DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN (SISTEMAS E INFORMATICA)

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

AUTOR: AÑAZCO MARCILLO VICTOR VLADIMIR

TEMA: ANÁLISIS COMPARATIVO ENTRE FRAMEWORKS: IONIC2 Y REACT, PARA EL DESARROLLO DE APLICACIONES MÓVILES EN LA EMPRESA SOFYA SYSTEMS S.A., APLICADO A UN CASO DE ESTUDIO.

DIRECTOR: ING. VILLACIS, CÉSAR

SANGOLQUÍ 2017

CERTIFICACIÓN

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

CERTIFICACIÓN

Certifico que el trabajo de titulación, "**ANÁLISIS COMPARATIVO ENTRE FRAMEWORKS: IONIC2 Y REACT, PARA EL DESARROLLO DE APLICACIONES MÓVILES EN LA EMPRESA SOFYA SYSTEMS S.A., APLICADO A UN CASO DE ESTUDIO**" realizado por el señor **VÍCTOR VLADIMIR AÑAZCO MARCILLO**, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar al señor **VÍCTOR VLADIMIR AÑAZCO MARCILLO** para que lo sustente públicamente.

Sangolquí, 23 de agosto 2017

Atentamente,

Ing. César Villacís
DIRECTOR

AUTORÍA DE RESPONSABILIDAD

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACION
CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

AUTORÍA DE RESPONSABILIDAD

Yo, **AÑAZCO MARCILLO VÍCTOR VLADIMIR**, con cédula de identidad N° 1725343782, declaro que este trabajo de titulación "**ANÁLISIS COMPARATIVO ENTRE FRAMEWORKS: IONIC2 Y REACT, PARA EL DESARROLLO DE APLICACIONES MÓVILES EN LA EMPRESA SOFYA SYSTEMS S.A., APLICADO A 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

Sangolquí, 23 de agosto 2017

AÑAZCO MARCILLO VÍCTOR VLADIMIR
1725343782

AUTORIZACIÓN

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACION
CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

AUTORIZACIÓN

Yo, **AÑAZCO MARCILLO VÍCTOR VLADIMIR**, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación "**ANÁLISIS COMPARATIVO ENTRE FRAMEWORKS: IONIC2 Y REACT, PARA EL DESARROLLO DE APLICACIONES MÓVILES EN LA EMPRESA SOFYA SYSTEMS S.A., APLICADO A UN CASO DE ESTUDIO**" cuyo contenido, ideas y criterios son de mi autoría y responsabilidad.

Sangolquí, 23 de agosto 2017

AÑAZCO MARCILLO VÍCTOR VLADIMIR
1725343782

DEDICATORIA

Dedico este trabajo a mi madre a mi padre y a mi hermano, quienes, con su paciencia, consejo y apoyo incondicional a través de los años, han sabido enseñarme como ser mejor y alcanzar las metas que me propongo, de no ser por ellos nada de esto sería posible.

Víctor Vladimir Añazco Marcillo

AGRADECIMIENTO

Primero, quiero agradecer a mi familia por su apoyo incondicional y su paciencia a lo largo de todo este trayecto.

A todos mis amigos que siempre estuvieron animándome y apoyándome para poder concluir con este proyecto.

A mi novia quien con su comprensión y optimismo me ayudó a ver el lado positivo de las cosas y sobre todo estuvo presente en los momentos de adversidad motivándome a alcanzar este logro.

Quisiera agradecer a todas las personas que me ayudaron para que esto sea posible ya sea con sus palabras o acciones que fueron de gran ayuda, se los agradezco de todo corazón.

Víctor Vladimir Añazco Marcillo

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Actualmente los navegadores internos de los dispositivos móviles permiten el uso de aplicaciones hibridas multiplataforma mismas que brindan una mejor experiencia y rendimiento. Para el desarrollo de estas aplicaciones es necesario el uso de un framework que sirva como base para la organización y el desarrollo de la mismas. SOFYA SYSTEMS S.A. es una empresa de tecnología dedicada al desarrollo de aplicaciones orientadas a la Web. Actualmente la organización está iniciando con las aplicaciones móviles y busca un framework que le permita desarrollar de manera ágil una aplicación móvil. Este proyecto se enfoca en la selección de un framework que se utilizará como soporte para la implementación de aplicaciones móviles en la empresa SOFYA SISTEMS. Para elegir un framework que brinde las mejores características se realizará un estudio comparativo, desarrollando un modelo de calidad, basado en la norma ISO/IEC 25000. Para verificar los beneficios del framework seleccionado en base al modelo creado, se desarrollará una aplicación móvil, utilizando una metodología de desarrollo combinada entre SCRUM y XP.

PALABRAS CLAVE:

- **APLICACIONES HIBRIDAS**
- **APLICACIÓN MÓVIL**
- **FRAMEWORK**
- **ESTUDIO COMPARATIVO**
- **SCRUM Y XP**

ABSTRACT

Currently the mobiles devices internals' browsers allow the use of hybrid multi-platform applications that provide better experience and performance. For the development of these applications it is necessary to use a framework that serves as a base for the organization and development of the same. SOFYA SYSTEMS S.A. Is a technology company dedicated to the development of applications oriented to the web. Currently the organization is starting with mobile applications and is looking for a framework that allows it to develop a mobile application in an agile way. This project focuses on the selection of a framework that will be used as support for the implementation of mobile applications in the company SOFYA SISTEMS. To choose a framework that provides the best characteristics, a comparative study will be carried out, developing a quality model based on the ISO / IEC 25000 standard. In order to verify the benefits of the selected framework based on the Model created, a mobile application will be developed, using a combined development methodology between SCRUM and XP.

KEYWORDS:

- **HYBRID APPLICATIONS**
- **MOBILE APP**
- **FRAMEWORK**
- **COMPARATIVE STUDY**
- **SCRUM AND XP**

CAPÍTULO 1

INTRODUCCIÓN

1.1 ANTECEDENTES

En la actualidad los navegadores internos de los dispositivos móviles están mejorando a tal punto que es posible el auge de aplicaciones híbridas multiplataforma, brindando una mejor experiencia y rendimiento como por ejemplo en aplicaciones nativas, con la gran ventaja de tener que implementarlas una sola vez. (Díaz Aguilera, 2016).

Los frameworks ayudan en la labor de desarrollar una aplicación al ser una base para la organización y el desarrollo de la misma. Existen diferentes tipos de framework que se ajustan a la necesidad o requerimiento que tenga un desarrollador específico, puesto que cada uno tiene sus características propias y presenta sus ventajas y desventajas. (Riehle, 2013).

SOFYA SYSTEMS S.A. es una empresa de tecnología dedicada al desarrollo de aplicaciones orientadas a la Web. Se estableció en el mercado el 04 de enero del 2013 y está ubicada en Cumbayá. Su objetivo es desarrollar software de calidad para satisfacer las necesidades de los clientes. (Acta constitución de la Empresa Sofya Systems S.A.).

Actualmente la organización está empezando con el desarrollo de aplicaciones móviles, pero, a pesar de que posee estándares para el desarrollo y está en constante mejora, la empresa SOFYA SYSTEMS S.A. no ha identificado un framework que permita desarrollar una aplicación móvil de forma ágil.

1.2 PLANTEAMIENTO DEL PROBLEMA

La empresa actualmente presenta dificultades en el desarrollo de aplicaciones móviles, como: se crea componentes que ya existen en frameworks, la compatibilidad de las aplicaciones con distintos dispositivos y el tiempo que conlleva realizar estas aplicaciones.

Esto provoca inconvenientes con los clientes al no poder asegurar un tiempo de entrega estimado, además de que genera una desventaja frente a la competencia porque no se cuenta con un framework de desarrollo y presenta posibles pérdidas económicas.

En base a la experiencia, el Director General de la empresa, considera que los problemas mencionados se ocasionan debido a que no se cuenta con un framework para el desarrollo de aplicaciones móviles, además menciona que el cambio de herramientas para el desarrollo reduce la productividad del mismo. Es por ello que, es necesario tomar una decisión sobre cual herramienta se utilizará en el desarrollo.

Debido a lo mencionado anteriormente, se ha previsto realizar una evaluación y selección de un framework que brinde las mejores características para poder desarrollar aplicaciones móviles de una manera ágil.

1.3 JUSTIFICACIÓN

En vista de los problemas mencionados en el planteamiento del problema, se genera la necesidad de establecer un framework para el desarrollo de aplicaciones móviles dentro de la empresa, con el fin de solventar una necesidad planteada, además de poder agilizar y facilitar el desarrollo de estas aplicaciones.

1.4 OBJETIVOS

1.4.1 Objetivo General

Realizar un análisis comparativo de frameworks para el desarrollo de aplicaciones móviles aplicando un modelo de calidad, caso de estudio en empresa SOFYA SYSTEMS S.A.

1.4.2 Objetivos Específicos

- Construir un modelo de calidad de software, utilizando algunos de los atributos de la Norma ISO 25000.
- Realizar un análisis FODA de los frameworks resultantes del modelo.
- Seleccionar un framework para el desarrollo en base a un análisis realizado con el modelo de evaluación y la matriz FODA.
- Aplicar el framework seleccionado en el desarrollo de una aplicación móvil.
- Desarrollar el aplicativo utilizando una metodología de desarrollo mixta entre SCRUM y XP.

1.5 ALCANCE

Construcción de un modelo de evaluación, en base al cual se evaluarán frameworks, este modelo se construirá considerando los atributos de calidad que indica la Norma ISO 25000.

Con el fin de identificar el framework que brinde las mejores características tale como son: la adecuación funcional, la eficiencia de desempeño, la compatibilidad, la usabilidad, la fiabilidad, la seguridad, la mantenibilidad y finalmente la portabilidad para el desarrollo de aplicaciones móviles, se aplicará el modelo de evaluación construido.

La aplicación se implementará con soporte del framework seleccionado y constará de los siguientes módulos y funcionalidades:

- Visualización de contenidos únicos por sesión y generales para toda la aplicación.

- Sesiones de Entrenamiento en donde el usuario registrará su progreso y creará compromisos para avanzar a nuevas sesiones.

- Gestor de compromisos para la creación y modificación de los mismos.

- Verificación del estado de los compromisos planteados por el usuario y el porcentaje de cumplimiento de cada sesión de entrenamiento.

- Reinicio de progresos para empezar con el programa desde cero.

- Módulo Administrador que permita la visualización de usuarios registrados, sus avances y estado de sesiones.

El aplicativo no contemplará:

- Generación automática de compromisos.

CAPITULO 2

MARCO TEÓRICO

2.1 APLICACIONES HÍBRIDAS

Las aplicaciones híbridas generalmente son apps que están embebidas en el navegador web del dispositivo. Para desarrollarlas se utilizan frameworks basados en lenguajes de programación y diseño web (PHP, HTML, CSS y JS).

En este tipo de Apps el nivel de integración con el SO dependerá del framework de desarrollo utilizado y de qué tan abierto sea el SO del dispositivo, teniendo en cuenta que en cada uno de ellos se presentan diferentes ventajas e inconvenientes.

En la actualidad con esta opción se puede tener gran acceso al hardware del teléfono e incluso en algunos casos a las librerías del Sistema Operativo del mismo. (anibalgoicochea, 2013).

La ventaja de las aplicaciones híbridas es que solo se necesita implementar una sola vez, y en la mayoría de los casos la aplicación se ejecutará sin problemas en dispositivos móviles independientemente de su sistema operativo ya sea Android, iOS o Windows Phone. (programacion.net, 2016).

2.2 FRAMEWORKS PARA EL DESAROLLO DE APLICACIONES MOVILES HÍBIRDAS

En software un framework es un esquema que ayuda con el desarrollo y/o la implementación de una aplicación. Conocido también como una infraestructura digital un framework es una estructura conceptual y tecnológica con soporte definido, que cuenta con módulos concretos de

software que pueden servir como base en la organización y el desarrollo de aplicaciones.

Generalmente pueden contener soporte de programas, librerías, lenguajes, entre otras herramientas que ayudan a desarrollar y enlazar los diferentes componentes de un proyecto. (Riehle, 2013).

En la actualidad existe una gran variedad de frameworks para desarrollar aplicaciones móviles los mismos que cuentan con sus características propias, así como ventajas y desventajas.

A continuación, se lista algunos de los frameworks más utilizados en el desarrollo de aplicaciones híbridas.

- Famous
- Native Script
- IONIC
- React
- jQuery Mobile
- Framework 7
- Onsen UI

(programacion.net, 2016)

2.3 NORMA ISO/IEC 25000

“ISO/IEC 25000, o SQuaRE (System and Software Quality Requirements and Evaluation), es el conjunto de normas con el objetivo de la creación de un marco de trabajo común para la evaluación de la calidad del producto software” (ISO 25000, 2017).

“La familia ISO/IEC 25000 es el resultado de la mejora de normas anteriores, como las ISO/IEC 9126 (particularidades de un modelo de calidad del producto software), e ISO/IEC 14598 (proceso de evaluación de productos software). La norma ISO/IEC 25000 se encuentra formada por cinco divisiones”. (ISO 25000, 2017).

2.3.3 ISO/IEC 2500n División de Gestión de Calidad

“Aquí se especifican todos los modelos, términos y definiciones comunes referenciados por todas las otras normas de la familia 25000 arquitecturas de SQuaRE, la terminología de la familia, modelos de referencia” (ISO/IEC 25000).

“Establece los requisitos y orientaciones para gestionar la evaluación y especificación de los requisitos del producto software” (ISO/IEC 25001).

2.3.4 ISO/IEC 2501n División de Medición de Calidad

Presenta modelos de calidad detallados que incluyen características en la calidad interna, externa y para uso del producto software.

“Describe el modelo de calidad para el producto software y para la calidad en uso, presenta características de calidad en base a las cuales se evaluará el producto software” (ISO/IEC 25010).

“Define un modelo general para la calidad de los datos, aplicable a aquellos datos que se encuentran almacenados de manera estructurada y forman parte de un Sistema de Información” (ISO/IEC 25012).

2.3.5 ISO/IEC 2502n División de Medición de Calidad

“Incluye un modelo de referencia de la medición de la calidad del producto, definiciones de medidas de calidad y guías prácticas para su aplicación.

Brinda una explicación introductoria y un modelo de referencia común a los elementos de medición de la calidad” (ISO/IEC 25020).

“Define y especifica un conjunto recomendado de métricas que puedan ser usadas a lo largo de todo el ciclo de vida del desarrollo software” (ISO/IEC 25021).

“Define específicamente las métricas para realizar la medición de la calidad en uso del producto” (ISO/IEC 25022).

“Define específicamente las métricas para realizar la medición de la calidad de productos y sistemas software” (ISO/IEC 25023).

“Define específicamente las métricas para realizar la medición de la calidad de datos” (ISO/IEC 25024).

2.3.6 ISO/IEC 2503n División de Requisitos de Calidad

“En este punto las normas ayudan a especificar requisitos de calidad que pueden ser utilizados en el proceso de elicitación de requisitos de calidad del producto software.

Provee de un conjunto de recomendaciones para realizar la especificación de los requisitos de calidad del producto software” (ISO/IEC 25030).

2.3.7 ISO/IEC 2504n División de Evaluación de Calidad

“Aquí se incluyen normas que proporcionan requisitos, recomendaciones y guías para llevar a cabo el proceso de evaluación del producto software.

Propone un modelo de referencia general para la evaluación, que considera las entradas, restricciones y recursos necesarios para obtener las salidas en el proceso de evaluación” (ISO/IEC 25040).

“Describe los requisitos y recomendaciones para la implementación práctica de la evaluación del producto” (ISO/IEC 25041).

“Define un módulo de evaluación y la documentación, estructura y contenido que se debe utilizar a la hora de definir uno de estos módulos” (ISO/IEC 25042).

“Define un módulo para la evaluación de la sub característica Recuperabilidad” (ISO/IEC 25045).

(ISO 25000, 2017)

2.4 ISO/IEC 25010

Según la ISO 25000: “un modelo de calidad representa el eje en torno al cual se establece el sistema para la evaluación de la calidad del producto.

En este modelo se van determinar las características de calidad que serán tomadas en cuenta al momento de evaluar las propiedades de un producto software determinado.

La calidad del producto software puede ser interpretada como el grado en que un producto satisface los requisitos de los usuarios agregando un valor.

Estos requisitos están representados en el modelo de calidad, el cual categoriza la calidad del producto en características y sub características, este modelo de calidad del producto definido por la ISO/IEC 25010 está conformado por ocho características de calidad" que se muestran en la

Figura 1 Características de calidad ISO 25000

Fuente:

CARACTERÍSTICAS DE CALIDAD ISO 25000

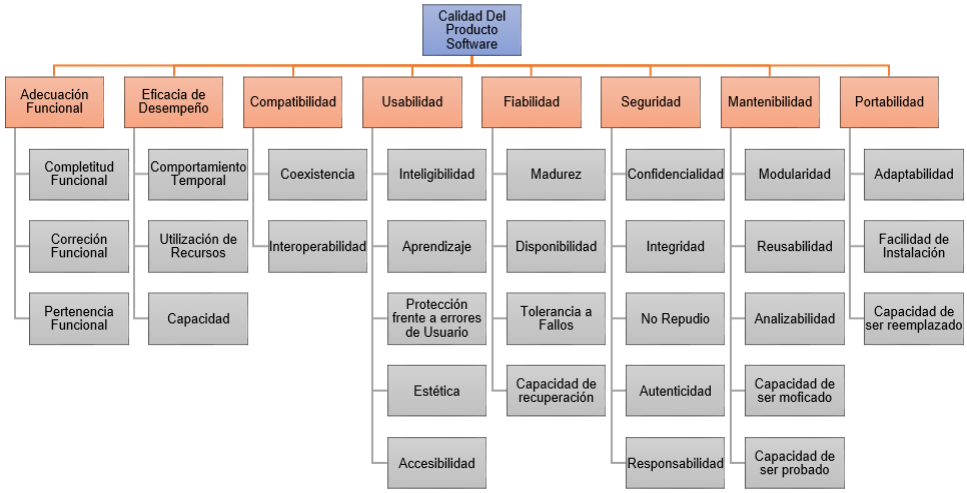

Figura 1 Características de calidad ISO 25000

Fuente: (ISO 25000, 2017)

2.4.1 Adecuación funcional (funcionalidad)

Representa la capacidad que tiene un producto software para proporcionar funciones que satisfacen las necesidades, cuando el producto se usa en las condiciones especificadas. Según (ISO 25010, 2017) está dividida en las siguientes subcategorías:

Completitud funcional. Es el grado en el cual el conjunto de funcionalidades cubre todas las tareas y los objetivos especificados por el usuario.

Corrección funcional. Es la capacidad del producto o sistema para brindar resultados correctos con el nivel de precisión requerido.

Pertinencia funcional. Representa la capacidad del producto software para proporcionar un conjunto apropiado de funciones para las tareas y objetivos especificados por un usuario.

2.4.2 Eficiencia de desempeño (Rendimiento)

Esta característica representa el desempeño relativo a la cantidad de recursos utilizados bajo determinadas condiciones. Según (ISO 25010, 2017) está dividida en las siguientes subcategorías:

“Comportamiento temporal. Los tiempos de respuesta y procesamiento de un sistema cuando lleva a cabo sus funciones bajo condiciones determinadas.

Utilización de recursos. Las cantidades y tipos de recursos utilizados cuando el software lleva a cabo su función bajo condiciones determinadas.

