Universidad del Azuay

Facultad de Ciencias de la Administración

Escuela de Ingeniería de Sistemas

"Análisis de frameworks para el desarrollo de aplicaciones móviles en la plataforma Android."

Trabajo de graduación previo a la obtención del título de:

Ingeniero de Sistemas

Autores:

Henry Solano

Israel Torres

Directora:

Ing. Catalina Astudillo.

Cuenca Ecuador

2013

DEDICATORIA

La presente monografía está dedicada en primer lugar a Dios que es el que me ha permitido llegar a culminar con éxito mi carrera, a mis padres que son el pilar fundamental de mi vida y me han dado la oportunidad de estudiar, a mi esposa y mi hijo que son sin duda el estímulo diario para salir adelante y cumplir cada una de mis metas y finalmente para todos mis amigos y familiares que siempre estuvieron a mi lado en las buenas y en las malas durante este periodo en especial para mi gran amigo que ya no está presente pero que siempre supo que era capaz de llegar a la meta (-.-).

Israel.

Este trabajo le dedico a mis padres, quienes han sido siempre un gran apoyo en todo momento y me han dado la oportunidad de poderme desarrollar profesionalmente.

A mis hermanas quienes han estado siempre pendientes de mi bienestar ya que, me han brindado su apoyo incondicional en todo momento.

De igual manera a mi compañero de monografía ya que junto a él hemos podido culminar con éxitos nuestros estudios.

Y para todas las personas que nos prestaron su ayuda para poder realizar este trabajo.

Henry.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS E ILUSTRACIONES

RESUMEN

El presente trabajo es un análisis de calidad de frameworks para el desarrollo de aplicaciones móviles en la plataforma android, de esta manera se realiza un estudio de varios modelos de calidad, definiendo el más óptimo para la evaluación de las herramientas. Una vez elegido el modelo de calidad se definen métricas para la medición de las características de tres frameworks diferentes, obteniendo como resultado la herramienta cuyas características ofrecen una mejor orientación para el desarrollo de aplicaciones para la plataforma android. Finalmente se realiza el desarrollo de una aplicación elaborada en el framework elegido, sin dejar de lado, el análisis, diseño y programación de dicha aplicación.

ABSTRACT

"ANALYSIS OF FRAMEWORKS FOR THE DEVELOPMENT OF MOBILE APPLICATIONS IN THE ANDROID PLATFORM"

The present work is an analysis of the quality of the frameworks for the development of mobile applications in the android platform. Consequently, a study of several quality models is carried out in order to define the most appropriate one for the assessment of the tools. Once the quality model is chosen, the software metrics are defined in order to measure the features of three different frameworks. As a result, we obtained the tools that offer better orientation for the development of the applications for android platform. Finally, we develop an application created in the chosen framework through its analysis, design, and programming.

UNIVERSIDAD DEL AZUAY
DPTO. IDIOMAS

Translated by,
Diana Lee Rodas

INTRODUCCIÓN

En la actualidad existe una gran tendencia al uso de dispositivos móviles inteligentes como: teléfonos celulares, tablets, ipods, etc. Los cuales por medio de diferentes aplicaciones brindan una gran cantidad de prestaciones a los usuarios.

Por estas razones es importante que una aplicación sea muy bien elaborada y agradable al usuario, lo que implica: el análisis, diseño y desarrollo de la misma.

Para poder lograr una correcta elaboración, una herramienta que facilite la realización de la aplicación contemplando el análisis y diseño de la misma.

Siendo así, en el presente trabajo se desarrolla un control de calidad aplicado a herramientas para el desarrollo de aplicaciones para la plataforma Android, así como el proceso de análisis y diseño de la aplicación.

Comenzando por el estudio de varios modelos de calidad, se selecciona el que permita evaluar de mejor manera este tipo de software para desarrollo. En segundo lugar se realiza el estudio de calidad en diferentes herramientas aplicando el modelo obteniendo, y obteniendo el software que brinde las mejores prestaciones para el desarrollo de aplicaciones.

Finalmente se procede a realizar el análisis, diseño y programación de una aplicación móvil que se la desarrolla con la herramienta Adobe Flash Builder, cuyos resultados fueron sido los mejores en el control de calidad.

CAPÍTULO I

1. Modelos de calidad de software para el desarrollo de aplicaciones móviles.

1.1. Definición de los modelos de calidad.

1.1.1 Modelo de McCall

El Modelo de McCall es el primer modelo de calidad desde que se inició la ingeniería de software, fue desarrollado en 1977 por Jim McCall. Debido a su forma de plantear el análisis del software es muy utilizado en la actualidad, está basado en varios factores de calidad orientado a satisfacer las necesidades tanto de desarrolladores como la de los usuarios.

Para McCall los factores que influyen en la calidad son demasiado abstractos y contienen un sinnúmero de características, debido a esto utiliza criterios de calidad con los que relaciona atributos internos y externos, de la misma forma es necesario descomponer un nivel más, teniendo como resultado que cada uno de los criterios tiene varias métricas y que estas son los atributos que se medirán directamente y a muy bajo nivel.

McCall en su modelo de calidad habla sobre diversos factores que afectan a la calidad, estos factores son 11, los cuales están dentro de varios ejes o puntos de vista detallados a continuación.

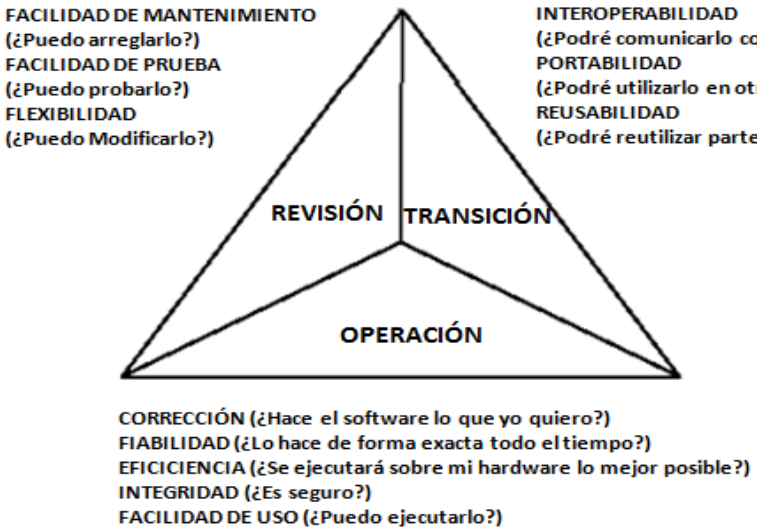

Gráfico 01 Factores de Calidad del Modelo de McCall

Puntos de Vista	Factores	Criterios	Descripción
Operación del Producto	Facilidad de Uso	Facilidad de Comunicación	Atributos que determinan la facilidad de uso que tiene el software
		Faciidad de Operación	Atributos del software que permiten asimilar fácilmente entradas-salidas
		Faciidad de Aprendizaje	Atributos del software que permiten al usuario familiarizarse con el software
		Formación	Capacidad del software para que nuevos usuarios puedan aplicar el sistema
	Integridad	Control de Acceso	Atributos del software que permiten controlar la información que maneja y el acceso al software
		Facilidad de auditoría	Atributos del software que facilitan realizar una auditoría al control de acceso al software
		Seguridad	Mecanismo capaz de controlar el acceso al software así como la información del mismo
	Corrección	Completitud	Atributos del software que proporcionan la implementación completa de todas las funciones requeridas.
		Consistencia	Atributos del software que proporcionan uniformidad en las técnicas y notaciones de diseño e implementación.
		Trazabilidad	Atributos del software que proporcionan una traza desde los requisitos a la implementación con respecto a un entorno operativo concreto.
	Fiabilidad	Precisión	Es el grado de precisión que proporciona el software al momento de realizar cálculos
		Consistencia	
		Tolerancia a fallos	Es aquella continuidad que ofrece el software bajo condiciones críticas
		Modularidad	Proporciona una estructura de módulos independientes
		Simplicidad	Proporciona la posibilidad de implementar funciones de la manera más sencilla posible
		Exactitud	Al igual que la precisión permite realizar cálculos de forma efectiva
	Eficiencia	Eficiencia en ejecución	Minimiza el tiempo de procesamiento
		Eficiencia de almacenamiento	Minimiza el espacio de almacenaiento necesario

