organization for Standarization. (2000). *ISO 9126-1: Information Technology - Software product quality. Part 1: Quality model.*

ISO. (15 de 09 de 2015). ISO 9000: Sistemas de gestión de la calidad. - Fundamentos y vocabulario.

Jácome, P. (2016). *Benchmarckin de los Frameworks opensource: Bootstrap y Uikit.* Ibarra.

Jaramillo, M. (2013). *Análisis comparativo entre los Frameworks MyFaces, IceFaces y RichFaces aplicado al sistema nutricional de la ESPOCH.* Riobamba.

Koch, N. (2001). *Software Engineering for Adaptive Hypermedia Systems.* Munich.

Kuenzi, M. (26 de 05 de 2017). *Google IO 2017: Future, Faster: Unlock the Power of Web Components with Polymer*. Obtenido de incrementalevolution.com: http://incrementalevolution.com/google-io-2017-future-faster/

Kumar, K., Kumar, P., & Mathur, J. (04 de 2014). Implementation of a Model for Websites Quality Evaluation – DU Website. Meerut, India.

Mariano, C. L. (2017). *Benchmarking JavaScript Frameworks.* Dublin: Dublin Institue of Technology.

Martin, R., & Shafer, L. (07 de 1996). Providing a Framework for Effective Software Quality Assessment. Bedford, Massachusetts, Estados Unidos.

Molina, J., Loja, N., Zea, M., & Loaiza, E. (2016). *Evaluación de los Frameworks en el desarrollo de aplicaciones web con Python.* Machala.

Morales, C. (08 de 10 de 2014). *Introducción a Web Components: ¡El HTML ha muerto, larga vida al HTML!* Obtenido de octuweb.com: http://octuweb.com/introduccion-web-components/

Munguía, E. (19 de Enero de 2016). *Comparación React JS vs Angular 2*. Obtenido de enrique7mc.com: http://www.enrique7mc.com/2016/01/comparacion-react-js-vs-angular-2/

Murillo, R. (s.f.). *Calidad del software.* Obtenido de Calidad del software: https://regimurillo.jimdo.com/m%C3%B3dulos/unidad-3-modelos/modelo-de-boehm/

Nieves, C., Ucán, J., & Menéndez, V. (2014). *UWE en Sistema de Recomendación de Objetos de Aprendizaje. Aplicando Ingeniería Web: Un Método en Caso de Estudio.* Yucatán.

Nolivos, G., Coronel, F., Salvador, S., & Campaña, M. (2013). *Implementación de un sistema web para el control de un taller técnico automotriz en plataforma PHP - MYSQL utilizando UWE para la empresa Metroautocefran CIA. LTDA.* Sangolquí.

Olsina, L. (11 de 1999). Metodología Cuantitativa para la Evaluación y Comparación de la Calidad de Sitios Web. La Plata, Argentina.

Olsina, L. (1999). Web-site Quality Evaluation Method: a Case Study on Museums. La Plata, Argentina.

Oriol, E. (17 de 03 de 2017). *Intro a Angular (parte I): Modulo, Componente, Template y Metadatos*. Obtenido de blog.enriqueoriol.com: http://blog.enriqueoriol.com/2017/03/introduccion-angular-modulo-y-componente.html

Ortega, E. (2010). *Estudio de aplicabilidad y comparativo de un modelo de calidad a productos de software con la norma ISO/IEC 9126.* Guayaquil.

Pantoja, L., & Pardo, C. (2015). *Evaluando la facilidad de aprendizaje de Frameworks MVC en el desarrollo de aplicaciones web.* Popayán.

Paradigma Digital. (24 de 11 de 2014). *HTML5 Web Components.* Obtenido de es.slideshare.net: https://es.slideshare.net/paradigmatecnologico/html5-web-components

Pressman, R. (2010). *Ingeniería del Software un enfoque práctico.* México: McGrawHill.

Quingaluisa, E. (2016). *Aplicación web para la gestión de soporte y garantía técnica de equipo informáticos en la empresa Ecuatech de la ciudad de Salcedo.* Latacunga.

Scalone, F. (06 de 2006). Estudio comparativo de los modelos y estandares de calidad del software. Buenos Aires, Argentina.

Solano, J. R. (17 de Mayo de 2014). *Norma ISO/IEC 9126.* Obtenido de https://prezi.com/kuzz9h7tkpjz/norma-isoiec-9126/

Sommerville, I. (2005). *Ingeniería del Software.* Madrid: Pearson Edución.

Sousa, G. d. (19 de 01 de 2014). *Frameworks*. Obtenido de https://es.slideshare.net/GeraldynDeSousa/framework-30197256

VECTOR ITC GROUP. (02 de 09 de 2015). *Cómo elegir el mejor framework para cada proyecto tecnológico*. Obtenido de http://www.vectoritcgroup.com/es/como-elegir-el-mejor-framework-para-cada-proyecto-tecnologico

Vega, V., Gasca, G., & Echeverry, J. (2012). Analisis comparativo de modelos de calidad. Identificación de mejores prácticas para la gestión de calidad en pequeños entornos. Medellín, Colombia.

Vélez, J. (29 de 11 de 2015). *La Web Orientada a Componentes.* Obtenido de es.slideshare.net: https://es.slideshare.net/jvelez77/orientando-a-componentes-la-web-55613507

W3C. (12 de 02 de 2017). *Custom Elements*. Obtenido de w3c.github.io: https://w3c.github.io/webcomponents/spec/custom/#custom-elements

Webcomponents.org. (2014). *Web Components: Introduction*. Obtenido de https://www.webcomponents.org/introduction

Zeiss, B., Vega, D., Schieferddecker, I., Neukirchen, H., & Grabowski, J. (2007). *Applying the ISO 9126 Quality Model to Test Specifications.* Berlin.