Capacidad. Grado en que los límites máximos de un parámetro de un producto o sistema software cumplen con los requisitos” (ISO 25010, 2017).

2.4.3 Compatibilidad

Es la capacidad de dos o más sistemas o componentes para intercambiar información y/o llevar a cabo sus funciones requeridas cuando comparten el mismo entorno hardware o software. Según (ISO 25010, 2017) está dividida en las siguientes subcategorías:

“Coexistencia. Capacidad del producto para coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes sin detrimento.

Interoperabilidad. Capacidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada" (ISO 25010, 2017).

2.4.4 Usabilidad

Es la capacidad del producto software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones. Según (ISO 25010, 2017) está dividida en las siguientes subcategorías:

"Capacidad para reconocer su adecuación. Permite al usuario entender si el software es adecuado para sus necesidades.

Capacidad de aprendizaje. Permite al usuario aprender su aplicación.

Capacidad para ser usado. Capacidad del producto que permite al usuario operarlo y controlarlo con facilidad.

Protección contra errores de usuario. Capacidad del sistema para proteger a los usuarios de cometer errores.

Estética de la interfaz de usuario. Capacidad de la interfaz de usuario de agradar y satisfacer la interacción con el usuario.

Accesibilidad. Capacidad del producto que permite el ser utilizado por los usuarios con determinadas características y discapacidades" (ISO 25010, 2017).

2.4.5 Fiabilidad

La capacidad de un sistema o componente para desempeñar funciones especificadas, cuando se usa bajo condiciones específicas y un periodo de tiempo determinados. Según (ISO 25010, 2017) está dividida en las siguientes subcategorías:

“Madurez. Capacidad del sistema para satisfacer las necesidades de fiabilidad en condiciones normales.

Disponibilidad. Capacidad del sistema o componente de estar operativo y accesible para su uso cuando se requiere.

Tolerancia a fallos. Capacidad del sistema o componente para operar según lo previsto en presencia de fallos hardware o software.

Capacidad de recuperación. Capacidad del producto software para recuperar datos afectados y reestablecer un estado en el sistema en caso de interrupción o fallo” (ISO 25010, 2017).

2.4.6 Seguridad

Capacidad de protección de la información y datos de manera que personas o sistemas no autorizados no puedan leerlos o modificarlos. Según (ISO 25010, 2017) está dividida en las siguientes subcategorías:

“Confidencialidad. Es la protección que se tiene en contra del acceso a los datos y/o información por no autorizados.

Integridad. Prevenir accesos o modificaciones no autorizados a datos o programas de ordenador.

No repudio. Capacidad de demostrar las acciones o eventos que han tenido lugar, de manera que dichas acciones o eventos no puedan ser repudiados posteriormente.

Responsabilidad. Capacidad de rastrear de forma inequívoca las acciones de una entidad.

Autenticidad. Capacidad de demostrar la identidad de un sujeto o un recurso” (ISO 25010, 2017).

2.4.7 Mantenibilidad

Esta característica representa la capacidad del producto software para ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas. Según (ISO 25010, 2017) está dividida en las siguientes subcategorías:

“Modularidad. Capacidad de un sistema o programa que permite que un cambio en un componente tenga un impacto mínimo en los demás.

Reusabilidad. Capacidad que permite que sea utilizado en más de un sistema software o en la construcción de otros.

Analizabilidad. Facilidad con la que se puede evaluar el impacto de un determinado cambio sobre el resto del software, diagnosticar las deficiencias o causas de fallos en el software, o identificar las partes a modificar.

Capacidad para ser modificado. Capacidad del producto que permite que sea modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño.

Capacidad para ser probado. Facilidad con la que se pueden establecer criterios de prueba para un sistema o componente y con la que se pueden llevar a cabo las pruebas para determinar si se cumplen dichos criterios” (ISO 25010, 2017).

2.4.8 Portabilidad

Capacidad del producto o componente de ser transferido de forma efectiva y eficiente de un entorno hardware, software, operacional o de utilización a otro. Según (ISO 25010, 2017) está dividida en las siguientes subcategorías:

“Adaptabilidad. Capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso.

Capacidad para ser instalado. Facilidad con la que el producto se puede instalar y/o desinstalar de forma exitosa en un determinado entorno.

Capacidad para ser reemplazado. Capacidad del producto para ser utilizado en lugar de otro producto software determinado con el mismo propósito y en el mismo entorno” (ISO 25010, 2017).

2.5 Modelo de evaluación ISO

El estándar ISO 9126 propone un modelo de calidad que se divide en tres vistas: interior, exterior y de uso. Estas están compuestas por características, las que se dividen en sub características, y estas a su vez se componen de atributos. El estándar está dividido en cuatro partes las cuales dirigen, respectivamente, lo siguiente:

- ISO 9126-1 Modelo de calidad (ISO/IEC, 2001)
- ISO 9126-2 Métricas externas (Mide el software en sí mismo)
- ISO 9126-3 Métricas de internas (mide el comportamiento del sistema)
- ISO 9126-4 Calidad en el uso de métricas (mide el efecto de usar el software en un contexto específico)

ESTÁNDAR ISO 9126

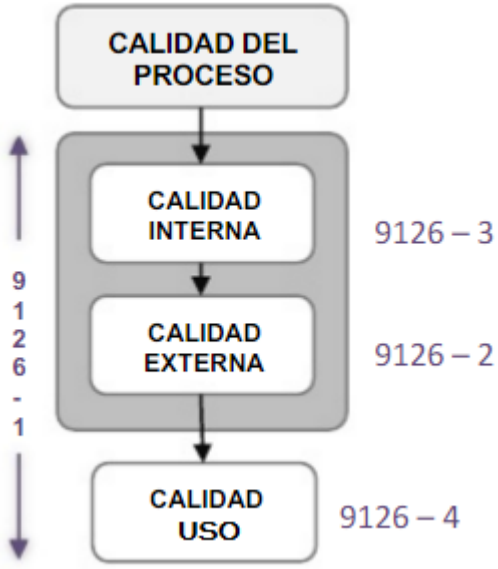

Figura 2 Estándar ISO 9126

Fuente: (Loaiza Morales, 2012)

2.5.1 CALIDAD INTERNA Y EXTERNA

2.5.1.1 Funcionalidad

En este grupo se encuentran una serie de atributos que permiten calificar si un producto de software maneja de manera adecuada un conjunto de funciones que satisfagan las necesidades para las cuales fue diseñado, establece los siguientes atributos:

2.5.1.2 Adecuación

Brindar un conjunto apropiado de funciones para realizar tareas y objetivos de usuario especificados.

2.5.1.3 Exactitud

Proporcionar resultados o efectos correctos, con un grado necesario de precisión.

2.5.1.4 Interoperabilidad

Interactuar con uno o más sistemas en específico.

2.5.1.5 Seguridad de acceso

Proteger la información y los datos de manera que las personas o sistemas no autorizados no puedan leerlos o modificarlos, mientras se permite el acceso a las personas o sistemas autorizados

2.5.1.6 Cumplimiento funcional

Adherirse a normas, convenciones o regulaciones en leyes y prescripciones similares relacionadas con funcionalidad.

2.5.1.7 Fiabilidad / Confiabilidad

En este punto se agrupan un conjunto de atributos que se refieren a la capacidad del software para mantener su nivel de ejecución bajo condiciones normales en un periodo de tiempo. Este estándar sugiere las siguientes características:

a) Madurez

Alcanzar un nivel que permita evitar fallar como resultado de anomalías en el software.

b) Tolerancia a fallos

Mantener un nivel específico de prestaciones en caso de fallos software.

c) Capacidad de recuperación:

Restablecer un nivel de prestaciones especificado y de recuperar los datos directamente afectados en un fallo.

d) Cumplimiento de la fiabilidad:

Adherirse a normas, convenciones o regulaciones relacionadas con la fiabilidad.

e) Usabilidad:

Abarca un conjunto de atributos que permiten evaluar el esfuerzo necesario que deberá invertir el usuario para manejar el sistema.

f) Capacidad para ser entendido

Permitir al usuario entender si el software es adecuado y cómo usarlo para realizar tareas.

g) Capacidad para ser aprendido

Permitir al usuario aprender sobre la aplicación.

h) Capacidad para ser operado

Permitir al usuario operar y controlar un aplicativo.

i) Capacidad de atracción

Capacidad del producto software para ser atractivo al usuario.

j) Cumplimiento de la usabilidad

Adherirse a normas, convenciones, guías de estilo o regulaciones relacionadas con la usabilidad.

2.5.1.8 Mantenibilidad

Contiene los atributos que permiten medir el esfuerzo para realizar modificaciones al software, ya sea por la corrección de errores o por aumento de necesidades.

a) Capacidad para ser analizado

Permitir diagnosticar deficiencias o causas de los fallos, además de identificar las partes que deben ser modificadas.

b) Capacidad para ser cambiado

Permitir que una determinada modificación sea realizada.

c) Estabilidad

Evitar efectos inesperados debidos a modificaciones.

d) Capacidad para ser probado

Capacidad del producto software que le permite ser probado luego de una modificación.

e) Cumplimiento de la mantenibilidad

Adherirse a normas o convenciones relacionadas con la mantenibilidad.

f) Portabilidad

Trata de la habilidad del software de ser transferido de un ambiente a otro, y considera los siguientes aspectos:

Adaptabilidad: Evalúa la oportunidad para adaptar el software a diferentes ambientes sin necesidad de aplicarle modificaciones.

Facilidad de Instalación: Es el esfuerzo necesario para instalar el software en un entorno determinado.

Conformidad: Permite evaluar si el software se adhiere a estándares o convenciones relativas a portabilidad.

Capacidad de reemplazo: Se refiere a la oportunidad y el esfuerzo empleado al sustituir un software por otro producto con funciones similares.

Capacidad de portabilidad: Capacidad del producto software para permitir que usuarios alcancen objetivos específicos en contextos de uso determinados.

2.5.1.9 Eficiencia

Permite que los usuarios alcancen objetivos con exactitud e integridad, en un contexto específico.

a) Comportamiento temporal

Consiste en proporcionar tiempos de respuesta, proceso y potencia apropiados, bajo condiciones determinadas.

b) Utilización de recursos

Es utilizar las cantidades y tipos de recursos adecuados cuando el software lleva a cabo su función bajo condiciones determinadas.

c) Cumplimiento de la eficiencia:

Cuando el software cumple con normas o convenciones relacionadas con la eficiencia.

2.5.2 CALIDAD EN USO

Para la vista en uso se contemplan cuatro características

CARACTERÍSTICAS DE VISTA EN USO

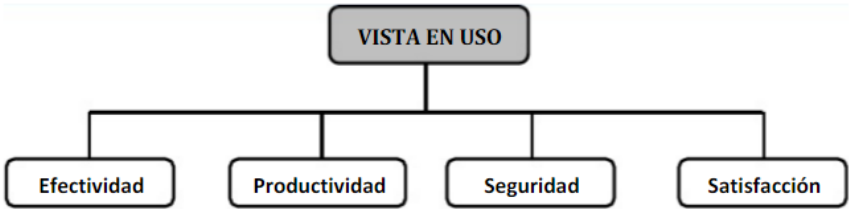

Figura 3 Características de Vista en Uso

Fuente: (Loaiza Morales, 2012)

2.5.2.1 Efectividad

Facilitar al usuario alcanzar objetivos con precisión y completitud.

2.5.2.2 Productividad

Permitir a los usuarios utilizar la cantidad apropiada de recursos en relación a la efectividad obtenida.

2.5.2.3 Seguridad

Cumplir con los niveles de riesgo emitidos tanto para posibles daños físicos como para posibles riesgos de datos.

2.5.2.4 Satisfacción

Cumplir con las expectativas de los usuarios en un contexto determinado.

2.6 FODA

FODA es el acróstico de: Fortalezas: (Son aquellos factores críticos positivos que se posee), Oportunidades: (Son aquellos aspectos positivos que se pueden aprovechar para utilizar nuestras fortalezas), Debilidades: (Son factores críticos negativos que se deberían reducir o eliminar de ser posible),

Amenazas: (Son aquellos aspectos negativos del exterior que podrían dificultar el cumplimiento de nuestros objetivos).

FODA es una herramienta para realizar un análisis que puede ser aplicado a cualquier situación, producto, empresa que sea un objeto de estudio en un momento determinado del tiempo.

El análisis FODA además permite diseñar un cuadro de la situación actual del objeto de estudio para obtener un diagnóstico preciso que facilita la toma de decisiones acordes con los objetivos y políticas formulados. (matrizfoda.com, 2017)

2.7 METODOLOGIAS AGILES

“En la última década el uso de metodologías ágiles de desarrollo de software se ha popularizado, debido a la rápida y constante evolución de la industria del software” (Pressman, 2010).

Pressman asevera que: “La ingeniería de software ágil representa una alternativa razonable a la ingeniería de software convencional para ciertas clases de software y en algunos tipos de proyectos. Asimismo, se ha demostrado que concluye con rapidez sistemas exitosos” (Pressman, 2010).

Para el desarrollo del proyecto en cuestión se eligió dos metodologías de desarrollo de software combinadas: Scrum, para la planificación del proyecto y Extreme Programming (XP) para la codificación del software.

2.7.1 Extreme Programming (XP)

Extreme Programming (XP) está definido como un método ágil para el desarrollo de software que es muy útil si se tiene que trabajar con proyectos que poseen requerimientos cambiantes o vagamente definidos, está basado en un conjunto de valores mismos que guían el desarrollo, los más destacados

son: comunicación, simplicidad, retroalimentación, coraje y respeto. Kent Beck (1999).

2.7.1.1 PROCESO XP

Según (Pressman, 2010) el proceso XP está compuesto por cuatro actividades:

- Planeación
- Diseño
- Codificación
- Pruebas

En la **Figura 4 Proceso XP**

Fuente: se muestra la secuencia de actividades que conforman el proceso de Extreme Programming.

PROCESO XP

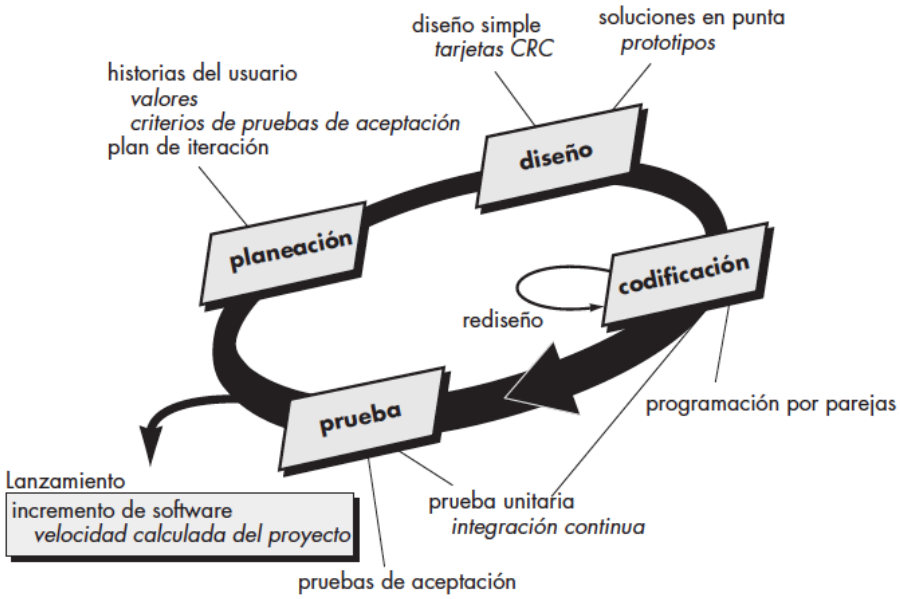

Figura 4 Proceso XP

Fuente: (Pressman, 2010)

2.7.1.2 ROLES

Según (Chromatic, 2013) Extreme Programming propone al menos cuatro roles, siendo el cliente y el desarrollador fundamentales dentro del ciclo de vida del proyecto.

a) Cliente

XP trata al cliente como una sola persona, quien define el proyecto y las metas que deben cumplirse en función al giro del negocio que se quiere atender.

b) Desarrollador

Es el encargado de llevar a cabo la mayoría de las prácticas que XP plantea. Se encarga de diseñar las pruebas unitarias y escribir el código.

c) Rastreador

Es quien da el seguimiento al proyecto en base a la velocidad de desarrollo del equipo. Esta medida se obtiene de la relación entre el tiempo estimado para la tarea y el tiempo real invertido en el desarrollo. Otros datos importantes que se pueden utilizar son, el sobretiempo trabajado y la relación entre las pruebas exitosas y las fallidas.

d) Coach o Tutor

Algunos proyectos que trabajan con XP, cuentan con un tutor, que es el encargado de ayudar al equipo a adoptar la metodología, ya sea sugiriendo o determinando la mejor manera de implementación de las prácticas.

2.7.1.3 PRACTICAS

Las prácticas son lo que determina la manera en que el equipo de desarrollo trabaja. La aceptación de una u otra práctica puede variar en función del contexto en el que se esté trabajando. (Álvarez García, de las Heras del Dedo, & Lasa Gómez, 2012).

a) Historias de usuario

Los requerimientos deben estar en lenguaje de cliente y ofrecer el mayor valor funcional posible a la aplicación.

b) Ciclos semanales

Propone atender los requerimientos del cliente en períodos de una semana.

c) Revisiones trimestrales

En escala de tiempo mayores, es oportuno analizar el estado del equipo, de los progresos y la continuidad de los objetivos.

d) Trabajar con holgura

Se trata de establecer tiempos de entrega realistas, tomando en cuenta posibles imprevistos.

e) Sentarse juntos

Para facilitar la comunicación entre los miembros del equipo de desarrollo, es importante sentar a todos en una misma sala y en un espacio abierto.

f) Equipo completo

El equipo debe estar compuesto por personas con los perfiles idóneos para abordar el desarrollo.

g) Información en el puesto de trabajo

Desde el puesto de trabajo debe poderse consultar el estado del proyecto y las tareas a realizarse.

h) Ritmo sostenible

Trabajar de manera productiva, eliminando cualquier distracción para evitar las horas extras de trabajo que desgasten al equipo de desarrollo.

i) Programación en parejas

Compartir el conocimiento y el código entre dos personas, ayuda a aclarar las ideas en el proceso de desarrollo.

j) Diseño incremental

El diseño incremental del software permite la evolución de la aplicación. Cada rediseño obliga, al equipo de desarrollo, a refactorar el código.

k) Pruebas antes de programar

Las pruebas unitarias obligan, al desarrollador, a enfocarse en la solución del problema y a confirmar el diseño.

l) Construir en diez minutos

Debe ser posible compilar y probar el sistema en diez minutos sin que esto impacte el ritmo de trabajo. Si algo falla en el proceso, esto permitirá rectificar de inmediato.

m) Integración continua

Cada vez que se tenga una nueva funcionalidad y que el código sea modificado, se debe recompilar y probar la integración del software.