Gráfico 02 Operaciones del producto del Modelo de McCall

Puntos de Vista	Factores	Criterios	Descripción
Revisión del Producto	Facilidad de Mantenimiento	Concisión	Implementar funciones con la menor cantidad de código fuente
		Modularidad	
		Simplicidad	
		Consistencia	
		Auto Descripción	Proporciona explicaciones sobre las funciones
	Facilidad de Prueba	Instrumentación	Atributos del software que posibilitan la observación del comportamiento del software durante su ejecución para facilitar las mediciones del uso o la identificación de errores.
		Modularidad	
		Simplicidad	
		Auto Descripción	
	Flexibilidad	Capacidad de Expansión	Permite expandir el software en cuanto a funcionalidad y datos.
		Auto Descripción	
		Modularidad	
		Generalidad	Proporciona amplitud a las funciones implementadas.
	Reusabilidad	Independencia entre sistema y software	Atributos de software que determian la dependencia que existe con respecto al entorno operativo.
		Auto Descripción	
		Generalidad	
		Modularidad	
		Independencia del hardware	Atributos que determina la dependencia existente con el hardware
	Interoperabilidad	Compatibilidad de comunicaciones	Atributos del software que posibilitan el uso de protocolos de comunicación e interfaces estándar.
		Modularidad	
		Compatibilidad de datos	Permiten el uso de representaciones de datos estándar.
		Estandarización de los datos	Uso de estructuras de datos de tipos estándar a lo largo de todo el programa.
	Portabilidad	Auto Descripción	
		Modularidad	
		Independencia entre sistema y software	
		Independencia del hardware	

Gráfico 03 Revisiones del producto del Modelo de McCall.

1.1.2 Modelo de Dromey

Fue presentado por Sr. R. Geoff Dromey en el año de 1996, es derivado del modelo de McCall, este modelo se basa en la idea de que cada producto debe ser evaluado independientemente de los demás, es así que trata la relación que existe entre los atributos y los sub-atributos; se enfoca en la facilidad y practicidad de evaluar la determinación de requerimientos así como el diseño y la implementación.

"Dromey sugiere el uso de cuatro categorías que implican propiedades de calidad, que son: correctitud, internas, contextuales y descriptivas."(Sánchez).

Dromey dentro de su modelo de calidad utiliza varias características dentro de las cuales podemos encontrar las siguientes:

- Eficiencia.
- Confiabilidad.
- Facilidad de mantener.
- Portabilidad.
- Facilidad de uso.
- Funcionalidad.

En el siguiente cuadro podemos observar una relación entre las categorías con cada una de sus características:

Factor	Criterio
Correctitud	Funcionalidad Confiabilidad
Internas	Mantenibilidad Eficiencia Confiabilidad
Contextuales	Mantenibilidad Reusabilidad Portabilidad Confiabilidad
Descriptivas	Mantenibilidad Reusabilidad Portabilidad Usabilidad

Tabla 01 Características o criterios asociados a los factores o categorías de calidad (Sánchez)

Dromey presenta cinco pasos relacionados con las categorías establecidas de acuerdo a sus características, detallados de la siguiente manera:

- “Especificación de los atributos de calidad de alto nivel (por ejemplo: confiabilidad, mantenibilidad).
- Determinación de los distintos componentes del producto a un apropiado nivel de detalle (por ejemplo: paquetes, subrutinas, declaraciones).
- Para cada componente, determinación y categorización de sus implicaciones más importantes de calidad.
- Proposición de enlaces que relacionan las propiedades implícitas a los atributos de calidad, o alternativamente, el uso de enlaces de las cuatro categorías de atributos propuestas.
- Iteración sobre los pasos anteriores, utilizando un proceso de evaluación y refinamiento.”(Camacho, Cardeso y Nuñez)

Dromey para ilustrar su hipótesis construye modelos de calidad de implementación, requerimientos y diseño.

1.1.3 Modelo de Boehm

Presentado en el año de 1978, se denomina también modelo espiral ya que realiza un conjunto de iteraciones, estas no son fijas debido a que el equipo de trabajo es quien las determina; es un modelo que por naturaleza utiliza jerarquías, así mismo por cada vuelta que ejecuta tiene etapas que se describen de la siguiente forma:

- Planeación: Se determinan los objetivos, alternativas y restricciones del proyecto.
- Análisis de riesgo: Se realiza el análisis de las alternativas, se identifican y se resuelven los riesgos encontrados.
- Ingeniería: Se procede al desarrollo del producto hasta "el siguiente nivel".
- Evaluación: El cliente realiza la valoración verificando los resultados obtenidos.

Regiones de Tareas del Modelo

Este modelo en espiral (Boehm), posee un número de actividades de marco de trabajo, denominadas también regiones de tareas.
El siguiente gráfico identifica dentro del modelo espiral cada una de sus regiones.

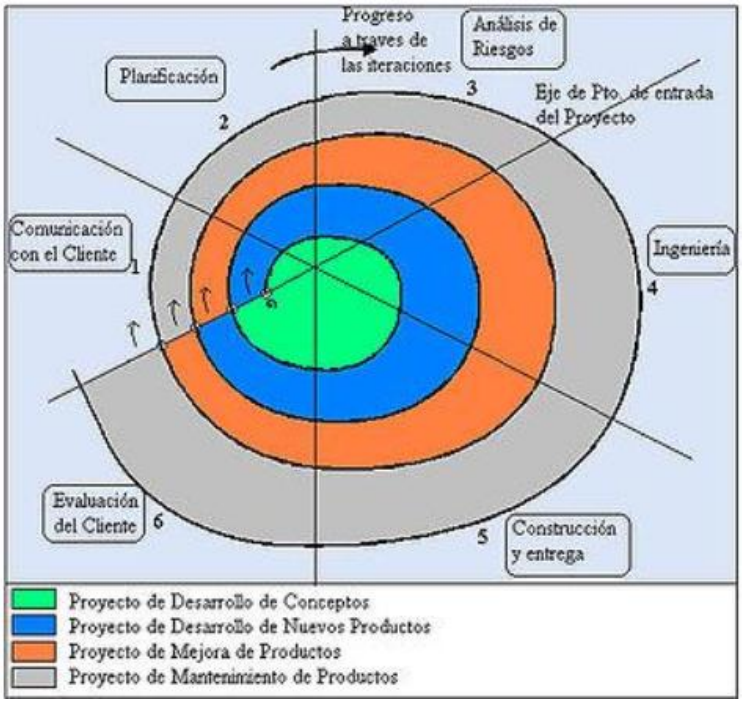

Gráfico 04 Regiones de tareas del modelo de Boehm

1. **Comunicación con el cliente**: Tareas requeridas para establecer la comunicación existente entre el desarrollador y el cliente.
2. **Planificación**: Se definen recursos, tiempo e información extra que se relacione con el proyecto.
3. **Análisis de riesgos**: Se realiza un conjunto de tareas para evaluar riesgos de gestión y técnicos.
4. **Ingeniería**: Tareas requeridas que permitan realizar una o varias representaciones de la aplicación.
5. **Construcción y acción**: Tareas que nos ayuden con la construcción, pruebas, instalación y soporte para el usuario (documentación).