2.7.2 SCRUM

Scrum se define como un marco referencial de trabajo que propone un conjunto de prácticas, principios y valores que permiten organizar y gestionar tareas en el desarrollo de productos o servicios. Takeduchi, Hirotaka & Nonaka, Ikujiro (1986) mencionan el término “Scrum” por primera vez en el documento *The New Product Development Game*, en donde se describe un proceso de desarrollo ágil, mismo que es capaz de responder a la necesidad de ser más competitivos (Álvarez García, de las Heras del Dedo, & Lasa Gómez, 2012)

2.7.2.4 PROCESO SCRUM

Según (Álvarez García, de las Heras del Dedo, & Lasa Gómez, 2012) Scrum define dos etapas en su proceso de organización del trabajo. La primera denominada *El Sprint 0* o etapa inicial; y la segunda de iteraciones sucesivas, también llamadas *Sprints.* En **Figura 5 Proceso SCRUM**

Fuente: (Álvarez García et al, 2012), se puede observar un esquema del proceso de Scrum.

PROCESO SCRUM

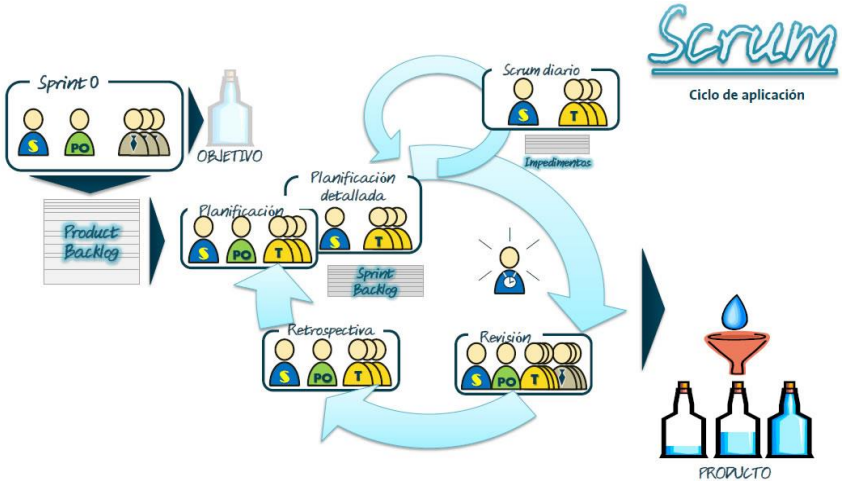

Figura 5 Proceso SCRUM

Fuente: (Álvarez García et al, 2012)

2.7.2.5 ROLES

Las actividades que plantea el Scrum, guían la elaboración del producto que el cliente necesita en base a sus requerimientos. Para dar vida al producto se requiere de un equipo de personas también conocido como: Scrum Team.

a) Product Owner

Es la única persona, dentro del Scrum Team, que conoce el giro del negocio del cliente, mantiene la visión del producto, trabaja con el cliente en

el levantamiento de los requerimientos y facilita la comunicación con el equipo de desarrollo.

Es también el encargado de mantener el Product Backlog actualizado y se asegura que el equipo de desarrollo construya el producto correcto.

b) Scrum Master

Esta persona debe dominar el marco de trabajo, que Scrum propone, con el fin de mantener en funcionamiento el proceso, ayudar al Product Owner con le elaboración y actualización del Product Backlog, además de gestionar los impedimentos que se van presentando en el desarrollo del proyecto.

c) Equipo de desarrollo

Es un grupo de profesionales que deben tener las habilidades necesarias para elaborar y entregar el producto en base a los requerimientos planteados por el cliente.

El equipo de desarrollo tiene la autoridad para auto-organizar su trabajo con el fin de alcanzar los objetivos de cada Sprint.

2.7.2.6 PRACTICAS

Según (Scrum-Alliance, 2014) Con el fin de mantener el control y realizar el seguimiento del trabajo, que el equipo de desarrollo va a realizar, Scrum define una serie de artefactos y herramientas

a) Product Backlog

Es una lista de ideas o requerimientos priorizados del producto, de donde se desprende el trabajo que el Scrum Team debe realizar. El encargado de elaborar y actualizar este listado es el Product Owner.

b) Sprint Backlog

Es el elemento resultante de la etapa de planificación del Sprint o Sprint Planning. Contiene el número de requerimientos refinados que el equipo de desarrollo se compromete a desarrollar en el período de duración de un Sprint.

c) Product Increment

Es el artefacto más importante de Scrum ya que constituye el incremento de producto u objetivo que cada Sprint debe producir. Debe cumplir con los requerimientos planteados, ser utilizable por parte del usuario y aceptado por el Product Owner.

2.7.3 SRUM COMBINADO CON XP

Tanto Scrum como XP son marcos de trabajo que se pueden complementar entre sí. Scrum trata de organizar la forma en la que los equipos abordan el trabajo mientras que el objetivo de Extreme Programming (XP) es construir software de manera productiva, con menos defectos y más económico. (Álvarez García, de las Heras del Dedo, & Lasa Gómez, 2012)

Henrik Kniber (2007) plantea que algunas prácticas de XP, como por ejemplo “Equipo completo”, “Sentarse juntos”, “Historias de usuario” y “Planificación” son tratadas directamente por Scrum.

Con base en estas dos afirmaciones se puede decir que es viable combinar estas dos metodologías de desarrollo y se puede definir las actividades y artefactos utilizaremos de Scrum y las prácticas, de XP, se aplicarán en el desarrollo de este proyecto.

2.7.3.7 PRACTICAS TOMADAS DE XP

Las prácticas escogidas para la programación y codificación del sistema son:

- **Diseño Incremental:** Mejorar el diseño del software conforme se avanza en el desarrollo. Reescribir los métodos y las clases que sean necesarias.
- **Integración Continua:** Compilar el código y desplegarlo cada vez que se tenga una nueva funcionalidad del software.

2.7.3.8 PRACTICAS TOMADAS DE SCRUM

Las actividades y artefactos escogidos para organizar el trabajo del presente proyecto son:

- **Etapa inicial o Sprint 0:** Determinar la viabilidad del proyecto y elaborar la primera versión del Product Backlog en base a los requerimientos generales expuestos por el cliente.
- **Product Backlog o pila del producto:** Documento que contiene los requerimientos del cliente y que deberá ser refinado y actualizado en cada iteración.
- **Planificación del Sprint:** Reunión con todo el equipo de desarrollo que definirá el trabajo a realizar en el Sprint.
- **Sprint Backlog o pila de tareas:** Documento que contendrá el listado de requerimientos refinados que se atenderán en el Sprint
- **Revisión o Sprint Review:** Reunión con el cliente que evaluará el cumplimiento del trabajo pactado en base a los criterios de aceptación.
- **Sprint Increment:** Producto final de cada sprint

2.8 UML

UML (Unified Modeling Language), es lenguaje de modelado de sistemas de software más conocido y utilizado actualmente. (uml.org, 2012)

Al ser un lenguaje gráfico permite la visualización, especificación, construcción y documentación de un sistema, además de brindar algo muy similar a un plano del mismo, que incluye aspectos como los procesos, las funciones del sistema, y los aspectos concretos como las expresiones de lenguajes de programación, o los diagramas de bases de datos.

El modelado es el diseño de las aplicaciones de software antes de su codificación, es una parte esencial en los grandes proyectos de software, y es útil para proyectos mediano y pequeños. Un modelo desempeña un papel semejante en el desarrollo de software al que los planos en la construcción de un edificio (uml.org, 2012).

2.8.1 Casos de Uso

Es un Diagrama UML que permite representar los casos de uso obtenidos en la especificación de requerimientos, consta de los siguientes componentes:

Actor: Alguien o algo que es externo al sistema y que interactúa con él desempeñando un rol en específico.

Caso de Uso: Es la representación de una acción a ejecutarse, que es siempre llamada por alguien o algo, los casos de uso pueden ser llamados a ejecutarse por otros casos de uso.

Relaciones/Asociaciones: Muestran la relación que existe entre los distintos elementos del modelo, pueden ser inclusión, cuando un actor o un caso de uso llama directamente a otro caso de uso, y extensión cuando existe

una llamada a otro caso de uso, pero que tiene características similares al mismo.

2.8.2 Diagrama de Secuencia

Este diagrama permite visualizar a detalle la manera en que interactúan los objetos de una forma ordenada según una secuencia temporal en los distintos eventos, muestra los objetos que participan en la interacción, como líneas de vida a lo largo del tiempo.

2.8.3 Diagrama de Clases

Los diagramas de clases describen los tipos de objetos, la variedad de clases y las relaciones que existen en un sistema y se componen de un conjunto de asociaciones, atributos, métodos, operaciones y restricciones para construir una clase.

CAPÍTULO 3

ESTUDIO COMPARATIVO

3.1 Revisión de frameworks existentes

En la actualidad vivimos en una era digital en donde la popularidad de las aplicaciones móviles está aumentando rápidamente. Ahora, cada vez más personas están utilizando smartphones y con ellos una gran cantidad de aplicaciones para sus tareas diarias. Con una creciente demanda y la popularidad de estas aplicaciones, muchas empresas de desarrollo de aplicaciones móviles se esfuerzan por mejorar la experiencia del usuario optando por la última tecnología en desarrollo de aplicaciones es por ello que es esencial elegir un framework adecuado para el desarrollo de las aplicaciones híbridas.

A continuación, se describen brevemente los 7 frameworks para el desarrollo de aplicaciones híbridas más populares y su evaluación según la empresa de consultoría e ingeniería digital PixelCrayons[1]

3.1.1 React

El objetivo principal de este framework es crear aplicaciones nativas, en lugar de desarrollar aplicaciones híbridas que se ejecutan en una vista web. Sin embargo, su desarrollo se realiza completamente por medio de React y JavaScript.

Este framework no es personalizado para los principiantes en el desarrollo de sitios web, pero cuenta con una gran comunidad que puede ayudar en el proceso de desarrollo. React tiene soporte para Android, por lo que ahora también puede desarrollar aplicaciones reales multiplataforma.

[1] Tomado de: www.pixelcrayons.com/blog/mobile/7-best-hybrid-app-development-frameworks-for-2017

Tabla 1

Características REACT

Prerrequisitos	React
Apariencia nativa	8/10
Comunidad	8/10
Documentación	5/10
Herramientas	React Developer Tools extensión para Chrome
A favor	Rendimiento Nativo
	Gran Comunidad
En Contra	Amplia curva de aprendizaje

Fuente: (pixelcrayons, 2017)

3.1.2 Onsen UI

Con la ayuda de este framework de código abierto, se puede crear aplicaciones combinando componentes de aspecto nativo. Este framework es fácil de usar y capaz de funcionar sin AngularJS. Posee una buena documentación que incluye gran cantidad de diseños y ejemplos de estructuras para aplicaciones comunes.

Tabla 2

Características Onsen UI

Prerrequisitos	Opcional Angular JS
Apariencia nativa	6/10
Comunidad	4/10
Documentación	9/10
Herramientas	Monaca Cloud IDE
A favor	Trabaja bien con componentes predefinidos
	Excelente documentacion con ejemplos
En Contra	Aún no soporta Material Design

Fuente: (pixelcrayons, 2017)

3.1.3 IONIC

Este es un framework popular y preferido por muchos desarrolladores. La parte CSS de este framework se puede utilizar para crear diseños de aspecto similar al nativo. Para aprovechar todo el potencial de Ionic, debe combinarse con AngularJS. También posee una interfaz de línea de comandos que incluyen un paquete de aplicación basado en Cordova y emuladores integrados.

Tabla 3

Características IONIC

Prerrequisitos	Opcional Angular JS
Apariencia nativa	7/10
Comunidad	9/10
Documentación	8/10
Herramientas	Interfaz de linea de comandos, Ionic SDK
A favor	Trabaja con componentes predefinidos
	Gran Comunidad
	CLI con variedad de funciones
En Contra	Requiere de Angular JS para trabajar a su máximo

Fuente: (pixelcrayons, 2017)

3.1.4 Framework 7

Framework 7 no tiene dependencias externas como React o Angular. Sin embargo, puede realizar aplicaciones que se sientan y se vean como nativas, con animaciones y componentes adecuadamente diseñados. Permite desarrollar aplicaciones de manera sencilla con conocimientos de CSS, HTML y JavaScript. Este framework no incluye ninguna herramienta para empaquetar o emular aplicaciones, es por ello que debe ser combinarlo con PhoneGap o Cordova.

Tabla 4

Características Framework 7

Prerrequisitos	HTML, CSS, JS
Apariencia nativa	8/10
Comunidad	6/10
Documentación	8/10
Herramientas	Ninguna
A favor	Simple para desarrollar aplicaciones con conocimientos básicos
	Buen rendimiento
	Puede combinarse con cualquier framework JS
En Contra	Requiere de Cordova o PhoneGap para las emulaciones

Fuente: (pixelcrayons, 2017)

3.1.5 jQuery Mobile

Uno de los frameworks móviles más antiguos, jQuery Mobile no intenta hacer aplicaciones parecidas a Android o iOS. Por su parte su objetivo es ayudar en el desarrollo de aplicaciones web que funcionen bien en todos los navegadores móviles (incluyendo Blackberry, Windows Phone y Symbian). Este framework es conocido por ser ligero, fácil de usar y aprender.

Tabla 5

Características jQuery Mobile

Prerrequisitos	jQuery
Apariencia nativa	3/10
Comunidad	8/10
Documentación	5/10
Herramientas	Ninguna
A favor	Soporte para varios navegadores móviles
	Sencillo de usar
En Contra	Estilos que no se asemejan a iOS o Android
	Requiere de Cordova o PhoneGap para las emulaciones

Fuente: (pixelcrayons, 2017)

3.1.6 Famous

Reconocido por su enfoque único de desarrollo, este framework combina el árbol DOM (su HTML) con WebGL y muestra todo en un lienzo similar a lo que hacen los motores de juegos HTML. Esta última técnica permite al framework ejecutar aplicaciones móviles en 60 fps, que es tan fluido como las otras aplicaciones nativas. Desafortunadamente, este proyecto ya no se está desarrollando y no tiene buena documentación de referencia.

Tabla 6

Características Famous

Prerrequisitos	WebGL, AngularJS
Apariencia nativa	7/10
Comunidad	3/10
Documentación	5/10
Herramientas	Ninguna
A favor	Rendimiento Nativo
En Contra	No cuenta con desarrollos activos
	Baja Documentación
	Pequeña comunidad

Fuente: (pixelcrayons, 2017)

3.1.7 Native Script

La característica más grande de Native Script es que permite escribir la funcionalidad de la aplicación en JavaScript una vez, que luego se cambiará de acuerdo con iOS, Android y Windows Phone. Sin embargo, este framework requiere habilidades avanzadas de codificación, que se compensa con una documentación extensa.

Tabla 7

Características Native Script

Prerrequisitos	JavaScript
Apariencia nativa	8/10
Comunidad	5/10
Documentación	9/10
Herramientas	CLI gratuita, otras opciones pagadas
A favor	Codifica una vez, use donde sea
	No cuenta con desarrollos activos
En Contra	Gran curva de aprendizaje
	Pequeña comunidad

Fuente: (pixelcrayons, 2017)

3.2 Selección de entornos

Para seleccionar los frameworks a evaluar se tomará en cuenta los 3 con la más alta calificación según la revisión de frameworks.

Tabla 8

Resumen Evaluación Frameworks

	React	Onsen UI	Ionic 2	Framework 7	jQuery Mobile	Famous	Native Script
Apariencia Nativa	8	6	7	8	3	7	8
Comunidad	8	4	9	6	8	3	5
Documentación	5	9	8	8	5	5	9
Total	**21**	**19**	**24**	**22**	**16**	**15**	**22**

De la Tabla 8

Resumen Evaluación Frameworks se eligió los 3 Frameworks con más alto puntaje que son: Ionic, Framework 7 y React, Native Script (se descartó por necesitar de herramientas pagadas para utilizar componentes avanzados) (Molina, 2017).

A estos 3 frameworks se les aplicará el modelo de calidad ISO para determinar los 2 mejores de acuerdo a los criterios de evaluación planteados en el modelo de calidad, finalmente con los resultados obtenidos se aplicará un FODA en ambos para elegir el framework con el cual se desarrollará el aplicativo.

3.3 Modelo de calidad

3.3.1 Determinación de las métricas

3.3.1.1 Métrica de cumplimiento (C)

Métrica basada en el cumplimiento de algo en específico puede tomar el valor de 0 o 1 dependiendo de si cumple o no con una condición.

Tabla 9

Métrica de cumplimiento

N°	Cumplimiento	Equivalente Contable
1	SI	1
2	NO	0

3.3.1.2 Métrica por Rangos (R)

Métrica basada en un rango de desempeño, puede tomar los valores del 0 al 4 dependiendo del nivel de cumplimiento de una condición.

a) Métrica por rangos ascendente (R1)

Permite cuantificar un valor mediante un rango ascendente siendo 0 el valor más bajo y 4 el valor más alto.

Tabla 10

Métrica por rangos ascendente

N°	Cumplimiento	Equivalente Contable
1	Nulo	0
2	Bajo	1
3	Medio	2
4	Alto	3
5	Muy Alto	4

b) Métrica por rangos descendente (R2)

Permite cuantificar un valor mediante un rango descendente siendo 4 el valor más bajo y 0 el valor más alto.

Tabla 11

Métrica por rangos descendente

N°	Cumplimiento	Equivalente Contable
1	Nulo	4
2	Bajo	3
3	Medio	2
4	Alto	1
5	Muy Alto	0

3.3.2 Construcción del Modelo

Para la construcción del modelo de calidad se tomó en cuenta las características de calidad según el Modelo de evaluación ISO, las métricas planteadas en la Tabla 9

Métrica de cumplimiento,

Tabla 10

Métrica por rangos ascendente, Tabla 11

Métrica por rangos descendente y los Frameworks seleccionados en la Selección de entornos.

Tabla 12

Características y Sub características modelo de Calidad

TIPO	CARACTERISTICA	SUB CARACTERISTICA
Calidad Interna y Externa	Funcionalidad	Adecuación
		Exactitud
		Interoperabilidad
		Seguridad de acceso
	Fiabilidad	Madurez
		Tolerancia a Fallos
		Capacidad de Recuperación
	Usabilidad	Capacidad para ser Entendido
		Capacidad para ser Aprendido
		Capacidad para ser Operado
		Capacidad de Atracción

Continua

TIPO	CARACTERISTICA	SUB CARACTERISTICA
	Mantenibilidad	Capacidad para ser Analizado
		Capacidad para ser Cambiado
		Estabilidad
		Capacidad para ser Probado
	Portabilidad	Adaptabilidad
		Facilidad de Instalación
		Conformidad
		Capacidad de reemplazo
	Eficiencia	Comportamiento Temporal
		Utilización de recursos
Calidad de Uso		Efectividad
		Productividad
		Seguridad
		Satisfacción

De la

Tabla 12

Características y Sub características modelo de Calidad se eligieron los elementos más relevantes que serán aplicados al modelo de calidad para el análisis comparativo entre frameworks, y se detalló una breve descripción de cada una en base a la necesidad del análisis.