6. **“Evaluación del cliente**: Las tareas requeridas para obtener la reacción del cliente según la evaluación de las representaciones del software creadas durante la etapa de ingeniería e implementada durante la etapa de instalación. Cada una de las regiones está compuesta por un conjunto de tareas del trabajo, que se adaptan a las características del proyecto que va a emprenderse. Para proyectos pequeños, el número de tareas de trabajo y su formalidad es bajo, para proyectos mayores y más críticos cada región contiene tareas de trabajo que se definen para lograr un nivel más alto de formalidad. En todos los casos, se aplican las actividades de protección.” (Jiménez)

1.1.4 Modelo de CMMI

Fue creado en 1985 por el SEI (Software Engineering Institute), este instituto elaboró el modelo CMMI luego de resultar ganador de un concurso que fue propuesto debido a los problemas que tenía para la elaboración de software el departamento de defensa de los Estados Unidos. Con estos antecedentes se convoca al concurso para que el ganador fuera quien solucione los problemas de software, dando como resultado el surgimiento de este modelo.

El modelo CMMI adopta varios conceptos del modelo CMM, de este modo podemos definirlo de la siguiente manera: “Es un modelo de calidad del software que clasifica las empresas en niveles de madurez. Estos niveles sirven para conocer la madurez de los procesos que se realizan para elaborar el software”.(Valencia y Velásquez)

El CMMI se emplea con dos fines:

- Guía para mejorar procesos relativos a la producción y mantenimiento del software.
- Criterio para determinar el nivel de madurez de una organización que produce y mantiene software.

El CMMI tiene cinco niveles los cuales serán tratados a continuación:

Inicial o Nivel 1 CMMI: En este nivel se encuentran todas aquellas empresas en las que los procesos no están presentes, esto tiene como consecuencia un gasto excesivo de dinero y por lo tanto los presupuestos se disparan, los proyectos no son entregados en las fechas ofrecidas ya que no se cuenta con un control sobre el estado del proyecto, el desarrollo del proyecto es incierto y no se sabe qué pasará con él.

Nivel 2 CMMI: En este nivel las cosas marchan de forma correcta es decir que el proyecto no tiene nada que ver con el nivel anterior ya que existe un control total sobre el desarrollo, los plazos se empiezan a cumplir y por lo tanto no hay un despilfarro de dinero.

Los procesos que hay que implantar para alcanzar este nivel son:

- Gestión de requisitos.
- Planificación de proyectos.
- Seguimiento y control de proyectos.
- Gestión de proveedores.
- Aseguramiento de la calidad.
- Gestión de la configuración.

Nivel 3 CMMI: Llegar a este nivel es hablar de que se ha encontrado la forma correcta para desarrollar proyectos, se ha

conseguido documentar, definir métricas, entre otros. Con este nivel podemos llegar a la creación de objetivos concretos.

Los procesos que hay que implantar para alcanzar este nivel son:

- Desarrollo de requisitos.
- Solución Técnica.
- Integración del producto.
- Verificación.
- Validación.
- Desarrollo y mejora de los procesos de la organización.
- Definición de los procesos de la organización.
- Planificación de la formación.
- Gestión de riesgos.
- Análisis y resolución de toma de decisiones.

Este nivel proporciona una gran cantidad de beneficios, es por eso que la mayoría de empresas toman como meta ya que todas sus necesidades han sido cumplidas.

Nivel 4 CMMI: Los proyectos usan objetivos medibles para alcanzar las necesidades de los clientes, usando métricas para gestionar la correcta organización del proyecto de software.

Los procesos que hay que implantar para alcanzar este nivel son:

- Gestión cuantitativa de proyectos.
- Mejora de los procesos de la organización.

Nivel 5 CMMI: Para perfeccionar cada una de las actividades de los proyectos se efectuarán mejoras incrementales e innovadoras de los procesos que mediante métricas son identificados, evaluados y puestos en práctica.

Los procesos que hay que implantar para alcanzar este nivel son:

- Innovación organizacional.
- Análisis y resolución de las causas.

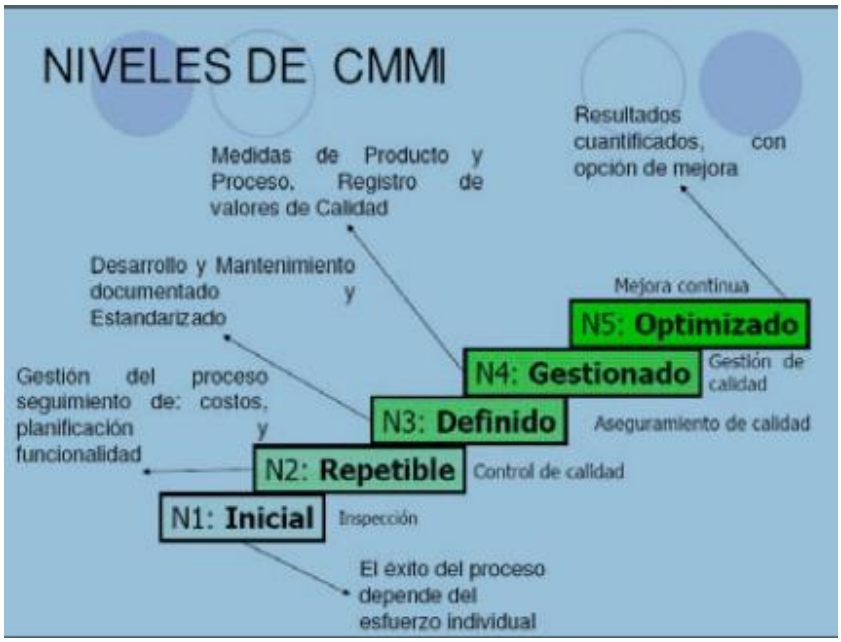

Gráfico 05 Niveles de CMMI.

1.1.5 Modelo ISO/IEC 9126.

El estándar ISO/IEC 9621 se establece en 1977, tomando como base el modelo de McCall, es un estándar internacional para la evaluación del software y consta de 4 partes:

- ISO/IEC 9126-1 evalúa la calidad de los productos software tomando dos subdivisiones:
  Calidad interna y externa:
  "La calidad interna, se define como la totalidad de las características de un software desde un enfoque interno; y la calidad externa, se define como la totalidad de las características del software desde un enfoque externo."(Melendez y Dávila)

Calidad en uso:

"Los atributos internos son indicadores de los atributos externos. Un atributo interno puede influir a una o más características, y una característica puede verse influida por uno o más atributos".(Tellez)

- La ISO/IEC 9621-2 categoriza la calidad de los atributos en software en seis características (funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad) las mismas que se subdividen en subcaracterísticas que pueden ser medidas con métricas internas o externas.

CALIDAD DE SOFTWARE INTERNA Y EXTERNA					
Funcionalidad	Fiabilidad	Usabilidad	Eficiencia	Mantenibilidad	Portabilidad
Adecuación	Madurez	Fácil comprensión	Comportamiento frente al tiempo	Facilidad de análisis	Adaptabilidad
Exactitud	Tolerancia a fallos	Fácil aprendizaje	Uso de recursos	Capacidad para cambios	Facilidad de instalación
Interoperabilidad	Capacidad de recuperación	Operatividad	Adherencia a normas	Estabilidad	Coexistencia
Seguridad	Adherencia normas	Software atractivo		Facilidad para pruebas	Facilidad de reemplazo
Adherencia a Normas		Adherencia a normas		Adherencia a normas	Adherencia a normas

Tabla 02 Características Internas y Externas del Modelo ISO 9621

Funcionalidad: son atributos que corresponden a un conjunto de funciones y sus propiedades específicas. Se compone de varias subcaracterísticas:

- Adecuación: Capacidad del software de proporcionar un conjunto apropiado de funciones para tareas específicas y objetivos del usuario.
- Exactitud: "Capacidad del software para proporcionar resultados correctos o que necesitan un determinado grado de precisión". (Tellez)

- Interoperabilidad: Capacidad del software para interactuar con uno o más sistemas especificados.
- Seguridad: Capacidad del software de proteger la información y los datos.
- Adherencia a normas: Capacidad del software relacionada con el grado de conformidad con estándares, convenciones o regulaciones existentes.