Tabla 13
Descripción de las características de calidad

TIPO	CARACTERISTICA	SUB CARACTERISTICA	DESCRIPCION
Calidad Interna y Externa	Funcionalidad	Adecuación	Cuenta con componentes predefinidos
		Exactitud	Concordancia de referencias a través del sitio oficial
		Interoperabilidad	Puede funcionar con otros frameworks
		Seguridad de acceso	Cuenta con licencias libres o propietarias
	Fiabilidad	Madurez	Tiempo en el mercado
	Usabilidad	Capacidad para ser Entendido	Continua

TIPO	CARACTERISTICA	SUB CARACTERISTICA	DESCRIPCION
		Capacidad para ser Aprendido	Documentación con Ejemplos teóricos y prácticos
		Capacidad para ser Operado	Ejemplos con Showcase
		Capacidad de Atracción	Apariencia Nativa
	Mantenibilidad	Capacidad para ser Analizado	Complejidad en la codificación
		Capacidad para ser Cambiado	Cuenta con una compatibilidad adecuada entre versiones
		Estabilidad	Soporte adecuado y respaldo de la comunidad
		Capacidad para ser Probado	Soluciones en la comunidad y sitio oficial
	Portabilidad	Adaptabilidad	Soporta Android, iOS y Windows Phone
		Facilidad de Instalación	Descarga desde sitio Oficial, con soporte de instalación
	Eficiencia	Comportamiento Temporal	Tiempos de respuesta
		Utilización de recursos	Cantidad de recursos utilizados en la máquina
Calidad de Uso		Efectividad	Cantidad de recursos utilizados en la aplicación

Finalmente se estableció que tipo de métrica utilizaría cada elemento en el modelo de evaluación teniendo como resultado el siguiente modelo de calidad.

Tabla 14
Estructura del Modelo de Calidad

TP	CARÁCTER.	SUB CARÁCTER.	DESCRIPCION	PUNTUACIÓN	METRICA
Calidad Interna y Externa	Funcionalidad	Adecuación	Cuenta con componentes predefinidos	1	C
		Exactitud	Concordancia de referencias a través del sitio oficial	4	R1

Continua

TP	CARÁCTER.	SUB CARÁCTER.	DESCRIPCION	PUNTUACIÓN	METRICA
		Interoperabilidad	Puede funcionar con otros frameworks	1	C
		Seguridad de acceso	Cuenta con licencias libres o propietarias	1	C
	Fiabilidad	Madurez	Tiempo en el mercado	4	R1
	Usabilidad	Capacidad para ser Entendido	Documentación con Ejemplos teóricos y prácticos	4	R1
		Capacidad para ser Aprendido			
		Capacidad para ser Operado	Ejemplos con Showcase	4	R1
		Capacidad de Atracción	Apariencia Nativa	1	C
	Mantenibilidad	Capacidad para ser Analizado	Complejidad en la codificación	4	R2
		Capacidad para ser Cambiado	Cuenta con una compatibilidad adecuada entre versiones	4	R1
		Estabilidad	Soporte adecuado y respaldo de la comunidad	4	R1
		Capacidad para ser Probado	Soluciones en la comunidad y sitio oficial	4	R1
	Portabilidad	Adaptabilidad	Soporta Android, iOS y Windows Phone	1	C
		Facilidad de Instalación	Descarga desde sitio Oficial, con soporte de instalación	1	C
	Eficiencia	Comportamiento Temporal	Tiempos de respuesta	4	R2
		Utilización de recursos	Cantidad de recursos utilizados en la máquina	4	R2
Calidad de Uso		Efectividad	Cantidad de recursos utilizados en la aplicación	4	R2
Total				50	

3.3.3 Aplicación del Modelo

Aplicando el modelo se obtuvieron los datos mostrados a continuación:

Tabla 15
Resumen Modelo de calidad con ponderaciones

TIPO	SUB CARACTERISTICA	DESCRIPCIÓN	FRAMEWORK 7	REACT	IONIC
			PUNTAJE	PUNT.	PUNT.
Calidad Interna y Externa	Adecuación	Cuenta con componentes predefinidos	1	1	1
	Exactitud	Concordancia de referencias a través del sitio oficial	4	2	4
	Interoperabilidad	Puede funcionar con otros frameworks	1	1	1
	Seguridad de acceso	Cuenta con licencias libres o propietarias	1	1	1
	Madurez	Tiempo en el mercado	3	4	4
	Capacidad para ser Entendido	Documentación con Ejemplos teóricos y prácticos	3	3	4
	Capacidad para ser Aprendido				
	Capacidad para ser Operado	Ejemplos con Showcase	3	2	4
	Capacidad de Atracción	Apariencia Nativa	1	1	1
	Capacidad para ser Analizado	Complejidad en la codificación	3	3	3
	Capacidad para ser Cambiado	Cuenta con una compatibilidad adecuada entre versiones	3	4	3
	Estabilidad	Soporte adecuado y respaldo de la comunidad	3	3	3
	Capacidad para ser Probado	Soluciones en la comunidad y sitio oficial	3	3	4
	Adaptabilidad	Soporta Android, iOS y Windows Phone	0	1	1
	Facilidad de Instalación	Descarga desde sitio Oficial, con soporte de instalación	1	1	1
	Comportamiento Temporal	Tiempos de respuesta	3	3	3
	Utilización de recursos	Cantidad de recursos utilizados en la máquina	3	3	2
Calidad de uso	Efectividad	Cantidad de recursos utilizados en la aplicación	3	3	3
Total			39	40	43

3.3.4 Resultados Modelo

Para poder tener una mejor visión de los resultados obtenidos se realizó una tabla con todas las características, ponderaciones, métricas y puntajes.

Tabla 16
Modelo de calidad completo

TIPO	CARACTERISTICA	SUB CARACTERISTICA	DESCRIPCIÓN	PONDERACIÓN MÁXIMA	MÉTRICA	FRAMEWORK 7	REACT	IONIC
						PUNTAJE	PUNTAJE	PUNTAJE
Calidad Interna y Externa	Funcionalidad	Adecuación	Cuenta con componentes predefinidos	1	C	1	1	1
		Exactitud	Concordancia de referencias a través del sitio oficial	4	R1	4	3	4
		Interoperabilidad	Puede funcionar con otros frameworks	1	C	1	1	1
		Seguridad de acceso	Cuenta con licencias libres o propietarias	1	C	1	1	1
	Fiabilidad	Madurez	Tiempo en el mercado	4	R1	3	4	4
	Usabilidad	Capacidad para ser Entendido	Documentación con Ejemplos teóricos y prácticos	4	R1	3	3	4
		Capacidad para ser Aprendido						
		Capacidad para ser Operado	Ejemplos con Showcase	4	R1	3	2	4
		Capacidad de Atracción	Apariencia Nativa	1	C	1	1	1
	Mantenibilidad	Capacidad para ser Analizado	Complejidad en la codificación	4	R2	3	3	3
		Capacidad para ser Cambiado	Cuenta con una compatibilidad adecuada entre versiones	4	R1	3	4	3
		Estabilidad	Soporte adecuado y resplado de la comunidad	4	R1	3	3	3
		Capacidad para ser Probado	Soluciones en la comunidad y sitio oficial	4	R1	3	3	4
	Portabilidad	Adaptabilidad	Soporta Android, iOS y Windows Phone	1	C	0	1	1
		Facilidad de Instalación	Descarga desde sitio Oficial, con soporte de instalación	1	C	1	1	1
	Eficiencia	Comportamiento Temporal	Tiempos de respuesta	4	R2	3	3	3
		Utilización de recursos	Cantidad de recursos utilizados en la máquina	4	R2	3	3	2
Calidad de Uso		Efectividad	Cantidad de recursos utilizados en la aplicación	4	R2	3	3	3
Total				50		39	40	43

Como se puede observar en la

Tabla 16

Modelo de calidad completo, el framework con mayor puntaje es Ionic con 43 puntos, seguido de React con 40 puntos y finalmente Framework 7 con 39 puntos, para poder entender de mejor manera como se llegó a este resultado se procedió a realizar una serie de diagramas mostrados a continuación:

En la **Figura 6 Resultados de adecuación**, podemos observar que los 3 frameworks poseen el mismo puntaje ya que todos poseen componentes predefinidos.

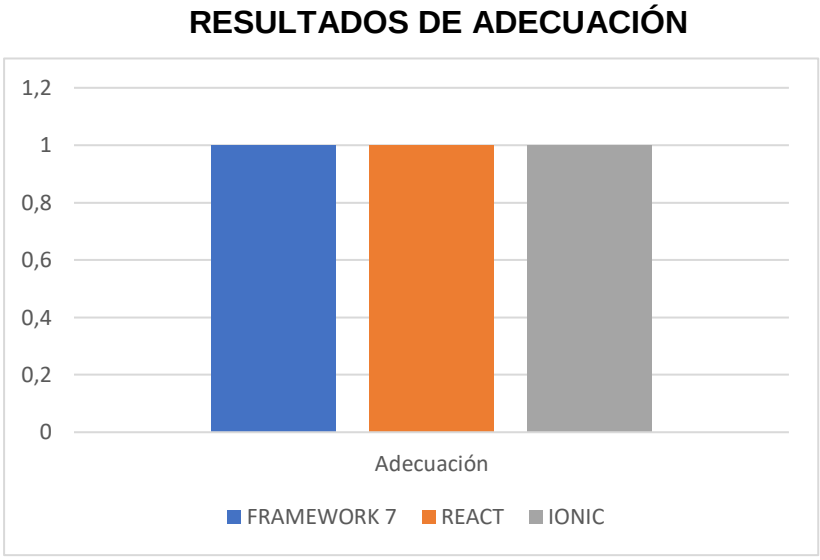

Figura 6 Resultados de adecuación

En la **Figura 7 Resultados de Exactitud**, observamos tanto Framework 7 como Ionic poseen un valor superior al de React ya que su concordancia de referencias en su sitio oficial no es tan exacta.

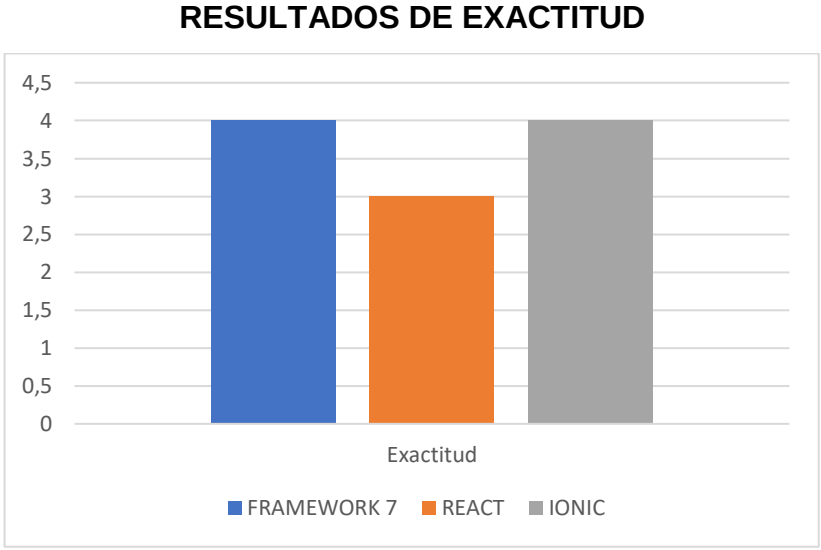

Figura 7 Resultados de Exactitud

La Figura 8 Resultados de Interoperabilidad nos muestra que todos los frameworks en cuestión tienen la facilidad de funcionar con otros frameworks

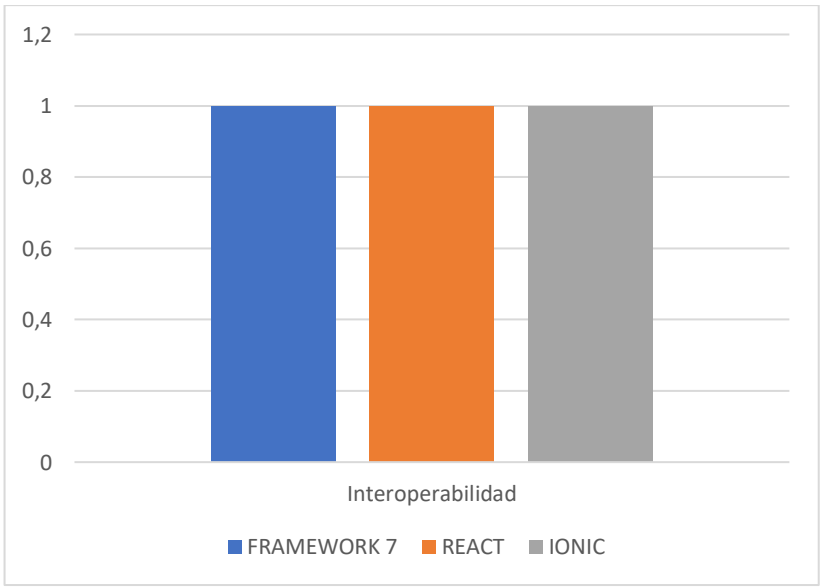

Figura 8 Resultados de Interoperabilidad

En la **Figura 9 Resultados de Seguridad de Acceso** podemos ver que los 3 frameworks comparten el mismo puntaje al poseer una licencia libre o propietaria.

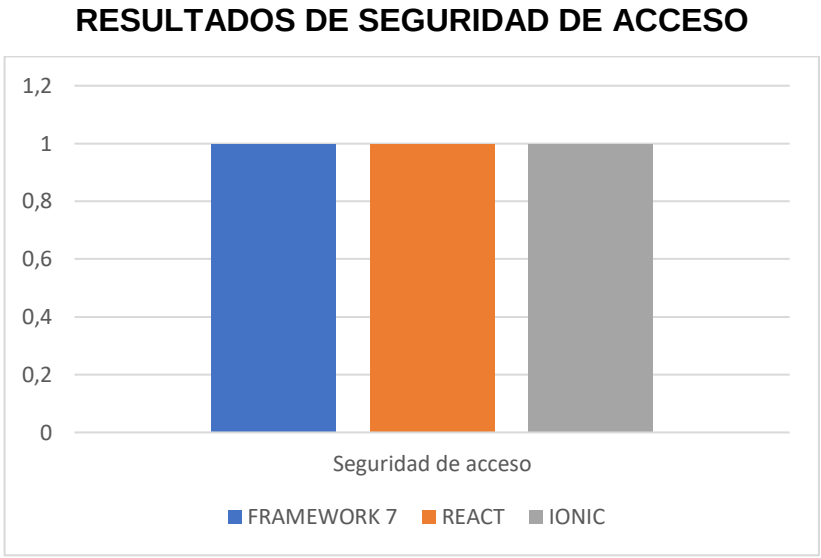

Figura 9 Resultados de Seguridad de Acceso

La **Figura 10 Resultados de Madurez¡Error! No se encuentra el origen de la referencia.**, nos muestra que Ionic y React superan por poco a Framework 7 en lo que se refiere a tiempo en el mercado

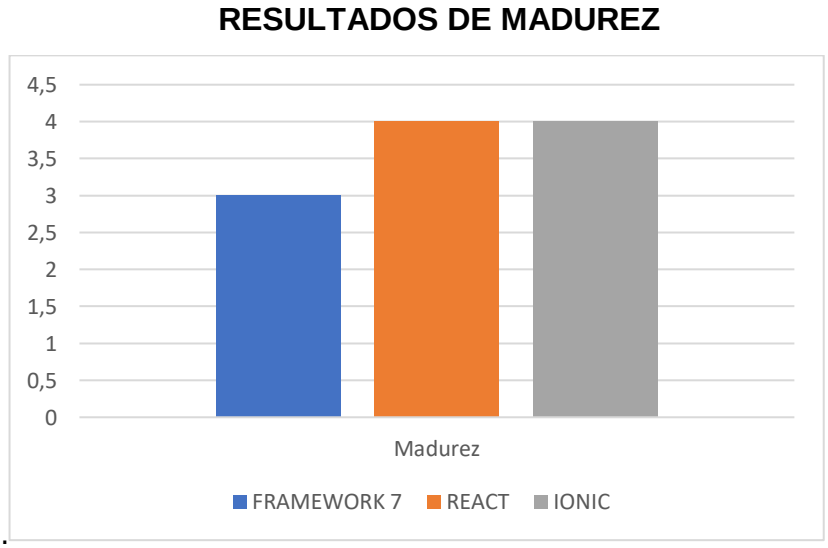

Figura 10 Resultados de Madurez

En la

Figura 11 Resultados de Capacidad para ser Entendido, podemos ver que Ionic tiene el puntaje más alto al contar con una mayor documentación y ejemplos teórico prácticos.

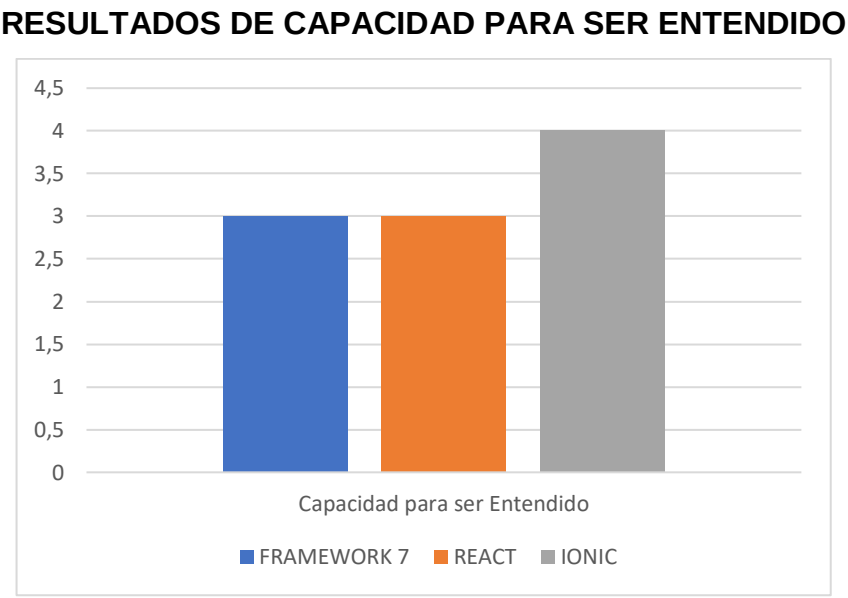

Figura 11 Resultados de Capacidad para ser Entendido

Ionic posee una cantidad mayor de ejemplos con un showcase que Framework 7 y esté supera a React respectivamente como se puede ver en la

Figura 12 Resultados de Capacidad para ser Operado

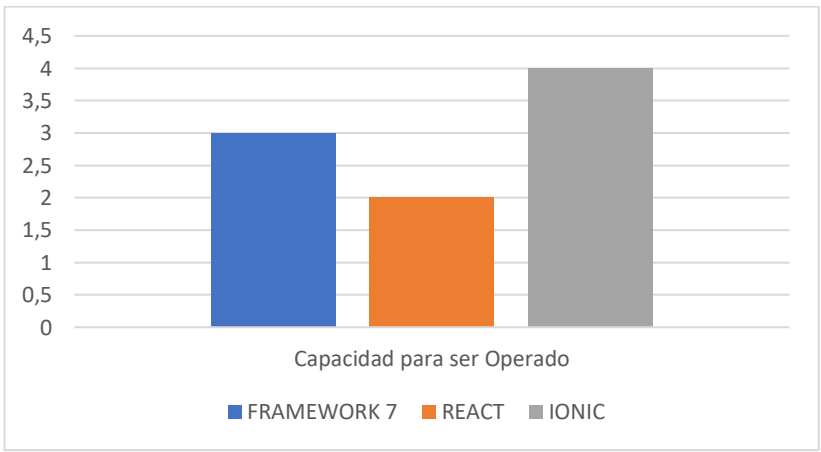

Figura 12 Resultados de Capacidad para ser Operado

La **Figura 13 Resultados de Capacidad de atracción**, nos muestra que los framework poseen la característica de contar con una apariencia nativa al momento de ejecutarse en distintas plataformas.