Fiabilidad: atributos que corresponden a la capacidad del software para mantener su nivel de prestación bajo condiciones establecidas durante un tiempo establecido.

- Madurez: Capacidad para evitar fallos que son el resultado del funcionamiento del software.
- Tolerancia a fallos: Capacidad para mantener un nivel establecido de rendimiento en caso de fallos durante el funcionamiento.
- Capacidad de recuperación: Restablecer su funcionamiento y recuperar la información afectada en caso de presentarse un fallo.
- Adherencia a normas: se relaciona con el grado de adaptación del software a estándares convenciones o regulaciones.

Facilidad de Uso: Capacidad del software para ser entendido, aprendido y usado.

- Fácil comprensión: permite al usuario dar a conocer si el producto es aceptable en determinadas formas de uso.
- Fácil aprendizaje: capacidad del software que permite al usuario aprender sus características.
- Operatividad: Capacidad del software que permite al usuario controlar y usar la aplicación.
- Software atractivo.

- Adherencia a normas: se relaciona con el grado de adaptación del software a estándares convenciones o regulaciones.

Eficiencia: Se define como la capacidad del software para proporcionar un rendimiento apropiado relacionado con los recursos que se utilizan bajo condiciones preestablecidas.

- Comportamiento frente al tiempo: Capacidad de respuesta del software dentro de un rango de tiempo de procesamiento apropiado.
- Uso de recursos: Desarrollar sus funciones dentro de un tiempo de procesamiento establecido con la utilización apropiada de recursos.
- Adherencia a normas: se relaciona con el grado de adaptación del software a estándares convenciones o regulaciones.

Mantenibilidad: Capacidad del software para ser modificado.

- Facilidad de análisis: Capacidad del software para diagnosticar deficiencias o causas de fallos.
- Capacidad para cambios: Capacidad del software que permite una modificación específica.
- Estabilidad: Capacidad del software para evitar defectos no esperados debido a modificaciones en el mismo.
- Facilidades para pruebas: Capacidad del software que permite al producto modificado ser evaluado.
- Adherencia a normas: Se relaciona con el grado de adaptación del software a estándares convenciones o regulaciones.

Portabilidad: “Capacidad del software que tiene para ser transferido de un entorno a otro”(Ruiz). Ya sea hardware o software.

- Adaptabilidad: Capacidad del software para ser adaptado a diferentes entornos especificados sin aplicar acciones alejadas de aquellas que el propio software proporcione.
- Facilidad de instalación: Capacidad del software para ser instalado en un entorno específico.
- Coexistencia: Capacidad del software de coexistir con otros programas independientes, compartiendo recursos.
- Facilidad de reemplazo: Capacidad del producto software de ser utilizado en lugar de otro con el mismo propósito.
- Adherencia a normas: se relaciona con el grado de adaptación del software a estándares convenciones o regulaciones.

La calidad de uso se define como: “La capacidad del software que posibilita la obtención de objetivos específicos con efectividad, productividad, satisfacción y seguridad”. (Ruiz)
Consta de las siguientes características:

- Eficiencia: Capacidad del software para permitir a los usuarios alcanzar objetivos específicos con precisión y dentro de un contexto específico de uso.
- Productividad: Capacidad del software para permitir a los usuarios emplear los recursos apropiados con relación a la eficiencia alcanzada en un contexto específico.
- Seguridad: La capacidad del producto software para alcanzar niveles aceptables de riesgo hacia la gente,

negocio, software, propiedad o medio ambiente, en un contexto específico de uso.

- Satisfacción: La capacidad del producto software para satisfacer al usuario dentro de un contexto específico de uso.

- ISO/IEC 9126-3 Métricas Externas.

Esta norma es un reporte técnico que contiene terminología relacionada con las medidas de las métricas, "este reporte proporciona al usuario una guía para la evaluación de planificación, selección, diseño, aplicación de métricas e interpretación de medidas de datos". (Tellez)Las medidas se pueden interpretar de tres maneras:

- "Medida directa: Es una medida de un atributo que no depende de las medidas de otros atributos.
- Medida indirecta: Es derivada de medidas de uno o más atributos.
- Indicadores: Son aquellas medidas que pueden ser estimadas o predichas desde otras." (Tellez)

Las métricas tienen unas propiedades deseables que se listan a continuación:

- Fiabilidad.
- Indicabilidad.
- Disponibilidad.
- Corrección.
- Imparcialidad.

El conjunto de métricas que contiene están organizadas por características y subcaracterísticas, donde cada una contiene:

- Nombre.
- Propósito.
- Método de aplicación.
- Medida, fórmula y cómputo de datos.

- Interpretación del valor medido.
- Tipo de escala.
- Tipo de medida.
- Fuente de medida.
- Referencia a ISO/IEC 12207 SLCP.
- Audiencia.

- ISO/IEC 9126-4 Métricas Internas.

  "Proporciona métricas internas para medir los atributos de las características definidas"(Ruiz)en 9126-1.

  Con las siguientes cualidades:

  - "Se aplican a un producto de software no ejecutable.
  - Se aplican durante las etapas de desarrollo.
  - Permiten medir la calidad de los entregables intermedios.
  - Permiten predecir la calidad del producto final.
  - Permiten al usuario iniciar acciones correctivas temprano en el ciclo de desarrollo."(Ruiz).

  ISO/IEC 9126-4 proporciona las métricas de calidad en uso para medir los atributos definidos en ISO/IEC 9126-1.

1.1.6 Modelo de FURPS

Tomando como base el modelo de McCall, Heweltt-Packard lo desarrolla en 1987. Los factores de calidad que lo componen son los siguientes: funcionalidad (Functionality), usabilidad (Usability), confiabilidad (Reliability), desempeño (Performance) y capacidad de soporte (Supportability). Formando un acrónimo de donde proviene su nombre. Se divide en dos grupos:

Requerimientos Funcionales (F): Especifican que el software debe ser capaz de realizarse sin tomar en cuenta restricciones físicas, y se definen a través de las entradas y salidas esperadas.

Funcionalidad: Los requisitos de funcionalidad deben incluir

1. Conjunto de Características,
2. Capacidades
3. Seguridad.

Requerimientos no funcionales (URPS): Describen atributos del sistema o del ambiente.

Usabilidad: Deben incluir subcategorías tales como:

1. Factores humanos.
2. Estéticos
3. Consistencia en la Interfaz de Usuario.
4. Ayuda en línea.
5. Asistentes.
6. Documentación del usuario.
7. Material de capacitación.

Confiabilidad: Se considera requisitos de confiabilidad:

1. Frecuencia y severidad de fallas.
2. Recuperación a fallos.
3. Tiempo entre fallos.

Desempeño (Rendimiento): Un requisito de rendimiento impone condiciones a los requisitos funcionales. Por ejemplo: a una acción dada, se pueden especificar los siguientes parámetros de rendimiento:

1. Velocidad.
2. Eficiencia.
3. Disponibilidad.
4. Tiempo de Respuesta.
5. Tiempo de Recuperación.
6. Utilización de Recursos.

Soporte: Los requisitos de soporte pueden incluir:

1. Requisitos de instalación.
2. Requisitos de Configuración.
3. Requisitos de Adaptabilidad.
4. Requisitos de Compatibilidad.

El desarrollo incluye además restricciones de diseño y requerimientos de implementación, físicos y de interfaz. No toma en cuenta la portabilidad de los productos. Los cuales conforman el FURPS +

Tipo de requerimiento		**Descripción**
F	Funcionalidad	Características, capacidades y algunos aspectos de seguridad.
U	Usabilidad	Factores Humanos (interacción), ayuda, documentación.
R	Confiabilidad	Frecuencia de fallos, capacidad de recuperación de un fallo y grado de previsión.
P	Desempeño	Tiempos de respuesta, productividad, precisión, uso de recursos.
S	Soporte	Adaptabilidad, facilidad de mantenimiento, internacionalización, facilidad de configuración.
+	Implementación	Limitación de recursos, lenguajes y herramientas, hardware.
	Interfaz	Restricciones impuestas para la interacción con sistemas externos.
	Operaciones	Gestión del sistema, pautas administrativas, puesta en marcha.
	Empaquetamiento	Forma de distribución.
	Legales	Licencia, derechos de autor, entre otros.