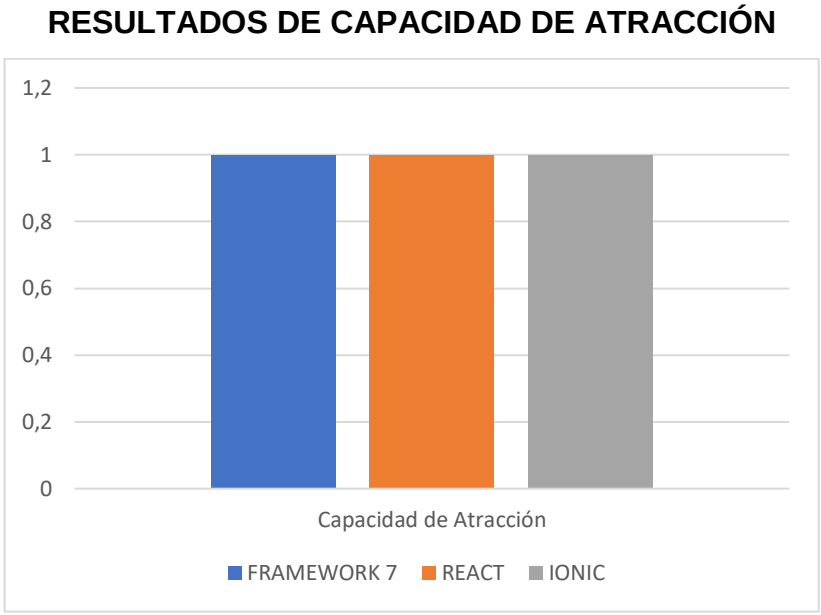

Figura 13 Resultados de Capacidad de atracción

En la **Figura 14 Resultados de Capacidad para ser analizado**, podemos observar que los 3 frameworks cuentan con la misma complejidad al momento de la codificación.

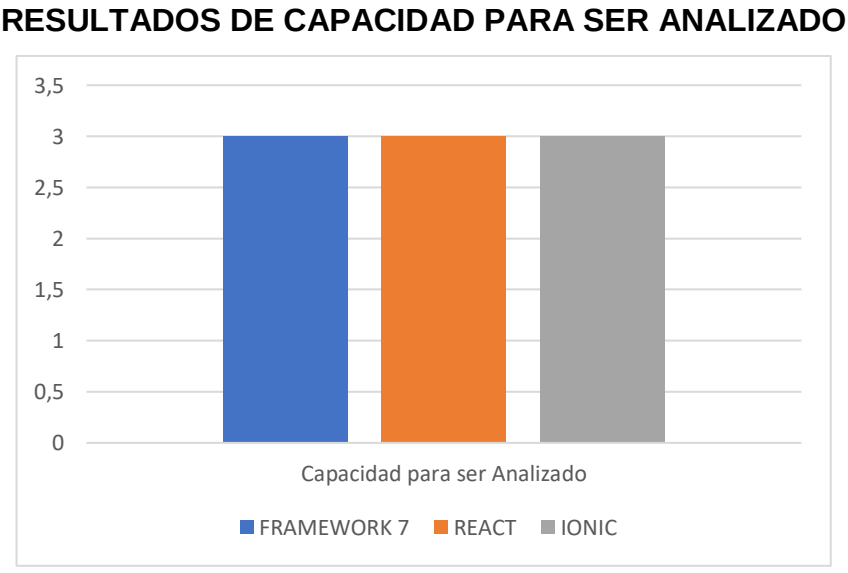

Figura 14 Resultados de Capacidad para ser analizado

React cuenta con una ventaja sobre Framework7 y Ionic en lo que se refiere a la compatibilidad con versiones anteriores al no presentar cambios tan grandes entre versión y versión como podemos observar en la **Figura 15 Resultados de Capacidad para ser cambiado**

RESULTADOS DE CAPACIDAD PARA SER CAMBIADO

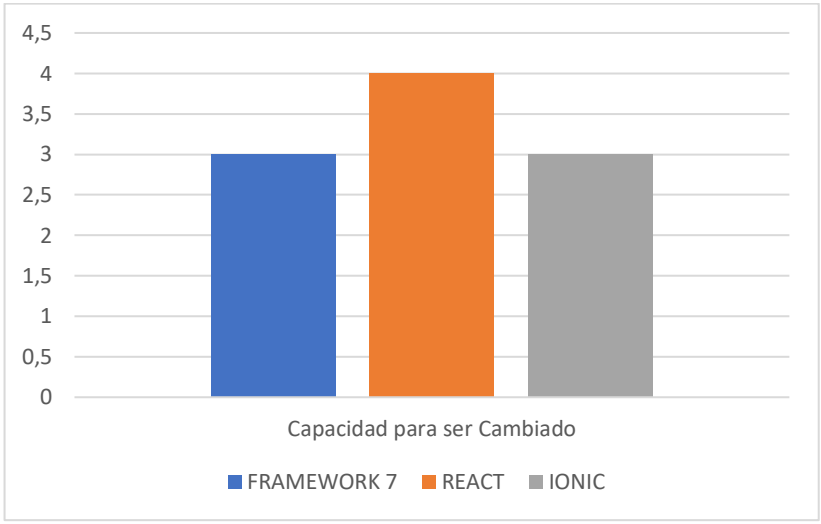

Figura 15 Resultados de Capacidad para ser cambiado

La **Figura 16 Resultados de Estabilidad**, nos muestra que tanto Framework 7, React y Ionic poseen un soporte adecuado con respaldo de la comunidad.

RESULTADOS DE ESTABILIDAD

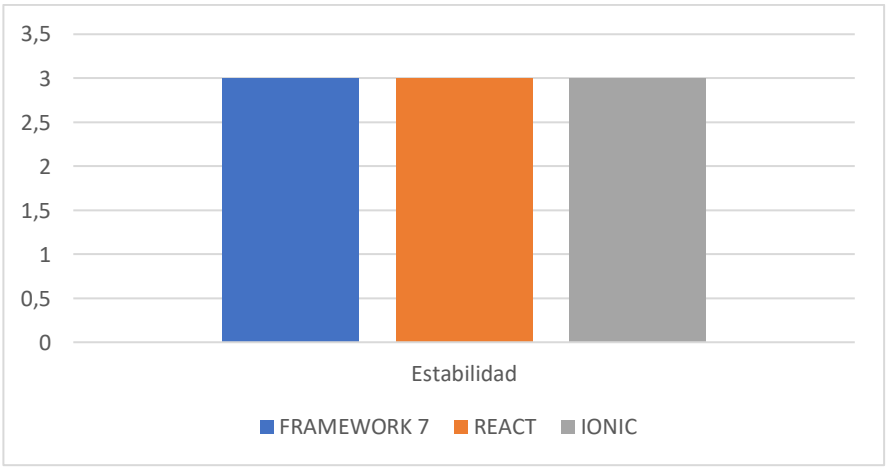

Figura 16 Resultados de Estabilidad

En la

Figura 17 Resultados de Capacidad para ser probado podemos observar que Ionic tiene una ligera ventaja en lo que se refiere a soluciones en la comunidad y su sitio Oficial.

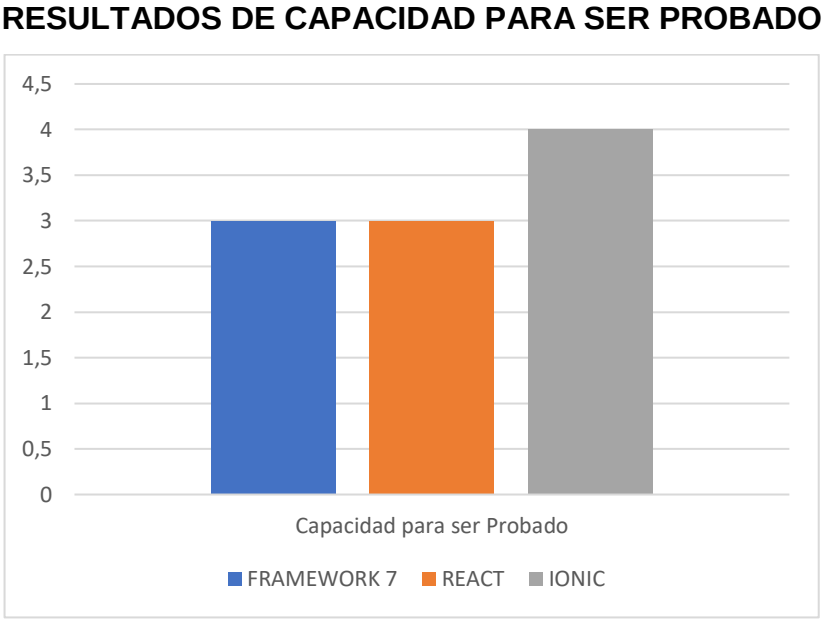

Figura 17 Resultados de Capacidad para ser probado

Ionic y React soportan el desarrollo para Android, iOS y Windows Phone, mientas que Framework 7 no soporta todas estas plataformas como indica la **Figura 18 Resultados de adaptabilidad**.

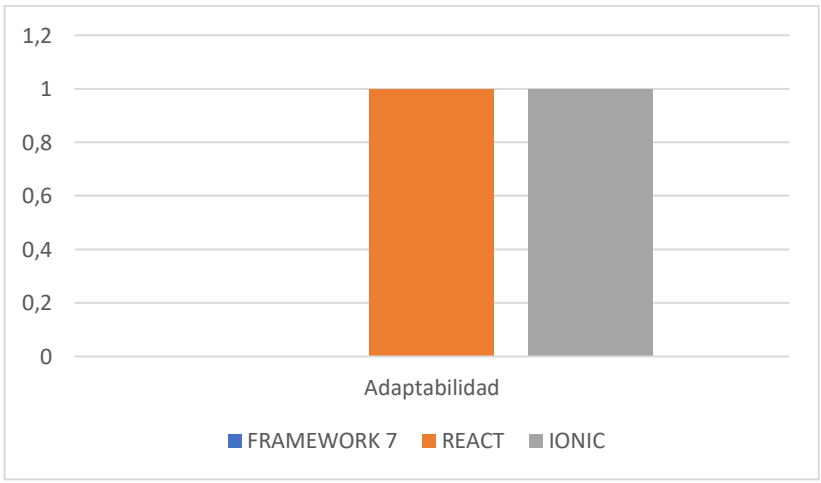

Figura 18 Resultados de adaptabilidad

La **Figura 19 Resultados de Facilidad de instalación** nos muestra que los 3 framework cuentan con la facilidad de descargarse desde su sitio oficial y tienen soporte en el proceso de instalación.

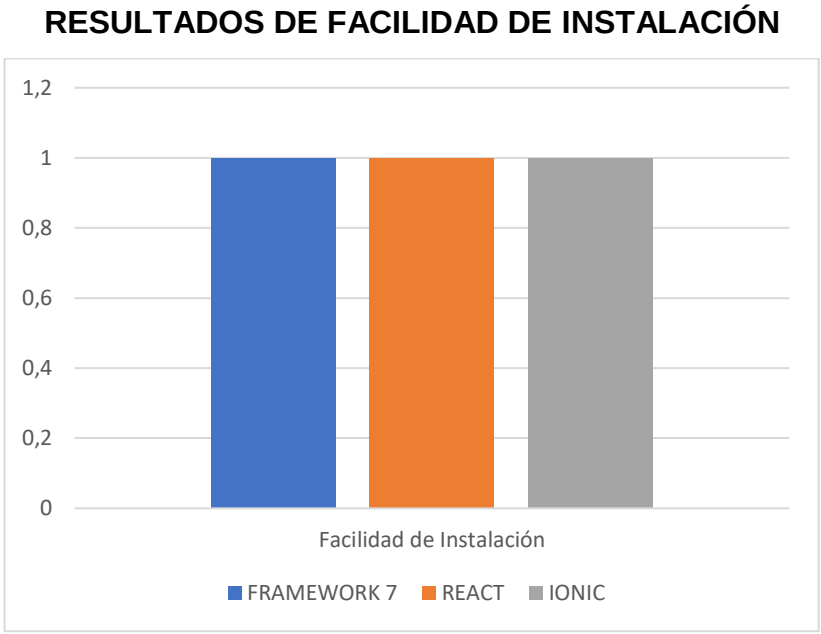

Figura 19 Resultados de Facilidad de instalación

En la **Figura 20 Resultados de Comportamiento temporal** podemos ver que los 3 frameworks poseen un tiempo de respuesta similar.

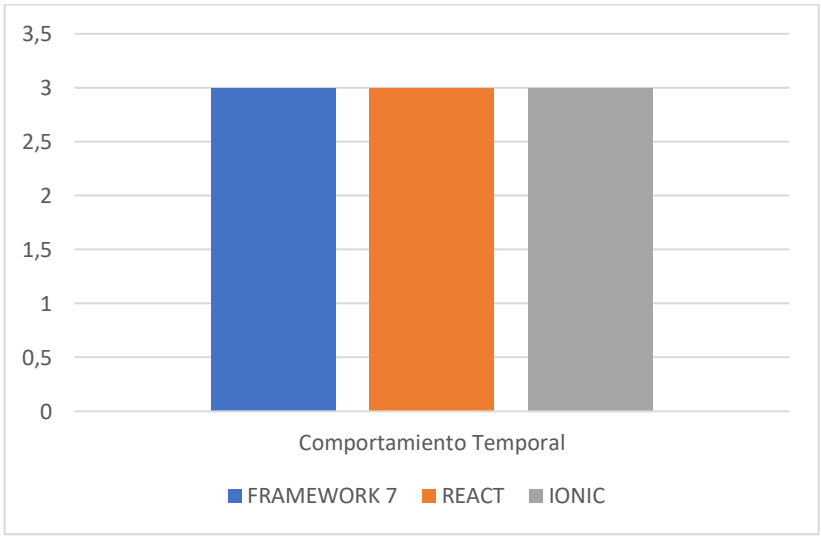

Figura 20 Resultados de Comportamiento temporal

La **Figura 21 Resultados de Utilización de recursos**, nos muestra que tanto Framework 7 y React utilizan ligeramente mejor los recursos que Ionic ya que éste cuenta con su propia CLI.

RESULTADOS DE UTILIZACIÓN DE RECURSOS

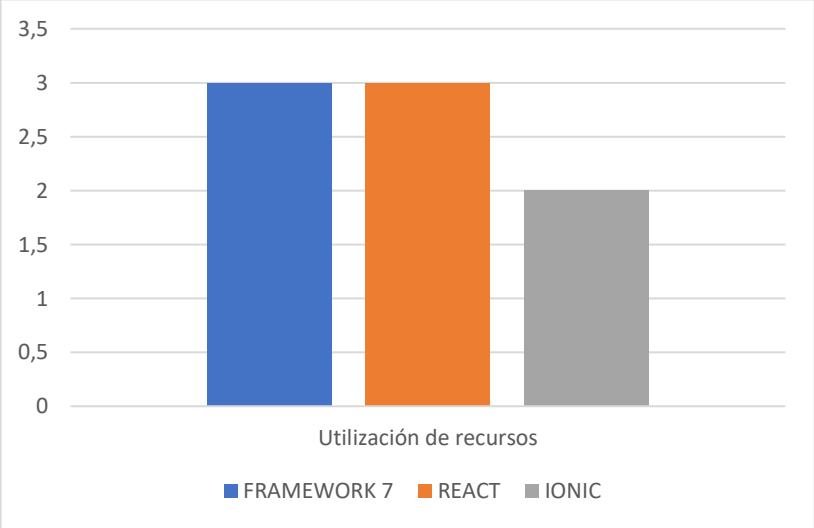

Figura 21 Resultados de Utilización de recursos

En la **Figura 22 Resultados de Efectividad**, podemos observar que todos los frameworks en cuestión poseen una cantidad similar de recursos utilizados en la aplicación.

RESULTADOS DE EFECTIVIDAD

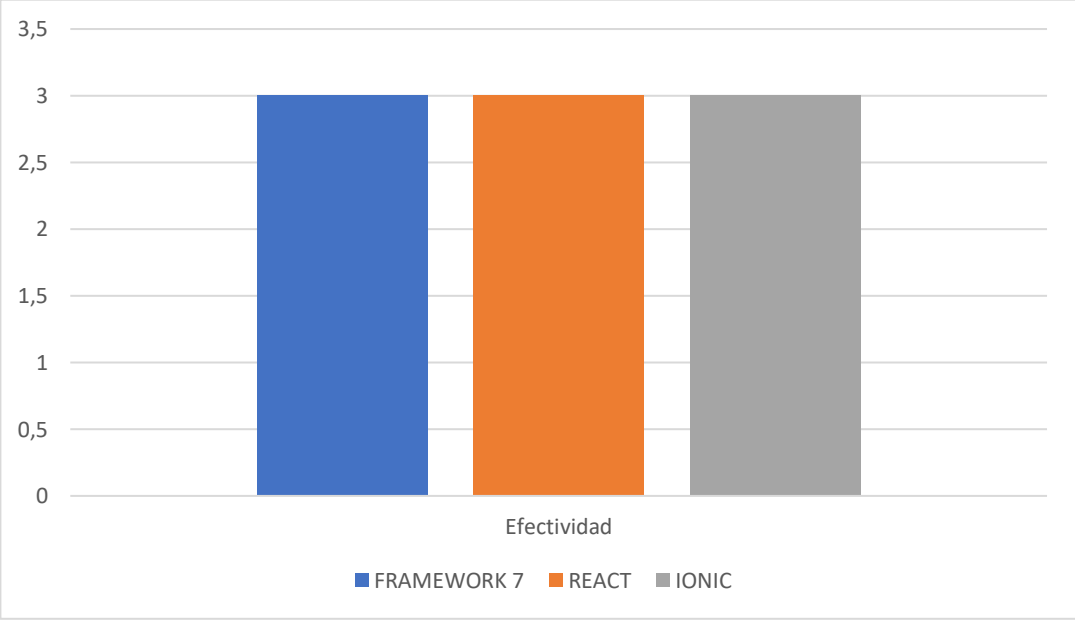

Figura 22 Resultados de Efectividad

Como podemos observar en las figuras anteriores, las métricas más determinantes para obtener un resultado son: exactitud, capacidad para ser entendido, capacidad para ser aprendido, capacidad para ser cambiado, capacidad para ser probado y la utilización de recursos.

3.3.5 Aplicación del FODA

En base a los resultados obtenidos en el modelo de calidad presentado anteriormente escogimos a Ionic y Reatc para aplicar una matriz FODA que ayudará a decidir cuál de los dos frameworks es más adecuado para la elaboración de aplicaciones híbridas.

3.3.5.1 FODA REACT

MATRIZ FODA REACT

FORTALEZAS

- Brinda un mejor apariencia al randerizar componentes nativos.
- Cuenta con una gran comunidad en GitHub.
- Posee un gran rendiemiento, además de facilitar un desarrollo ágil, ordenado y con una arquitectura mantenible.

OPORTUNIDADES

- Produce renderización nativa que permitirá ver el resultado de modificaciones a medida que las haga sin la necesidad de recompilar o reconstruir.
- Cuenta con el respaldo de Facebook.

REACT

DEBILIDADES

- Necesita de código específico para cada plataforma.
- Curva de aprendizaje con pocos ejemplos.
- Documentación muy básica.

AMENAZAS

- Dificulta en trabajo coopetativo con diseñadoresya que al basarse en la esctuctura Javascript React utiliza código Javascript semejante a HTML, por ende la lógica de visualización se combina con la lógica de negocio.