Tabla 03 Tipos de Requerimientos. (Anónimo, FURPS)

1.1.7 Modelo de Gilb

Definido en 1988 por Gilb, presenta como aspecto fundamental la definición de los atributos y el nivel de calidad que debe tener cada uno de ellos para satisfacer al usuario, pues no tiene sentido exigir

calidad en un producto, si no se cuenta con esta base. La especificación de requisitos de calidad la realiza conjuntamente el usuario con el analista. Estos pueden ser de dos tipos: originales y de métodos tradicionales.

Las características se pueden medir mediante subcaracterísticas o métricas detalladas. Para cada una de ellas debe definirse los siguientes conceptos:

- Nombre y definición de la característica.
- Escala o unidades de medición.
- Recopilación de datos o prueba.
- Valor previsto.
- Valor óptimo.
- Valor en el sistema actual.
- Comentarios.

Consta de las siguientes características:

- Corrección: grado en el que el software lleva a cabo una función requerida. Si un programa no opera correctamente no dará valor agregado a sus usuarios.
- Facilidad de mantenimiento: Posibilidad de corregir un programa si se encuentra un error, adaptarlo si cambia su entorno, o mejorarlo si el cliente desea un cambio.
- Integridad: habilidad del sistema para resistir ataques. Tanto accidentales como intencionados, contra su seguridad, en todos sus niveles. Para medir la integridad se sugiere otros dos atributos como son:
    - Amenaza: es la probabilidad de que un ataque de cualquier tipo ocurra.
    - Seguridad: es la probabilidad de que se pueda repeler un determinado ataque.
    - Facilidad de uso: es un intento por cuantificar lo amigable que puede ser el producto con el usuario.

1.2 Definición de métricas.

El estándar de la IEEE define una métrica como: "Una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado" (Desarrolloweb.com)

Son también un medio para entender, monitorizar, controlar, predecir y probar el desarrollo de software y los proyectos de mantenimiento. Las mediciones de software persiguen tres objetivos principales:

1. Dar a entender que está sucediendo durante el desarrollo y el mantenimiento del proyecto.
2. Permiten controlar lo que ocurre en los proyectos.
3. Permiten mejorar los proyectos y sus productos.

Para medir el software se asigna un amplio rango de actividades diversas las cuales son establecidas por métricas. Por ejemplo:

- Medidas y modelos de estimación de coste y esfuerzo.
- Modelos y medidas de productividad.
- Aseguramiento y control de calidad.
- Recolección de datos.
- Medidas y modelos de calidad.
- Modelos de fiabilidad.
- Modelos de evaluación de ejecución.
- Complejidad computacional o algorítmica.

El software tiene tres tipos de entidades, sus atributos pueden intentar medir:

- Procesos: Es un conjunto estructurado de actividades requeridas para desarrollar un sistema de software de alta calidad.

  Actividades

  - Especificación.
  - Diseño.
  - Validación.
  - Evolución.
- Productos: Son entregables, artefactos o documentos generados en el ciclo de vida del software. Ejemplos de atributos externos: la fiabilidad del código, la comprensión de un documento de

especificación, la mantenibilidad del código fuente. Ejemplos de atributos internos: la longitud, funcionalidad, modularidad, o corrección de los documentos de especificación.

- Recursos: Es una fuente o suministro del cual se produce un beneficio. Algunos recursos relacionados a la elaboración del software son:
    - El personal.
    - Materiales.
    - Herramientas.
    - Métodos.

De esta manera tenemos la siguiente clasificación de métricas que se pueden aplicar:

Métricas de Tamaño: El tamaño se vuelve una característica común a pesar de la diversidad de lenguajes de programación existentes. Para su determinación se puede tomar como referencia el código fuente. Así también el espacio de almacenamiento, el tamaño se emplea fundamentalmente por tres razones: fácil de obtener al finalizar el programa, es uno de los factores más importantes en los métodos de desarrollo y la productividad se expresa tradicionalmente con el tamaño del código.

De esta manera la medida más utilizada para establecer el tamaño son las líneas de código.(Viramontes Aguilar, Perez Becerra y Gonzalez de la Cruz)

Métricas de estructuras de control: La estructura lógica de un programa es el mecanismo que le permite realizar las distintas funciones para las que fue construido. Se representa por los algoritmos lógicos empleados en su diseño y procesa los datos. Los algoritmos se representan mediante los diagramas de flujo. El flujo de control de un programa por lo general es secuencial aunque en ocasiones puede ser interrumpido:

- En una decisión, se divide en dos nuevas líneas de flujo que responden a la evaluación de una condición determinada.
- Un salto hacia atrás devuelve el flujo de control a una instrucción que ya ha sido ejecutada.

- Una transferencia de control a una rutina o procedimiento externo, hace que el flujo discurra por un camino externo al programa.

De aquí obtenemos las siguientes métricas:

- Cuenta de decisión: cuenta el número de instrucciones de decisión y bucles condicionales. Debe tomarse en cuenta si las condiciones son compuestas, es decir los operadores AND y OR incrementan el valor de una cuenta de decisión.
- Número de complejidad ciclomática: Se refiere a valorar el número de caminos linealmente independientes de un programa, lo que está relacionado con la facilidad para probar y mantener el código.
- El anidamiento: Afecta a lenguajes de alto nivel. Cada instrucción tiene un nivel de anidamiento determinado.

Métricas compuestas: Son basadas en medidas más sencillas, como son medidas establecidas en métricas: de estructuras de control, de esfuerzo, de diseño, etc.

- Longitud estimada del programa: depende únicamente del número de operadores y operandos distintos.
- Volumen del programa: se interpreta como el número de comparaciones mentales necesarias para escribir un programa de cierta longitud.

Métricas de esfuerzo: A la hora de valorar un sistema software debe considerarse la cantidad de esfuerzo que debe invertir el equipo de desarrollo para culminar su construcción.

- Persona-Mes: El esfuerzo requerido para construir un sistema puede ser medido con muchas unidades. Desde las discriminaciones mentales hasta el número real de horas y minutos que invierte un programador. En proyectos pequeños puede tomarse por Persona–Día.

- Tiempo de comprensión y aprendizaje que se requiere para completar los requerimientos, el diseño o cualquier documento previo a la codificación. Aprender a manejar herramientas y lenguajes.
- Interrupciones: pueden presentarse de varios tipos son numerosas y habituales, no son un factor de ruptura importante.

Métricas de diseño: Cada vez se vuelve más complejo un programa debido a que crece en tamaño, por lo tanto son más difíciles de depurar, escribir y comprender. Para reducir esta complejidad, "los diseñadores de software han hecho un uso progresivo de técnicas de modularización y diseño estructurado"(Anónimo, Organización y Estructura de la Información). Las ventajas de las técnicas son las siguientes:

- "Comprensibilidad: Programadores y usuarios pueden comprender fácilmente la lógica del programa.
- Manejabilidad: Los gestores pueden asignar fácilmente personal y recursos a los distintos módulos representados por tareas.
- Eficiencia: El esfuerzo de implementación puede reducirse.
- Reducción de errores: Los planes de prueba se simplifican notablemente.
- Reducción del esfuerzo de mantenimiento: la división en módulos favorece que las distintas funciones las lleven a cabo módulos diferenciados."(Anónimo, Organización y Estructura de la Información).