Figura 23 Matriz FODA React

3.3.5.2 FODA IONIC

MATRIZ FODA IONIC

Figura 24 Matriz FODA Ionic

3.3.6 Resultados Comparativa

Como podemos observar en 3.3.4 Resultados del Modelo, ambos framework poseen características favorables para desarrollo, acordes con las necesidades de la empresa SOFYA SYSTEMS S.A., teniendo Ionic una ligera ventaja sobre React en lo que se refiere a Capacidad para ser entendido, aprendido y probado, por otro lado, React supera a Ionic en capacidad para ser cambiado y el consumo de recursos.

De la **Figura 23 Matriz FODA React** y **Figura 24 Matriz FODA Ionic**, podemos ver que Ionic posee más ventajas que React en la codificación y documentación, mientras que React permite un mejor diseño nativo con un mayor rendimiento.

Ambos frameworks tienen sus pro y contras, y luego de haber aplicado el modelo de calidad y la matriz FODA, podemos llegar a la conclusión de que

Ionic supera a React como herramienta para el desarrollo de aplicaciones híbridas en la empresa SOFYA.

Tomando en cuenta la conclusión anterior, se procede a seleccionar a IONIC como el framework que dará soporte al desarrollo de la aplicación de este proyecto investigativo.

CAPÍTULO 4

DESARROLLO DE LA APLIACIÓN

4.1 ESPECIFICACIÓN DE REQUERIMIENTOS

4.1.1 Introducción

La Ingeniería de Requerimientos es un extenso conjunto de tareas y técnicas que brindan un mecanismo adecuado para analizar, entender y especificar las necesidades de un cliente (Pressman, 2010)

A continuación, se elaborará una Especificación de Requerimientos de Software (ERS) para el proyecto planteado, misma que pretende identificar el contexto del proyecto y determinará los requerimientos funcionales y no funcionales del mismo.

La siguiente especificación estará basará en el estándar del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE Institute of Electrical and Electronics Engineers), Practicas Recomendadas para los Requerimientos de Software IEEE/ANSI 830-1998 (IEEE, 2014).

4.1.2 Requisitos Específicos

A continuación, se detallan los requisitos funcionales y no funcionales de la aplicación utilizando los siguientes acrónimos para un mejor manejo de la información:

RE: Requerimiento Específico
NR: Nombre Del Requerimiento

4.1.2.1 Requisitos funcionales

En la Tabla 17

Requisitos Funcionales se muestra una descripción, nombre y un código asignado a cada uno de los requisitos funcionales, mismo que fueron realizados en base a las necesidades del cliente.

Tabla 17
Requisitos Funcionales

RE	NR	Descripción
RE01	Registrar Usuario	Se podrá crear un usuario para que este pueda hacer uso de la app
RE02	Iniciar sesión	El usuario podrá ingresar a la app con sus credenciales
RE03	Visualizar contenido	El usuario podrá ver el contenido en la aplicación como Introducción, Instrucciones y una Historia
RE04	Sesiones de Entrenamiento	El usuario podrá ingresar a sus sesiones de entrenamiento
RE05	Control de avance	El usuario podrá consultar su progreso en la aplicación en cualquier momento
RE06	Desbloqueo de sesiones	El usuario podrá desbloquear nuevas sesiones cuando cumpla con las anteriores
RE07	Visualizar contenido único por sesión	La app permitirá ver contenido único por sesión como: Frase, Lectura, Preguntas, Ejercicios
RE08	Gestor de objetivos	El usuario podrá agregar, modificar y verificar el cumplimiento de objetivos por sesión

Continua

RE	NR	Descripción
RE09	Evaluar Avance	La app permitirá realizar una evaluación individual del avance por cada sesión
RE10	Reiniciar Progreso	El usuario podrá reiniciar su avance en la aplicación, luego de una confirmación por el mismo
RE11	Información Adicional	La app permitirá visualizar información extra en una pantalla independiente.
RE12	Módulo Administrador	El administrador podrá ver el número de usuarios y su avance en la app.

4.1.2.2 Requisitos no funcionales

A continuación, se detalla los atributos no funcionales que deberá cumplir la aplicación, estos factores garantizarán que la misma tenga: rendimiento, seguridad, fiabilidad, disponibilidad y usabilidad del producto.

a) Rendimiento

La aplicación, que se ejecuta en el dispositivo móvil, garantizará un correcto desempeño, todos los procesos que se realizarán con el sistema de base de datos deberán estar diseñados para que la respuesta sea rápida y no comprometa el rendimiento del software.

b) Seguridad

Cada usuario debe contar con sus credenciales para acceder a la app y deben ser validas al momento de loguearse en la aplicación.

c) Fiabilidad

El almacenamiento de la información en el la base de datos deberá garantizar la integridad y correcta relación entre los datos de manera que facilite consultas posteriores.

d) Disponibilidad

La aplicación será diseñada para siempre estar disponible para los usuarios, esta condición dependerá del equipo donde será instalada la aplicación y del servicio de Internet.

e) Usabilidad

La aplicación deberá contar con diseños amigables e intuitivos para el usuario, además de poseer un diseño “Responsive” a fin de garantizar la adecuada visualización en múltiples dispositivos.

4.2 DISEÑO DEL PROTOTIPO

4.2.1 Diseño de la base de datos

El diseño de la base de datos se construirá tomando en cuenta los requerimientos establecidos y se utilizará un modelo de base de datos no relacional para lograr una representación que ayude a la solución del problema planteado en la **Figura 25 Diseño de base de datos** se muestra la estructura de los documentos(colecciones) correspondientes a la base de datos documental.

DISEÑO DE BASE DE DATOS

```
//collection usuarios
{
    "_id": ObjectId,
    "usuarioApellido": String,
    "usuarioLogin": String,
    "usuarioMail": String,
    "usuarioNombre": String,
    "usuarioPass": String
}
```

```
//collection: avances
{
    "_id": ObjectId,
    "estado": Boolean,
    "porcentaje": Double,
    "sesionID": NumberInt (1-12),
    "usuarioLogin": String
}
```

```
// collection: respuestas
{
    "_id": ObjectId,
    "usuarioLogin": String,
    "sesionID": NumberInt(1-12),
    "respuestas": [
        {
            "respuesta": String
        }
    ]
}
```

```
// collection: evaluaciones
{
    "_id": ObjectId,
    "usuarioLogin": String,
    "sesionID": NumberInt(1-12),
    "respuesta1": String,
    "respuesta2": String,
    "notas": String
}
```

```
// collection: compromisos
{
    "_id": ObjectId,
    "usuarioLogin": String,
    "sesionID": NumberInt(1-12),
    "compromisos": [
        {
            "numero": NumberInt,
            "descripcion": String,
            "cumplido": Boolean,
            "fechaInicio": DateTime,
            "fechaEstimada": Date,
            "fechaFin": DateTime
        }
    ]
}
```

Figura 25 Diseño de base de datos

4.2.2 Diagrama de casos de uso

DIAGRAMA DE CASOS DE USO

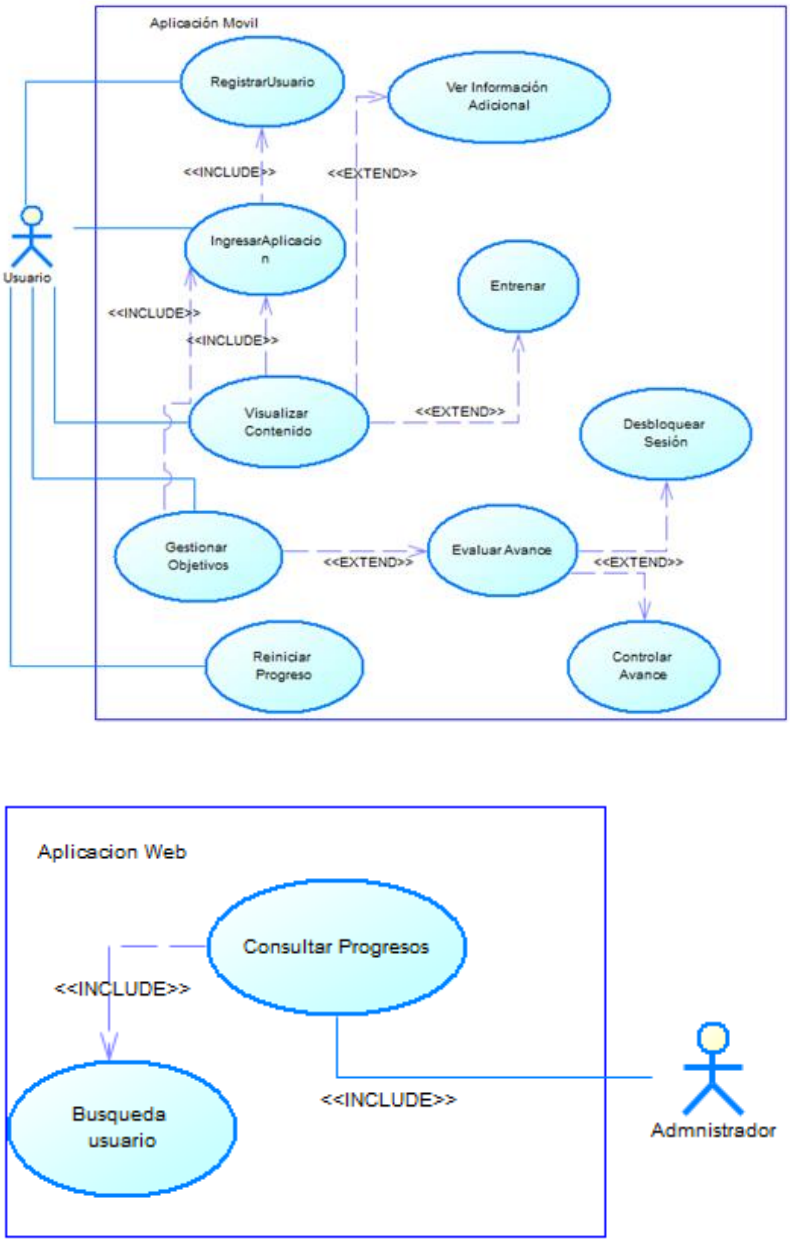

Figura 26 Diagrama de casos de uso

4.2.3 Diagrama de secuencia

DIAGRAMA SE SECUENCIA

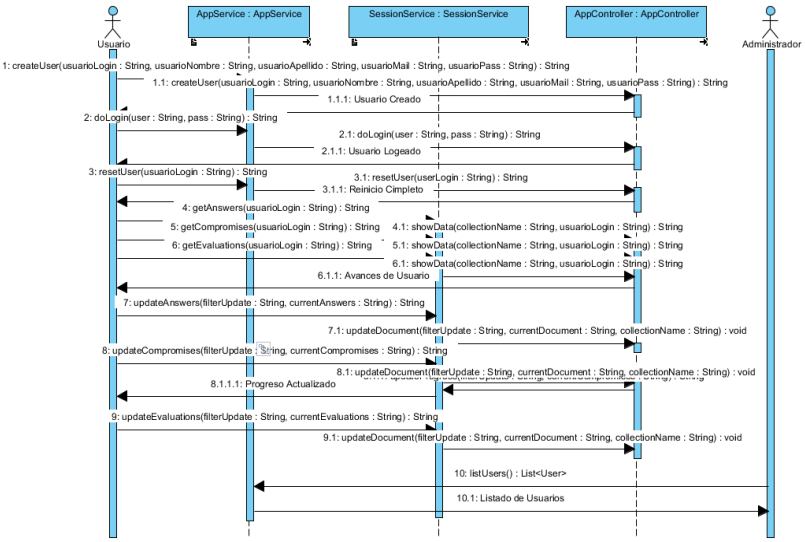

Figura 27 Diagrama se secuencia

4.2.4 Diagrama de clases

DIAGRAMA DE CLASES

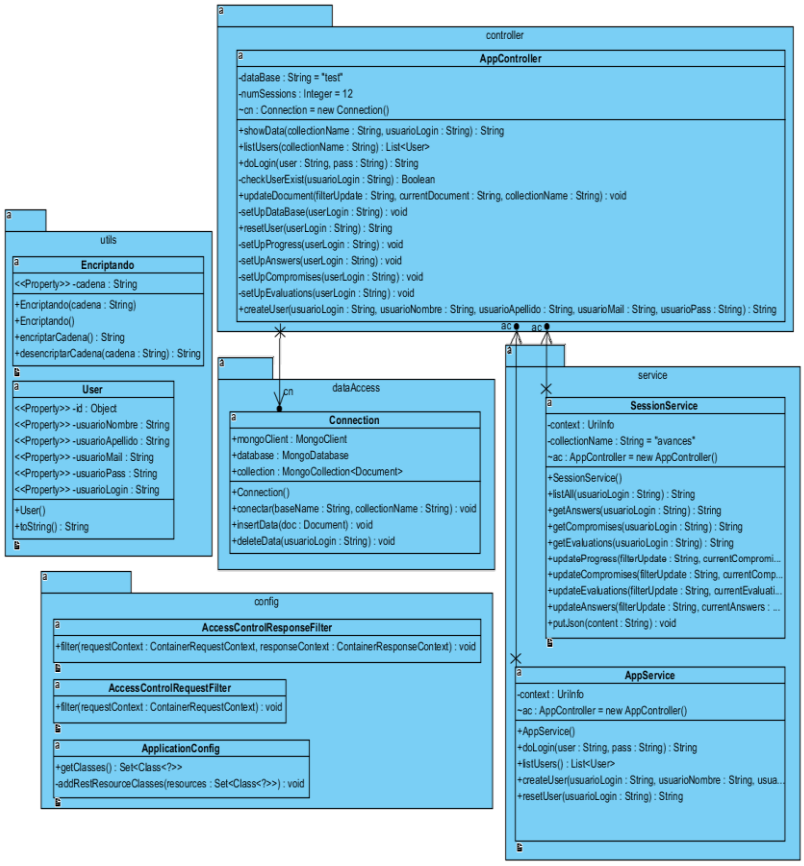

Figura 28 Diagrama de clases

4.2.5 Diagrama de arquitectura

La **Figura 29 Diagrama de arquitectura** muestra la arquitectura de la aplicación, en donde se tiene un usuario y un administrador mismos que por medio de sus respectivos dispositivos pueden acceder a la aplicación en Internet para conectarse a un servidor de aplicaciones mismo que accede a los datos almacenados.

DIAGRAMA DE ARQUITECTURA

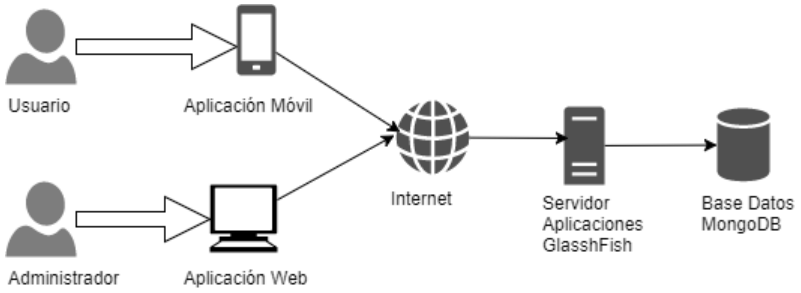

Figura 29 Diagrama de arquitectura

CAPITULO 5

PLANIFICACIÓN Y DESARROLLO

5.1 Planificación SCRUM

Según la metodología Scrum el punto de partida para el desarrollo, es la construcción de la pila de producto (Product Backlog) misma que se realizará en base a los requerimientos mencionados en el capítulo 4 en la sección ESPECIFICACIÓN DE REQUERIMIENTOS.

Para tener un mejor entendimiento del proceso es necesario definir un formato para el documento que se obtendrá de la construcción del Product Backlog para ello se escogió los siguientes campos en función de identificar a cada ítem del Product Backlog:

- **Id:** Identificador único del ítem en el listado.
- **Historia de usuario:** Descripción breve del requerimiento.
- **Estimado:** Tiempo estimado necesario para desarrollo.
- **Prioridad:** Grado de importancia que tendrá un ítem.
- **Criterio aceptación:** Breve descripción de la funcionalidad esperada.

En la

Tabla 18
Product Backlog podemos observar el Product Backlog generado en base a los requerimientos, cada historia de usuario engloba las tareas que serán abordadas en la etapa de iteraciones(Sprints) y que se evidenciarán en la pila de tareas (Sprint Backlog).

Tabla 18
Product Backlog

ID	HISTORIA USUARIO	ESTIMADO	PRIORIDAD	CRITERIO ACEPTACION
1	Registrar Usuario	2	5	Desplegar el formulario de registro verificar que no exista el usuario, Confirmar la creación o no del usuario
2	Iniciar sesión	1	5	Ingresar las credenciales de usuario, verificar si son correctas, en caso afirmativo permitir el ingreso del usuario, caso contrario mostrar mensaje de error
3	Visualizar contenido	1	3	Mostrar una Introducción, Instrucciones y una Historia, en pantallas diferentes para cada una
4	Sesiones de Entrenamiento	2	5	Visualizar las sesiones de entrenamiento del usuario su nombre y porcentaje de cumplimiento
5	Control de avance	2	4	Visualizar el número de sesión y el porcentaje faltante en cada una
6	Desbloqueo de sesiones	1	4	Al cumplir con el porcentaje mínimo de cumplimiento se debe desbloquear la siguiente sesión
7	Visualizar contenido único por sesión	2	3	Al ingresar en cada sesión se visualizará una Frase, luego se mostrará el contenido de Lectura, Preguntas o Ejercicios en un menú
8	Gestor de objetivos	2	5	Se visualizará los objetivos actuales y se mostrará la opción para agregar nuevos, permitiendo modificarlos siempre que estos no hayan sido cumplidos
9	Evaluar Avance	1	3	Se desplegará un formulario en donde se contestarán preguntas de evaluación, un espacio para agregar notas personales y se podrá ver el porcentaje de cumplimiento de la sesión en curso

Continua

ID	HISTORIA USUARIO	ESTIMADO	PRIORIDAD	CRITERIO ACEPTACION
10	Reiniciar Progreso	1	1	Se mostrará una descripción del reinicio, al acceder a esta opción se mostrará una confirmación, en caso de que se acepte se borraran todos los progresos actuales para empezar el programa desde cero
11	Información Adicional	1	2	En una pantalla se visualizará información extra totalmente independiente del resto de la aplicación
12	Módulo Administrador	2	4	Solo el administrador podrá ver el número de usuarios registrados en la app, así como sus avances y estado de sus sesiones

El Product Backlog debe ser refinado para facilitar las iteraciones es por ello que se ha definido dos indicadores para esta tarea:

- Tiempo Estimado: 1 – 4 semanas.
- Prioridad de Tarea: 1 – 5. Siendo 5 la más importante.

Una vez realizado el Product Backlog, se comienza con la etapa de iteraciones o Sprints. En esta etapa es donde se desarrollará la actividad de planificación (Sprint Planning), en donde los miembros del equipo eligen una o varias historias de usuario del Product Backlog para desarrollarlas en una sola iteración, este listado de tareas a cumplir es denominado como pila de tareas o Sprint Backlog.

5.2 Desarrollo Iteración 1

Para empezar el desarrollo de la aplicación se ha tomado las 3 primeras historias de usuario que en conjunto tienen una duración estimada de un mes, la

Tabla 19
Historias usuario Iteración 1 muestra las historias de usuario involucradas en esta iteración.

Tabla 19
Historias usuario Iteración 1

ID	HISTORIA USUARIO	ESTIMADO	PRIORIDAD	CRITERIO ACEPTACION
1	Registrar Usuario	2	5	Desplegar el formulario de registro verificar que no exista el usuario, Confirmar la creación o no del usuario
2	Iniciar sesión	1	5	Ingresar las credenciales de usuario, verificar si son correctas, en caso afirmativo permitir el ingreso del usuario, caso contrario mostrar mensaje de error
3	Visualizar contenido	1	3	Mostrar una Introducción, Instrucciones y una Historia, en pantallas diferentes para cada una

5.2.1 Sprint Backlog 1

Se procede a determinar las tareas necesarias para cumplir con las historias de usuario de la primera iteración. Para ello en la

Tabla 20
Sprint 1 se muestran las tareas asignadas, su responsable y el tiempo de entrega de cada una.

Tabla 20
Sprint 1

Sprint	Inicio		Duración (Semanas)	Semanas			
1	13-mar-17		4				
ID	Tarea	Responsable	N° Product Backlog	13-marzo 17-marzo	20-marzo 24-marzo	27-marzo 31-marzo	3-abril 7-abril
1	Implantación del modelo de datos	Víctor Añazco	1	x			
2	Codificación de back end de registro	Víctor Añazco	1	x			
3	Diseño de Interfaz de registro	Víctor Añazco	1	x			
4	Codificación de front end de registro	Víctor Añazco	1	x			
5	Pruebas de registro	Víctor Añazco	1		x		
6	Codificación de back end de login	Víctor Añazco	2		x		
7	Diseño de Interfaz de login	Víctor Añazco	2		x		
8	Codificación de front end de login	Víctor Añazco	2			x	
9	Pruebas de login	Víctor Añazco	2			x	
10	Codificación de back end de Introducción, Instrucciones e Historia	Víctor Añazco	3			x	
11	Diseño de interfaces de Introducción, Instrucciones e Historia	Víctor Añazco	3				X

Continua

Sprint	Inicio		Duración (Semanas)	Semanas			
1	13-mar-17		4				
ID	Tarea	Responsable	N° Product Backlog	13-marzo 17-marzo	20-marzo 24-marzo	27-marzo 31-marzo	3-abril 7-abril
12	Codificación de front end de Introducción, Instrucciones e Historia	Víctor Añazco	3				x
13	Pruebas de Introducción, Instrucciones, Historia	Víctor Añazco	3				x

5.2.2 Revisión 1

Para la revisión del Sprint es necesario listar todas las tareas que han sido completadas y aquellas que quedan pendientes por cada desarrollador, con el fin de medir el avance en las tareas asignadas. En la Tabla 21 Tareas Completadas Sprint 1 se detalla las tareas completadas en el Sprint 1.

Tabla 21
Tareas Completadas Sprint 1

ID	Responsable	Víctor Añazco	Número de Tareas	13
	Tarea	Iteración	Estado	Fecha
1	Implantación del modelo de datos	1	Completada	14-mar-17
2	Codificación de back end de registro	1	Completada	15-mar-17
3	Diseño de Interfaz de registro	1	Completada	16-mar-17
4	Codificación de front end de registro	1	Completada	17-mar-17

Continua

ID	Responsable	Víctor Añazco	Número de Tareas	13
	Tarea	**Iteración**	**Estado**	**Fecha**
5	Pruebas de registro	1	Completada	21-mar-17
6	Codificación de back end de login	1	Completada	22-mar-17
7	Diseño de Interfaz de login	1	Completada	24-mar-17
8	Codificación de front end de login	1	Completada	28-mar-17
9	Pruebas de login	1	Completada	29-mar-17
10	Codificación de back end de Introducción, Instrucciones e Historia	1	Completada	31-mar-17
11	Diseño de interfaces de Introducción, Instrucciones e Historia	1	Completada	4-abr-17
12	Codificación de front end de Introducción, Instrucciones e Historia	1	Completada	5-abr-17
13	Pruebas de Introducción, Instrucciones, Historia	1	Completada	7-abr-17

5.2.3 Demo 1

A continuación, se presenta el demo del Sprint 1 correspondiente a las siguientes funcionalidades:

5.2.3.1 Registrar Usuario

La funcionalidad de registro de usuario se encuentra en la interfaz de ingreso como se muestra en la **Figura 30 Interfaz de Ingreso** al ingresar en la opción Crear Usuario como se puede ver en la **Figura 31 Formulario registro usuario**, el mismo que cuenta con las respectivas validaciones y confirmaciones como se puede ver en la **Figura 32 Validaciones registro de usuario**.

INTERFAZ DE INGRESO

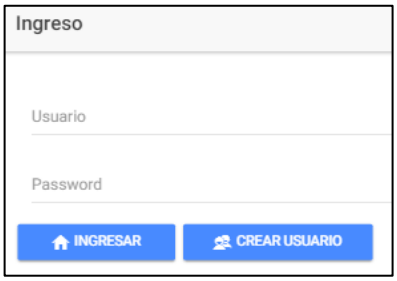

Figura 30 Interfaz de Ingreso

FORMULARIO REGISTRO USUARIO

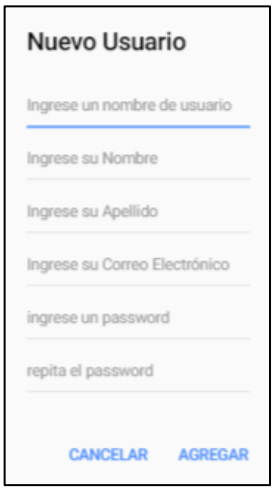

Figura 31 Formulario registro usuario

VALIDACIONES REGISTRO DE USUARIO

Figura 32 Validaciones registro de usuario

5.2.3.2 Iniciar Sesión

Para la funcionalidad inicio de sesión se cuenta con la interfaz de ingreso mostrada en la **Figura 30 Interfaz de Ingreso**, se accede a esta funcionalidad al seleccionar la opción Ingresar, misma que cuenta con sus respectivas validaciones como se puede ver en la **Figura 33 Validaciones Inicio Sesión**.

VALIDACIONES INICIO SESIÓN

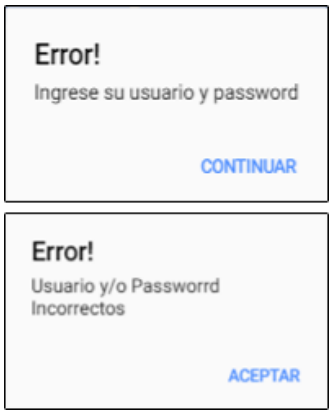

Figura 33 Validaciones Inicio Sesión

5.2.3.3 Visualizar Contenido

En esta funcionalidad se puede observar las tres pantallas mencionadas anteriormente en

Tabla 19
Historias usuario Iteración 1funcionando independientemente cada una.

Introducción que muestra un mensaje de Bienvenido junto con el nombre del usuario y la distribución de la introducción a la aplicación como podemos ver en la **Figura 34 Interfaz de Introducción**.

INTERFAZ DE INTRODUCCIÓN

Figura 34 Interfaz de Introducción

Instrucciones en esta pantalla se muestra la distribución de las instrucciones del programa de liderazgo como se puede observar en la **Figura 35 Interfaz de Instrucciones**.

INTERFAZ DE INSTRUCCIONES

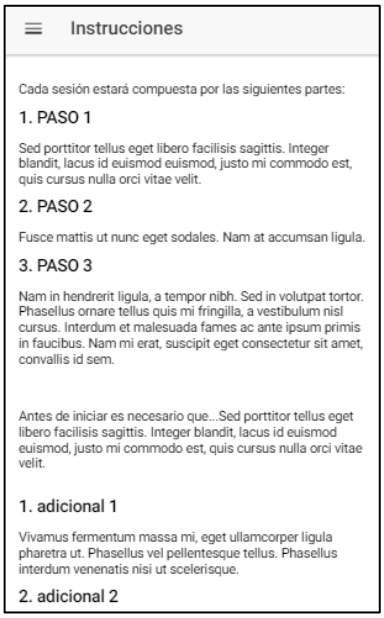

Figura 35 Interfaz de Instrucciones

En la pantalla Historia se observa la distribución de una historia para el programa como se ve en la **Figura 36 Interfaz de Historia**.

INTERFAZ DE HISTORIA

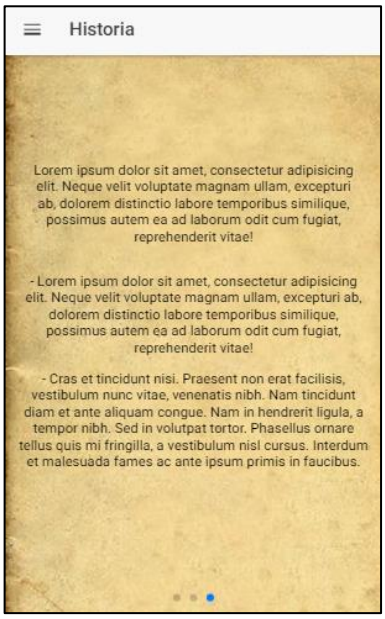

Figura 36 Interfaz de Historia

5.3 Desarrollo Iteración 2

Para la segunda iteración de la aplicación se ha tomado las 3 siguientes historias de usuario (4 - 6) que en conjunto tienen una duración estimada de 5 semanas, la Tabla 22
Historias usuario Iteración 2 muestra las historias de usuario involucradas en esta segunda iteración.

Tabla 22
Historias usuario Iteración 2

ID	HISTORIA USUARIO	ESTIMADO	PRIORIDAD	CRITERIO ACEPTACION
4	Sesiones de Entrenamiento	2	5	Visualizar las sesiones de entrenamiento del usuario su nombre y porcentaje de cumplimiento
5	Control de avance	2	4	Visualizar el número de sesión y el porcentaje faltante en cada una
6	Desbloqueo de sesiones	1	4	Al cumplir con el porcentaje mínimo de cumplimiento se debe desbloquear la siguiente sesión

5.3.1 Sprint Backlog 2

Se procede a determinar las tareas necesarias para cumplir con las historias de usuario de la segunda iteración. Para ello en la

Tabla 23

Sprint 2 se muestran las tareas asignadas, su responsable y el tiempo de entrega de cada una.

Tabla 23
Sprint 2

Sprint	Inicio		Duración (Semanas)	Semanas				
2	10-abr-17		5					
ID	Tarea	Responsable	N° Product Backlog	10-abril 14-	17-abril 21-	24-abril 28-	1-mayo 5-	8-mayo 12- mayo
1	Codificación de back end de sesiones de entrenamiento	Víctor Añazco	4	x				
2	Diseño de Interfaz de sesiones de entrenamiento	Víctor Añazco	4	x				
3	Codificación de front end de sesiones de entrenamiento	Víctor Añazco	4	x				
4	Pruebas de sesiones de entrenamiento	Víctor Añazco	4		x			
5	Codificación de back end de control de avance	Víctor Añazco	5		x			
6	Diseño de Interfaz de control de avance	Víctor Añazco	5		x			
7	Codificación de front end de control de avance	Víctor Añazco	5			x		
8	Pruebas de control de avance	Víctor Añazco	5			x		
9	Codificación de back end de desbloqueo de sesiones	Víctor Añazco	6				x	Continua

Sprint	Inicio		Duración (Semanas)	Semanas				
2	10-abr-17		5					
ID	Tarea	Responsable	N° Product Backlog	10-abril 14-	17-abril 21-	24-abril 28-	1-mayo 5-	8-mayo 12- mayo
10	Diseño de interfaces de desbloqueo de sesiones	Víctor Añazco	6				x	
11	Codificación de front end de desbloqueo de sesiones	Víctor Añazco	6					x
12	Pruebas de desbloqueo de sesiones	Víctor Añazco	6					x

5.3.2 Revisión 2

Para la revisión del Sprint es necesario listar todas las tareas que han sido completadas y aquellas que quedan pendientes por cada desarrollador, con el fin de medir el avance en las tareas asignadas. En la

Tabla 24
Tareas Completadas Sprint 2 se detalla las tareas completadas en el Sprint 2.

Tabla 24
Tareas Completadas Sprint 2

ID	**Responsable**	Víctor Añazco	**Número de Tareas**	12
	Tarea	**Iteración**	**Estado**	**Fecha**
1	Codificación de back end de sesiones de entrenamiento	2	Completada	11-abr-17
2	Diseño de Interfaz de sesiones de entrenamiento	2	Completada	13-abr-17
3	Codificación de front end de sesiones de entrenamiento	2	Completada	14-abr-17
4	Pruebas de sesiones de entrenamiento	2	Completada	18-abr-17
5	Codificación de back end de control de avance	2	Completada	20-abr-17
6	Diseño de Interfaz de control de avance	2	Completada	21-abr-17
7	Codificación de front end de control de avance	2	Completada	26-abr-17
8	Pruebas de control de avance	2	Completada	28-abr-17
9	Codificación de back end de desbloqueo de sesiones	2	Completada	3-may-17
10	Diseño de interfaces de desbloqueo de sesiones	2	Completada	5-may-17
11	Codificación de front end de desbloqueo de sesiones	2	Completada	9-may-17
12	Pruebas de desbloqueo de sesiones	2	Completada	12-may-17

5.3.3 Demo 2

A continuación, se presenta el demo del Sprint 2 correspondiente a las siguientes funcionalidades:

5.3.3.1 Sesiones de Entrenamiento

Para esta funcionalidad se cuenta con una pantalla en donde se muestra las sesiones de entrenamiento por usuario, su nombre y su porcentaje de cumplimiento como se puede ver en la **Figura 37 Interfaz de Sesiones de Entrenamiento**.

INTERFAZ DE SESIONES DE ENTRENAMIENTO

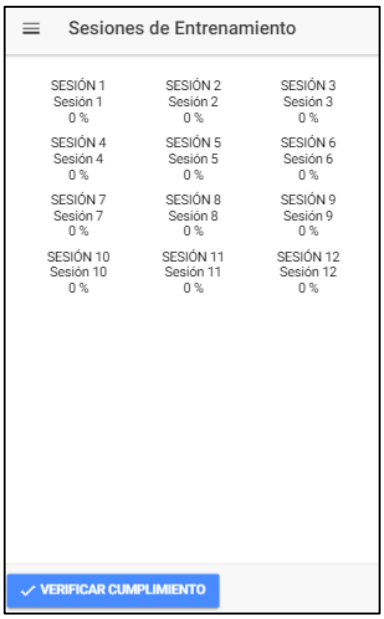

Figura 37 Interfaz de Sesiones de Entrenamiento

5.3.3.2 Control de avance

Se puede acceder a esta funcionalidad por medio de la opción Verificar cumplimiento que se encuentra en la **Figura 37 Interfaz de Sesiones de Entrenamiento**.

En esta interfaz se puede ver el número de cada sesión y el porcentaje faltante de cada una como se muestra en la **Figura 38 Interfaz de Control de avance**.

INTERFAZ DE CONTROL DE AVANCE

Figura 38 Interfaz de Control de avance

5.3.3.3 Desbloqueo de sesiones

Esta funcionalidad permite desbloquear nuevas sesiones al alcanzar un porcentaje mínimo de cumplimiento en la sesión actual, si se accede a una sesión bloqueada se muestra un mensaje de bloqueo, estos mensajes se pueden ver en la **Figura 39 Mensajes de cumplimiento de sesiones**.

MENSAJES DE CUMPLIMIENTO DE SESIONES

Figura 39 Mensajes de cumplimiento de sesiones

5.4 Desarrollo Iteración 3

En la tercera iteración de la aplicación se ha tomado las 3 siguientes historias de usuario (7 - 9) que en conjunto tienen una duración estimada de 5 semanas, la Tabla 25

Historias de usuario Iteración 3muestra las historias de usuario involucradas en esta segunda iteración.

Tabla 25
Historias de usuario Iteración 3

ID	HISTORIA USUARIO	ESTIMADO	PRIORIDAD	CRITERIO ACEPTACION
7	Visualizar contenido único por sesión	2	3	Al ingresar en cada sesión se visualizará una Frase, luego se mostrará el contenido de Lectura, Preguntas o Ejercicios en un menú
8	Gestor de objetivos	2	5	Se visualizará los objetivos actuales y se mostrará la opción para agregar nuevos, permitiendo modificarlos siempre que estos no hayan sido cumplidos
9	Evaluar Avance	1	3	Se desplegará un formulario en donde se contestarán preguntas de evaluación, un espacio para agregar notas personales y se podrá ver el porcentaje de cumplimiento de la sesión en curso

5.4.1 Sprint Backlog 3

Se procede a determinar las tareas necesarias para cumplir con las historias de usuario de la tercera iteración. Para ello en la se muestran las tareas asignadas, su responsable y el tiempo de entrega de cada una.

Tabla 26
Sprint 3

Sprint	Inicio		Duración (Semanas)	Semanas				
3	10-may-17		5					
ID	Tarea	Responsable	N° Product Backlog	15-mayo 19-mayo	22-mayo 26-mayo	29-mayo 2-junio	5-junio 9-junio	12-junio 16-junio
1	Codificación de back end de Visualizar contenido único por sesión	Víctor Añazco	7	x				
2	Diseño de Interfaz de Visualizar contenido único por sesión	Víctor Añazco	7	x				
3	Codificación de front end de Visualizar contenido único por sesión	Víctor Añazco	7		x			
4	Pruebas de Visualizar contenido único por sesión	Víctor Añazco	7		x			
5	Codificación de back end de Gestor de objetivos	Víctor Añazco	8			x		
6	Diseño de Interfaz de Gestor de objetivos	Víctor Añazco	8			x		
7	Codificación de front end de Gestor de objetivos	Víctor Añazco	8			x		
8	Pruebas de Gestor de objetivos	Víctor Añazco	8				x	

Continua

Sprint	Inicio		Duración (Semanas)	Semanas				
3	10-may-17		5					
ID	Tarea	Responsable	N° Product Backlog	15-mayo 19-mayo	22-mayo 26-mayo	29-mayo 2-junio	5-junio 9-junio	12-junio 16-junio
9	Codificación de back end de Evaluar Avance	Víctor Añazco	9				x	
10	Diseño de interfaces de Evaluar Avance	Víctor Añazco	9					x
11	Codificación de front end de Evaluar Avance	Víctor Añazco	9					x
12	Pruebas de Evaluar Avance	Víctor Añazco	9					x

5.4.2 Revisión 3

Para la revisión del Sprint es necesario listar todas las tareas que han sido completadas y aquellas que quedan pendientes por cada desarrollador, con el fin de medir el avance en las tareas asignadas. En la se detalla las tareas completadas en el Sprint 3.