De acuerdo a estas ventajas las métricas se definen por los siguientes principios:

- Acoplamiento: Mide el número de interconexiones entre módulos.
- Cohesión: Valora las relaciones entre los elementos de un módulo, en un diseño cohesivo las funciones están ubicadas en un solo módulo.

- Complejidad: Crece con el número de construcciones de control, y el número de módulos de un programa.
- Modularidad: Afecta a la calidad de un diseño, es preferible un exceso, a un defecto de modularidad, pues los defectos tendrán más errores.
- Tamaño: Un programa con grandes módulos tendrá más errores, "La complejidad y tamaño están muy relacionados, las consecuencias de un exceso en cualquiera de los dos tendrán el mismo resultado."(Anónimo, Organización y Estructura de la Información).

1.2.1 Métricas Seleccionadas

Métrica	Funcionalidad.
Interviene	Adecuidad, Exactitud.
Fórmula	X=1-A/B
Interpretación	0<=X<=1 Entre más cercano a 1, más completo.
Fuente de Medición	Especificación de requisitos. Diseño.
Variables	A= número de funciones que faltan. B= Número de funciones que están en la especificación.

Tabla 04 Métrica de Funcionalidad

Métrica	Fiabilidad.
Interviene	Tolerancia a Fallos, Capacidad de recuperación.
Fórmula	X=A+B
Interpretación	Entre X sea menor, mayor la fiabilidad
Fuente de Medición	Fallos del sistema.
Variables	A Tiempo Medio de fallo. B Tiempo medio de repetición. X=Tiempo Medio entre fallos.

Tabla 05 Métrica de Fiabilidad

Métrica	Eficiencia.
Interviene	Comportamiento temporal, Utilización de recursos.
Fórmula	X=Tiempo calculado o simulado
Interpretación	Entre más corto mejor.
Fuente de Medición	Tiempo de respuesta.
Variables	Tiempo.

Tabla 06 Métrica de Eficiencia

Métrica	Integridad.
Interviene	Amenaza y Seguridad
Fórmula	X= 1-(A/B)
Interpretación	Más cercano a 1 es mejor.
Fuente de Medición	Número de Entradas y Número de fallos.
Variables	A = Entradas. B = fallos.

Tabla 07 Métrica de Integridad

Métrica	Usabilidad.
Interviene	Número de funciones evidentes al usuario y funciones el número total de funciones.
Fórmula	X=A/B
Interpretación	0<= X <= 1. Más cercano a 1 es mejor.
Fuente de Medición	Especificación de requisitos.
Variables	A =Número de funciones evidentes al usuario. B = Total de funciones.

Tabla 08 Métrica de Usabilidad

Métrica	Mantenibilidad.
Interviene	Estabilidad, Capacidad para ser probado, Capacidad para cambios, Facilidad de análisis.
Fórmula	X=A/B
Interpretación	0<=X<=1 Más cercano a 1 más registrable. Y más cercano a 0, más estable.
Fuente de Medición	Corregir un error, Realizar un cambio.
Variables	A= Número de cambios a funciones o módulos que tienen comentarios confirmados. B= Total de funciones o módulos modificados.

Tabla 09 Métrica de Mantenibilidad

Métrica	Portabilidad.
Interviene	Instalabilidad, Adaptabilidad, Facilidad de remplazo.
Fórmula	X=A/B
Interpretación	0<=X<=1 Entre X cercano a 1, más completo.
Fuente de Medición	Diseño. Especificación de conformidad y estándares.
Variables	A= Número de artículos implementados de conformidad. B= Total de artículos que requieren conformidad.

Tabla 10 Métrica de Portabilidad

1.3 Ventajas y desventajas de los modelos de calidad.

1.3.1 Ventajas de los Modelos de Calidad

- Permite encontrar, evaluar y resolver errores que se están presentando en un producto (software).
- Detecta errores al principio del proyecto ya que si estos son encontrados posteriormente, pueden tener un efecto catastrófico.
- Evalúa qué es lo que realmente necesita nuestro producto, determinando así la inversión exacta que se debe hacer para la conclusión de nuestro proyecto.

- Impone la necesidad de mucha disciplina, planificación y administración, en el proceso de desarrollo de software, venciendo así la filosofía de los procesos de codificar y probar.
- Permite una estrecha relación entre usuarios y desarrolladores, obteniendo así claramente los objetivos del proyecto.

1.3.2 Desventajas de los Modelos de Calidad

- Cuando el mejoramiento se concentra en un área específica de la organización, se pierde la perspectiva de la interdependencia que existe entre todos los miembros de la empresa.
- Requiere de un cambio en toda la organización, ya que para obtener el éxito es necesario la participación de todos los integrantes de la organización y a todo nivel.
- En vista de que los gerentes en la pequeña y mediana empresa son muy conservadores, el mejoramiento continuo se hace un proceso muy largo.
- Hay que hacer inversiones importantes.

Modelos	McCall	Dromey	Boehm	CMMI	ISO 9126	FURPS	GILB
Ventajas							
Existe una relación directa entre los desarrolladores y el usuario.	**X**						**X**
Evalúa el producto a nivel bajo.	**X**						
Evalúa un producto de forma independiente.		**X**					
Existe una relación directa entre los atributos y sub-atributos.		**X**					**X**
Utiliza niveles jerárquicos.	**X**	**X**	**X**	**X**			
Clasifica a las empresas en niveles según su madurez.				**X**			
Permite guiar paso para mejorar a través de niveles o etapas.				**X**			
Evalúa el producto mediante retroalimentación.					**X**		
Evalúa el software sin tomar en cuenta las restricciones físicas.						**X**	
Desventajas							
Es muy costoso.				**X**			
Es excesivamente detallado.				**X**			
Presta atención a la gestión dejando de lado aspectos técnicos.				**X**	**X**		
Difícil entender.				**X**			

Tabla 11 Ventajas y Desventajas de los Modelos de Calidad

1.3.3 Cuadro Comparativo de las Características de los Modelos de Calidad

	Modelos						
Características de Calidad	McCall	Dromey	Boehm	CMMI	ISO 9126	FURPS	GILB
Facilidad de Uso	**X**	**X**			**X**	**X**	**X**
Integridad	**X**			**X**			**X**
Corrección	**X**						**X**
Confiabilidad	**X**	**X**	**X**		**X**	**X**	
Eficiencia	**X**	**X**	**X**	**X**	**X**		
Facilidad de mantenimiento	**X**	**X**			**X**		**X**
Facilidad de prueba	**X**		**X**				
Flexibilidad	**X**						
Facilidad de reutilización	**X**						
Interoperabilidad	**X**			**X**			
Portabilidad	**X**	**X**	**X**		**X**		
Fácil de entender			**X**				
Fácil de modificar			**X**				
Funcionalidad		**X**			**X**	**X**	
Ambigüedad				**X**			
Documentación				**X**			

Tabla 12 Cuadro Comparativo de las Características de los Modelos de Calidad

1.4 Elección del Modelo de Calidad a ser Aplicado

Después de realizar un análisis exhaustivo en el que se ha podido determinar en primer lugar las cualidades de cada uno de los modelos de calidad, así como también obtener cada una de sus ventajas y desventajas, características, entre otros; concluimos que el modelo de calidad que aplicaremos para el tratamiento del siguiente capítulo es el Modelo McCall.

1.5 Conclusiones

En la investigación del presente capítulo se ha podido obtener como conclusión que existen varios modelos de calidad, pero que dentro de ellos son muy importantes la orientación que tienen, es decir que algunos son basados en la eficiencia, otros en la portabilidad, facilidad de mantener, entre otros. Fue de gran utilidad el realizar cuadros de características para poder saber cuál de los modelos investigados es el más adecuado para desarrollar la aplicación en los capítulos posteriores; en segundo lugar el análisis contribuyó para poder obtener ventajas y desventajas de cada modelo y de esta manera saber cuál el más conveniente en el futuro de esta investigación.

Así mismo se definieron las métricas mediante las cuales se podrá medir cada uno de los parámetros de calidad de los productos de software a ser tratados.