Tabla 27
Tareas Completadas Sprint 3

ID	Responsable	Víctor Añazco	Número de Tareas	12
	Tarea	Iteración	Estado	Fecha
1	Codificación de back end de Visualizar contenido único por sesión	3	Completada	16-may-17

Continua

ID	Responsable	Víctor Añazco	Número de Tareas	12
	Tarea	**Iteración**	**Estado**	**Fecha**
2	Diseño de Interfaz de Visualizar contenido único por sesión	3	Completada	19-may-17
3	Codificación de front end de Visualizar contenido único por sesión	3	Completada	24-may-17
4	Pruebas de Visualizar contenido único por sesión	3	Completada	26-may-17
5	Codificación de back end de Gestor de objetivos	3	Completada	1-jun-17
6	Diseño de Interfaz de Gestor de objetivos	3	Completada	2-jun-17
7	Codificación de front end de Gestor de objetivos	3	Completada	7-jun-17
8	Pruebas de Gestor de objetivos	3	Completada	8-jun-17
9	Codificación de back end de Evaluar Avance	3	Completada	9-jun-17
10	Diseño de interfaces de Evaluar Avance	3	Completada	13-jun-17
11	Codificación de front end de Evaluar Avance	3	Completada	15-jun-17
12	Pruebas de Evaluar Avance	3	Completada	16-jun-17

5.4.3 Demo 3

A continuación, se presenta el demo del Sprint 3 correspondiente a las siguientes funcionalidades:

5.4.3.1 Visualizar contenido único por sesión

Esta funcionalidad permite ver una frase al ingresar en cada sesión, además de una lectura, preguntas de evaluación y ejercicios, todo dentro de un menú como se muestra en la **Figura 40 Contenido único por sesión**.

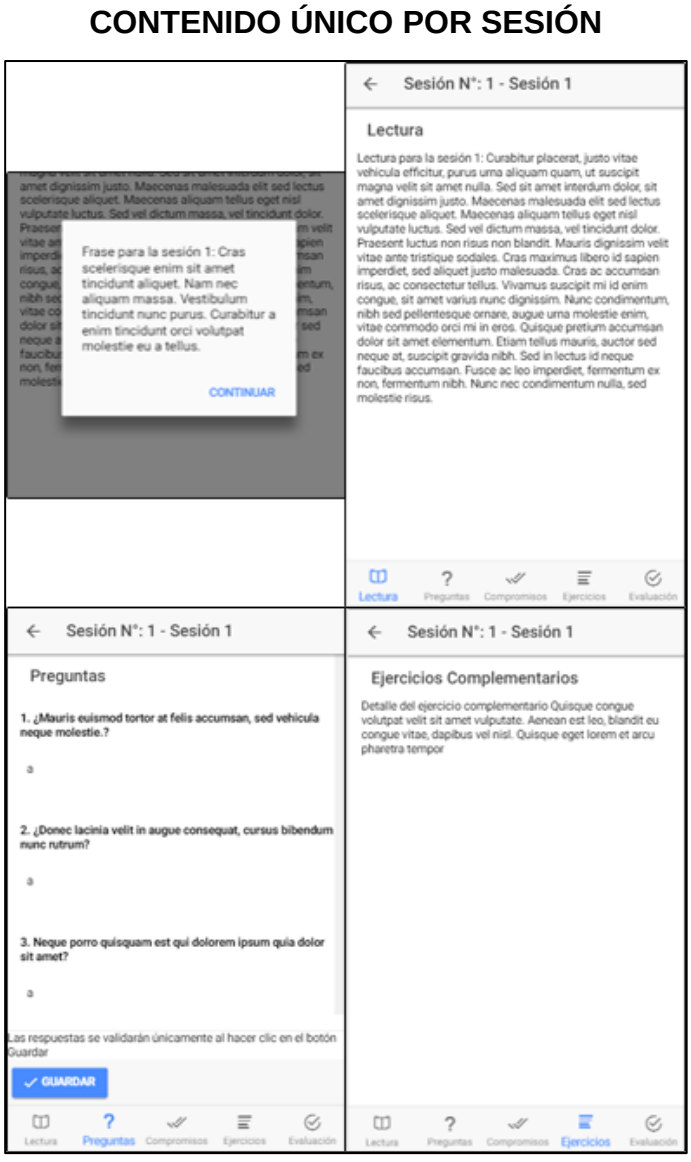

Figura 40 Contenido único por sesión

5.4.3.2 Gestor de objetivos

Esta funcionalidad permite crear objetivos y modificarlos siempre que no hayan sido cumplidos se encuentra en la interfaz de compromisos mostrada en la Figura 41 Interfaz de Compromisos, para acceder a ella se debe ingresar a la opción Agregar donde se despliega un formulario que nos permite ingresar un nuevo compromiso como podemos ver en la Figura 42 Formulario Nuevo Compromiso.

INTERFAZ DE COMPROMISOS

Figura 41 Interfaz de Compromisos

FORMULARIO NUEVO COMPROMISO

Figura 42 Formulario Nuevo Compromiso

Al crear un nuevo compromiso es validado y si es correcto se muestra una confirmación como se puede ver en la **Figura 43 Validaciones de Compromisos**, luego se agrega a una lista mostrando la fecha de creación, fecha estimada y fecha de cumplimiento, así como un check que indica el cumplimiento o no del mismo tal como indica la **Figura 44 Lista de compromisos**.

VALIDACIONES DE COMPROMISOS

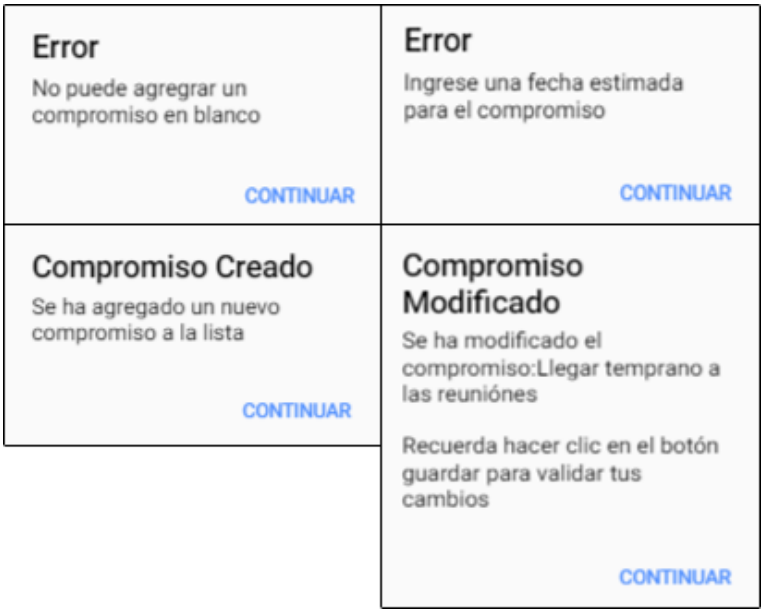

Figura 43 Validaciones de Compromisos

LISTA DE COMPROMISOS

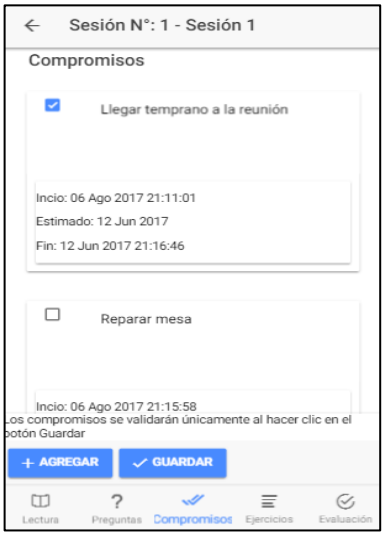

Figura 44 Lista de compromisos

5.4.3.3 Evaluar Avance

Esta funcionalidad permite contestar preguntas de evaluación por cada sesión, así como también guardar notas personales y observar el porcentaje de cumplimiento en la sesión actual tal como indica la **Figura 45 Interfaz de Evaluación**.

INTERFAZ DE EVALUACIÓN

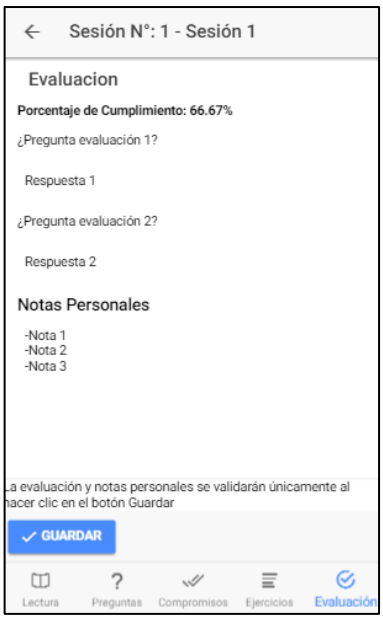

Figura 45 Interfaz de Evaluación

5.5 Desarrollo Iteración 4

Para la cuarta última iteración de la aplicación se ha tomado las 3 últimas historias de usuario (10 - 12) que en conjunto tienen una duración estimada de 4 semanas, la Tabla 25
Historias de usuario Iteración 3muestra las historias de usuario involucradas en esta segunda iteración.

Tabla 28
Historias de usuario Iteración 4

ID	HISTORIA USUARIO	ESTIMADO	PRIORIDAD	CRITERIO ACEPTACION
10	Reiniciar Progreso	1	1	Se mostrará una descripción del reinicio, al acceder a esta opción se mostrará una confirmación, en caso de que se acepte se borraran todos los progresos actuales para empezar el programa desde cero
11	Información Adicional	1	2	En una pantalla se visualizará información extra totalmente independiente del resto de la aplicación
12	Módulo Administrador	2	4	Solo el administrador podrá ver el número de usuarios registrados en la app, así como sus avances y estado de sus sesiones

5.5.1 Sprint Backlog 4

Se procede a determinar las tareas necesarias para cumplir con las historias de usuario de la tercera iteración. Para ello en la Tabla 29
Sprint 4 se muestran las tareas asignadas, su responsable y el tiempo de entrega de cada una.

Tabla 29
Sprint 4

Sprint	Inicio		Duración (Semanas)	Semanas			
4	19-jun-17		4				
ID	Tarea	Responsable	N° Product Backlog	19-junio 23-junio	26-junio 30-junio	3-julio 7-julio	10-julio 14-julio
1	Codificación de back end de Reiniciar Progreso	Víctor Añazco	10	x			
2	Diseño de Interfaz de Reiniciar Progreso	Víctor Añazco	10	x			
3	Codificación de front end de Reiniciar Progreso	Víctor Añazco	10	x			
4	Pruebas de Reiniciar Progreso	Víctor Añazco	10		x		
5	Codificación de back end de Información Adicional	Víctor Añazco	11		x		
6	Diseño de Interfaz de Información Adicional	Víctor Añazco	11			x	
7	Codificación de front end de Información Adicional	Víctor Añazco Continua	11			x	

Sprint	Inicio		Duración (Semanas)	Semanas			
4	19-jun-17		4				
ID	Tarea	Responsable	N° Product Backlog	19-junio 23-junio	26-junio 30-junio	3-julio 7-julio	10-julio 14-julio
8	Pruebas de Información Adicional	Víctor Añazco	11			x	
9	Codificación de back end de Módulo Administrador	Víctor Añazco	12				x
10	Diseño de interfaces de Módulo Administrador	Víctor Añazco	12				x
11	Codificación de front end de Módulo Administrador	Víctor Añazco	12				x
12	Pruebas de Módulo Administrador	Víctor Añazco	12				x

5.5.2 Revisión 4

Para la revisión del Sprint es necesario listar todas las tareas que han sido completadas y aquellas que quedan pendientes por cada desarrollador, con el fin de medir el avance en las tareas asignadas. En la Tabla 30 Tareas Completadas Sprint 4 se detalla las tareas completadas en el Sprint 4.

Tabla 30
Tareas Completadas Sprint 4

ID	**Responsable**	Víctor Añazco	**Número de Tareas**	12
	Tarea	**Iteración**	**Estado**	**Fecha**
1	Codificación de back end de Reiniciar Progreso	4	Completada	20-jun-17
2	Diseño de Interfaz de Reiniciar Progreso	4	Completada	23-jun-17
3	Codificación de front end de Reiniciar Progreso	4	Completada	26-jun-17
4	Pruebas de Reiniciar Progreso	4	Completada	28-jun-17
5	Codificación de back end de Información Adicional	4	Completada	30-jun-17
6	Diseño de Interfaz de Información Adicional	4	Completada	4-jul-17
7	Codificación de front end de Información Adicional	4	Completada	6-jul-17
8	Pruebas de Información Adicional	4	Completada	7-jul-17
9	Codificación de back end de Módulo Administrador	4	Completada	11-jul-17
10	Diseño de interfaces de Módulo Administrador	4	Completada	12-jul-17

Continua

ID	Responsable	Víctor Añazco	Número de Tareas	12
	Tarea	**Iteración**	**Estado**	**Fecha**
11	Codificación de front end de Módulo Administrador	4	Completada	13-jul-17
12	Pruebas de Módulo Administrador	4	Completada	14-jul-17

5.5.3 Demo 4

A continuación, se presenta el demo del Sprint 4 correspondiente a las siguientes funcionalidades:

5.5.3.4 Reiniciar Progreso

Para esta funcionalidad se cuenta con una pantalla que describe lo que hace proceso de reinicio del progreso como nos muestra la **Figura 46 Interfaz de Reiniciar Progreso**, al ingresar en la opción reiniciar se mostrará una confirmación que en caso de ser aceptada procederá a eliminar los progresos actuales y empezar el programa desde cero como podemos ver en la **Figura 47 Confirmación Reinicio**.

INTERFAZ DE REINICIAR PROGRESO

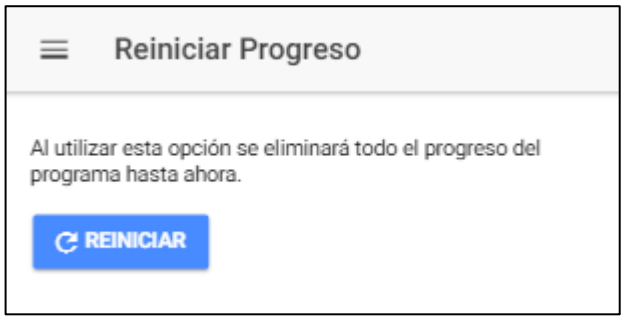

Figura 46 Interfaz de Reiniciar Progreso

CONFIRMACIÓN REINICIO

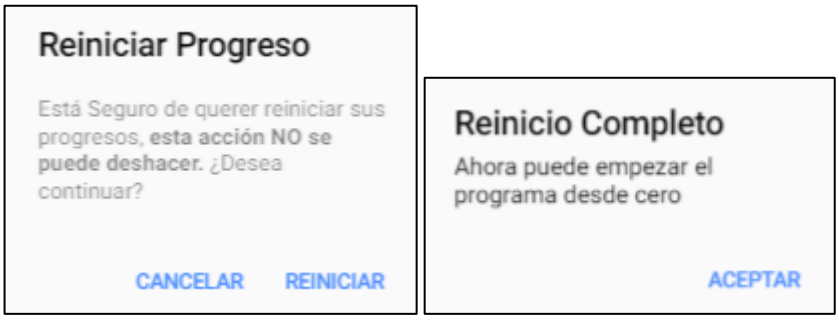

Figura 47 Confirmación Reinicio

5.5.3.5 Información Adicional

La presente funcionalidad permite mostrar en una pantalla información extra totalmente independiente del resto de la aplicación como se puede ver en la **Figura 48 Interfaz Información Adicional**

INTERFAZ INFORMACIÓN ADICIONAL

Figura 48 Interfaz Información Adicional

5.5.3.6 Módulo Administrador

Este módulo es únicamente para el administrador del sistema y le permite visualizar el número de usuarios registrados en la aplicación, sus nombres, apellidos, login y correo electrónico, además de sus progresos y el estado de sus sesiones de manera individual tal como se observa en la **Figura 49 Módulo de Administrador**.

Figura 49 Módulo de Administrador

CAPITULO 6

CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones

- El modelo de calidad construido permitió visualizar y analizar las características de cada framework ayudando al momento de elegir el más acorde con las necesidades de la empresa SOFYA S.A.
- Gracias al uso de la Matriz FODA fue posible tener una visión más clara de las ventajas y desventajas de cada framework.
- Se eligió a IONIC como framework para el desarrollo ya que tiene ventaja sobre React en: capacidad para ser entendido, aprendido y probado, además de poseer más ventajas en lo que se refiere a codificación y documentación factores muy importantes al momento de desarrollar una aplicación.
- La combinación de metodologías SCRUM y XP facilitó el desarrollo de la aplicación al brindar un mejor control de avances con cada sprint e iteración durante el proceso.

6.2 Recomendaciones

- Al momento de realizar un modelo de evaluación tomar en cuenta las características más relevantes para no tener una sobrecarga de información y un grado alto de complejidad para realizar un análisis.
- Aplicar una matriz FODA a un número reducido de objetos de estudio para ayudar a la toma de decisiones.
- Revisar constantemente la documentación del framework para familiarizarse con el mismo y estar al pendiente de posibles actualizaciones.
- Durante el proceso de desarrollo llevar un buen control de actividades realizadas y tareas pendientes para poder obtener mejores resultados al final del proceso.

BIBLIOGRAFÍA

Aggarwal, A. (4 de Abril de 2017). *medium*. Obtenido de https://medium.com/@ankushaggarwal/ionic-vs-react-native-3eb62f8943f8

Álvarez García, A., de las Heras del Dedo, R., & Lasa Gómez, C. (2012). *Métodos Ágiles y Scrum.* Madrid, España: Anaya Multimedia.

anibalgoicochea. (26 de Julio de 2013). Obtenido de anibalgoicochea: https://anibalgoicochea.com/2013/07/26/aplicaciones-nativas-aplicaciones-web-y-aplicaciones-hibridas/

C. Rodríguez, H. Enríquez. (octubre de 2014). Características del desarrollo en Frameworks multiplataforma. *15*(30), 101-117.

Chromatic. (2013). *Extreme Programming Pocket Guide.*

Díaz Aguilera, R. (18 de Abril de 2016). *adictosaltrabajo*. Obtenido de https://www.adictosaltrabajo.com/tutoriales/empezando-con-ionic-2/

IEEE. (10 de 10 de 2014). *IEEE Standards*. Obtenido de www.ieee.org

ISO 25000. (2017). *iso25000.com*. Obtenido de http://iso25000.com/index.php/normas-iso-25000

ISO 25010. (2017). *iso25000.com*. Obtenido de http://iso25000.com/index.php/normas-iso-25000/iso-25010?limit=3&limitstart=0

Loaiza Morales, A. A. (2012). Modelos de Calidad de Software.

Massart, F. (18 de Enero de 2017). *codementor*. Obtenido de https://www.codementor.io/fmcorz/react-native-vs-ionic-du1087rsw

matrizfoda.com. (2017). Obtenido de http://www.matrizfoda.com/dafo/

mobile-frameworks-comparison-chart.com. (s.f.). Obtenido de http://mobile-frameworks-comparison-chart.com/

Molina, N. (8 de Enero de 2017). *www.ion-book.com*. Obtenido de https://www.ion-book.com/blog/tips/ionic-vs-nativescript/

pixelcrayons. (6 de Marzo de 2017). Obtenido de http://www.pixelcrayons.com/blog/mobile/7-best-hybrid-app-development-frameworks-for-2017/

Pressman, R. (2010). *Ingeniería del Software. .* México, México, México: McGraw-Hill,.

programacion.net. (17 de Junio de 2016). Obtenido de http://programacion.net/articulo/comparacion_de_frameworks_para_desarrollar_apps_para_movil_hibridas_1370

Riehle, D. (2013). *Framework Design: A Role Modeling Approach.*

Scrum-Alliance. (2014). *Scrum Alliance*. Obtenido de https://www.scrumalliance.org/why-scrum/core-scrum-values-roles

uml.org. (22 de 12 de 2012). Obtenido de http://www.uml.org/what-is-uml.htm