CAPÍTULO II

2. Evaluación de Frameworks para el análisis, diseño y desarrollo de aplicaciones móviles en la plataforma Android.

2.1. Definición de los frameworks.

2.1.1. Basic4Android.

Es un entorno de desarrollo simple pero poderoso que se dirige a los dispositivos Android. El idioma Basic4android es similar al lenguaje Visual Basic con el apoyo adicional de los objetos. Las aplicaciones compiladas Basic4android son aplicaciones Android nativas, no hay tiempos de ejecución o dependencias adicionales. Es compatible con Android 1.6 y superiores, incluyendo tablets Android 3.x. En un IDE y lenguaje de programación 100% enfocada en el desarrollo de Android. Compila a código de bytes nativo. No se necesitan bibliotecas en tiempo de ejecución. Los archivos APK son creados exactamente como los archivos APK creados con Java / Eclipse. El rendimiento es similar a las aplicaciones escritas en Java. Los eventos de lenguaje de programación son similares a Visual Basic con soporte para objetos y módulos de código.

Soporta todos los core Android, entre ellos los: bases de datos SQL, GPS, widgets de pantalla de Inicio, Servicios en segundo plano y oyentes de difusión; Bluetooth, USB, servicios Web, cámara, XML, JSON y CSV, vistas animadas, gestos multitouch, Networking (TCP, UDP, FTP, SMTP y POP3), notificaciones Push (C2DM), texto a voz y reconocimiento de voz, gráficos y diagramas; sensores, integrar con servicios como: Dropbox, Google Analytics, Twitter y otros, objetos de serialización.(Basic4android - Free download and software reviews - CNET Download.com)

2.1.2. Eclipse.

"La plataforma Eclipse consiste en un Entorno de Desarrollo Integrado (IDE, Integrated Development Environment) abierto y extensible. Un IDE es un programa compuesto por un conjunto de herramientas útiles para un desarrollador de software. Como elementos básicos, un IDE cuenta con en un editor de código, un compilador/intérprete y un depurador. Eclipse sirve como IDE Java y cuenta con numerosas herramientas de desarrollo de software. También da soporte a otros lenguajes de programación, como son C/C++, Cobol, Fortran, PHP o Python. A la plataforma base de Eclipse se le pueden añadir extensiones (plugins) para extender la funcionalidad.

El término Eclipse además identifica a la comunidad de software libre para el desarrollo de la plataforma Eclipse. Este trabajo se divide en proyectos que tienen el objetivo de proporcionar una plataforma robusta, escalable y de calidad para el desarrollo de software con el IDE Eclipse. Este trabajo está coordinado por la Fundación Eclipse, que es una organización sin ánimo de lucro, creada la promoción y evolución de la plataforma Eclipse dando soporte tanto a la comunidad como al ecosistema Eclipse."(Eclipse)

2.1.3. Adobe Flash Builder.

Está diseñado para ayudar a los desarrolladores de software a desarrollar rápidamente contenido y RIA (Aplicaciones Enriquecidas de Internet) multiplataforma utilizando el marco Flex de código abierto. Incluye compatibilidad con la codificación inteligente, la depuración, diseño visual y proporciona poderosas herramientas de prueba que aceleran el desarrollo y conducen a aplicaciones de mayor rendimiento.

Acelera el desarrollo de aplicaciones de Flex (tecnología que da soporte al desarrollo de las aplicaciones RIA). Es una herramienta de desarrollo basada en Eclipse que permite la codificación inteligente,

la depuración interactiva y el diseño visual de la disposición de la interfaz de usuario, la apariencia y el comportamiento de las RIAs. Esta aplicación también integra el marco de trabajo completo de Flex, así como su biblioteca de componentes, compiladores, o depuradores. (Adobe Flash Builder descarga gratis)

2.2. Aplicación del modelo de calidad para cada framework.

De acuerdo con el análisis desarrollado en el capítulo anterior, se procederá a aplicar el modelo calidad de McCall a cada una de las herramientas expuestas anteriormente: Basic4Android, Eclipse y Flash Buider. Obteniendo como resultado la herramienta más óptima para el desarrollo de una aplicación Android.

2.3. Aplicación de métricas para cada framework.

2.3.1. Funcionalidad.

Funciones necesarias para desarrollar en android:

Editor de Código fuente.

- Compilador.
- Diseñador de pantallas.
- Depurador de aplicaciones.
- Conector con Bases de datos.

Funciones	**Basic4Android**	**Eclipse**	**Flash Builder**
Editor de Código fuente.	X	X	X
Diseñador de pantallas.			X
Depurador de aplicaciones.	X	X	X
Conector con Bases de datos.	X	X	X
Fórmula: X =A/B	**0,75**	**0,75**	**1**

Tabla 13 Análisis de funcionalidad

2.3.2. Fiabilidad.

Para medir la fiabilidad se establece un período tiempo de pruebas, para determinar cuántas veces falla el sistema.

Medido en un período de 8 horas, los fallos encontrados fueron:

- El sistema no responde.
- La aplicación se cierra durante la prueba.
- Depuración fallida.

Fallos	**Basic4Android**	**Eclipse**	**Flash Builder**
El sistema no responde.	3	1	0
La aplicación se cierra durante la prueba	0	0	0
Depuración fallida.	2	2	1
Total Fallos	**5**	**3**	**1**
Fórmula=1-(A/B)	**0**	**0,4**	**0,8**

Tabla 14 Análisis de fiabilidad

2.3.3. Eficiencia.

Para su determinación, se toma en cuenta el tiempo empleado en algún proceso y los recursos utilizados. Entre los cuales podemos citar:

- Memoria.
- Tiempo de instalación.
- Configuración.
- Tiempo de arranque.
- Arranque depurador.
- Tamaño instalador.

	Basic4Android	Eclipse	Flash Builder	Unidad
Memoria	26144	353234	474192	KB
Tiempo de instalación	0,5	0 (Ejecutable directo)	6	Minutos
Configuración	60 (Instalación Android SDK)	60 (Instalación Android SDK)	4 (AndroidDeveloperTool)	Minutos
Tiempo de arranque	3	32	25	Segundos
Arranque depurador	4	15	12	Segundos
Tamaño instalador	4	300	1100	MB
Fórmula=A/B	**0,67**	**0,17**	**0,17**	

Tabla 15 Análisis de eficiencia

2.3.4. Usabilidad.

Se establece el número de funciones evidentes al usuario y el total de funciones que se presentan, para este caso se ha tomado el mayor número de funciones encontradas.

FUNCIONES	Basic4Android	Eclipse	Flash Builder
Resaltado de sintaxis	X	X	X
Los números de línea	X	X	X
Compilar y ejecutar operaciones	X	X	X
Vista de tareas		X	X
Depurador		X	X
Vista de recursos	X		X
Importación / exportación de proyectos		X	X
Distingue archivos .class y .java		X	X
Configuración del depurador para android			X
Filtros de depuración			X
Construcción automática.		X	
Navegador HTML entre clases		X	X
Fórmula: X = A/B	**0,33**	**0,67**	**0,92**

Tabla 16 Análisis de usabilidad

2.3.5. Interoperabilidad.

Para la interoperabilidad se tiene en cuenta los diferentes sistemas operativos como son: los de Microsoft Windows, basados en Linux y OS X de Apple.

Así mismo se considera las plataformas para las que es posible desarrollar con cada uno de los frameworks. Para el caso se ha tomado en cuenta, iOS y Android.

Compatibilidad	Basic4Android	Eclipse	Flash Builder
Sistema operativo Windows	X	X	X
Sistema operativo Linux		X	X
Sistema operativo OS X		X	X
Desarrollo para Android	X	X	X
Desarrollo para iOS			X
Fórmula: X=A/B	0,4	0,8	1

Tabla 17 Análisis de interoperabilidad

2.3.6. Mantenibilidad.

Para el análisis se ha tomado en cuenta las operaciones que se puede tener sobre cada una de las aplicaciones, así como el mantenimiento del framework.

	Basic4Android	Eclipse	Flash Builder
Actualización de complementos		X	X
Capacidad de probar la aplicación	X	X	X
Cambio de resolución de pantalla	X	X	X
Capacidad de modificar aplicaciones	X	X	X
Capacidad de modificar proyectos	X	X	X
Fórmula: X = A/B	**0,8**	**1**	**1**

Tabla 18 Análisis de mantenibilidad

2.3.7. Portabilidad.

Para analizar este punto se establece las siguientes medidas:

- Compatibilidad con Windows
- Compatibilidad con Linux
- Compatibilidad con OS X
- Migración a otro sistema operativo
- Productos compatibles con otros sistemas

Compatibilidad	Basic4Android	Eclipse	Flash Builder
Sistema operativo Windows	X	X	X
Sistema operativo Linux		X	X
Sistema operativo OS X		X	X
Migración a otro sistema operativo		X	
Productos compatibles con otros sistemas		X	X
Formula: X=A/B	0,2	1	0,8

Tabla 19 Análisis de portabilidad

2.4. Cuadro comparativo de los diferentes frameworks.

Resultados de las métricas, representados en porcentajes.

Métrica	Basic4Android	Eclipse	Flash Builder
Funcionalidad	75	75	100
Fiabilidad	0	40	80
Eficiencia	67	17	17
Usabilidad	50	67	92
Interoperabilidad	40	80	100
Mantenibilidad	77	83	100
Portabilidad	20	100	80

Tabla 20 Cuadro comparativo del análisis de métricas aplicadas a: Basic4Andorid, Eclipse y Flash Builder.

2.5. Elección del framework a ser aplicado.

En la medición se han tomado en cuenta varios criterios para establecer la calidad de los frameworks y poder seleccionar el más óptimo.

De acuerdo con los valores obtenidos en la medición se procede a realizar el siguiente análisis:

Funcionalidad. Los resultados fueron:

- **Basic4Android.** Un 75% debido a que no posee un diseñador propio de pantallas.
- **Eclipse.** Un 75% debido a que no posee un diseñador propio de pantallas.
- **Flash Builder.** Obteniendo el 100% cumpliendo con los requerimientos básicos de un IDE para el desarrollo de aplicaciones Android.

Fiabilidad.

- **Basic4Android.** Se considera su número total de fallos como referencia, debido a que fue el framework en el que más fracasos ocurrió durante el período de prueba.
- **Eclipse.** Fallando en depuración por dos ocasiones, y con otro error presentado como un colapso del framework, se obtiene un 40% de fiabilidad.
- **Flash Builder.** Con un error en depuración durante el período de pruebas se establece como el framework más fiable de los tres, con un 80% de fiabilidad.

Eficiencia.

- **Basic4Android.** En cuanto a la eficiencia obteniendo un 67% ya que es el que menos recursos consume y sus tamaños físicos son menores, se presenta como el más eficiente de los tres fremeworks. Sin embargo, necesita del SDK de android para poder realizar sus compilaciones y el Android Device Manager como emulador.
- **Eclipse.** Con un 17 % de eficiencia y necesitado del SDK de android y presentando problemas al momento de integrar el plugin Android Developer Tool. Se establece que no es un

framework eficiente para el desarrollo de aplicaciones además necesita del Android Device Manager como emulador.

- **Flash Builder.** Siendo el más pesado de los tres por el consumo de recursos y tamaños físicos de 1100 MB, obtiene un 17% de eficiencia, cabe recalcar que integrado el plugin Android Developer Tool, no es necesario un emulador ya que cuenta con uno propio, además también es compatible con Android Device Manager en caso de que se lo requiera.

Usabilidad.

- **Basic4Android.** Tomando en cuenta el número máximo de funciones evidentes para el usuario en este framework, se obtiene el 33% de usabilidad.
- **Eclipse.** Se obtiene 67% de usabilidad.
- **Flash Builder.** Llegando a un 92% de usabilidad, se establece como el mejor de los tres en cuanto a este criterio.

Interoperabilidad.

- **Basic4Android.** Con un 40% obtenido en el análisis, no tiene acogida en otros sistemas operativos a más de Windows, así como únicamente permite desarrollar aplicaciones para la plataforma andorid.
- **Eclipse.** Se puede integrar con los sistemas operativos tanto Windows, Linux y OS X, lamentablemente no presenta la opción de desarrollo para iOS. Por esta razón el resultado obtenido desciende a un 80%.
- **Flash Builder.** Se integra con los sistemas operativos Windows, Linux y OS X, y presenta opciones de desarrollo para android e iOS, siendo el que más opciones presenta en cuanto a interoperabilidad, se toma como referencia todas sus prestaciones para el análisis. Obteniendo un 100% de interoperabilidad.

Mantenibilidad.

- **Basic4Android.** Al no contar con sus herramientas como complementos en forma de plugins por ejemplo, como es el caso de Eclipse y Flash Builder, no se puede actualizar con facilidad. Obtiene un 80% de mantenibilidad.
- **Eclipse.** Por su capacidad para complementarse con pulgins se puede actualizar fácilmente. Así como también permite dar mantenimiento para cambiar de resolución a sus proyectos, lo que beneficia al desarrollar aplicaciones móviles, que se usan en dispositivos con varias dimensiones de pantalla y resolución de los mismos. Se obtiene un 100% de mantenibilidad.
- **Flash Builder.** De la misma manera que eclipse alcanza un 100% de mantenibilidad. Con un comportamiento similar, por su facilidad para actualizarse mediante plugins y para cambiar de resolución a sus proyectos.

Portabilidad.

- **Basic4Android.** En cuanto a la portabilidad, este framework únicamente se instala en Windows. Tampoco permite compartir proyectos de otros frameworks. Por esta razón alcanza un 20%.
- **Eclipse.** Otorga la facilidad para migrar tanto el framework como sus proyectos se acopla a todas los sistemas operativos OS X, Windows y Linux. Permitiendo compartir sus proyectos entre todas las plataformas. Usando la máquina virtual de java como base de su desarrollo. Obtiene un 100% de portabilidad.
- **Flash Builder.** Presenta un 80% de portabilidad debido a que se acopla perfectamente con Windows, OS x y Linux, sin embargo, no se permite migrar el framework entre los sistemas operativos.

Una vez analizados los criterios se puede definir que el framework más óptimo es Flash Builder, pese al no presentar un buen resultado en

cuanto a eficiencia, este criterio está medido en cuanto a espacio físico, consumo de memoria, tiempo de instalación, etc. En la actualidad no es complicado conseguir un equipo que cuente con los requerimientos necesarios para su correcto funcionamiento.

2.6. Conclusiones.

Al terminar el estudio de este capítulo se puede definir que un estudio de calidad para frameworks, no es nada fácil, ya que la mayoría de modelos de calidad están orientados a medir productos software como sistemas o aplicaciones más no herramientas de desarrollo.

Así mismo para definir las medidas de las métricas se tiene que realizar de manera empírica, búsqueda de comentarios y experiencias personales. Sobre el desempeño de las herramientas para poder establecer los valores y criterios de medición de las características.

CAPÍTULO III

3. Análisis, diseño, programación e implementación de una aplicación móvil (Reservación de canchas sintéticas) según el framework seleccionado.

3.1 Diseño de interfaz.

"La creación de la experiencia del usuario puede parecer abrumadora complejidad. Con tantas cuestiones en juego - facilidad de uso, identidad de marca, arquitectura de información, diseño de interacción, puede parecer como si la única manera de construir un sitio exitoso es que gastar una fortuna en los especialistas que conocen todos los detalles. Por esta y muchas más razones Jesse James Garret utiliza los 5 planos para de esta manera hacer el diseño mucho más sencillo y eficaz." (Garret).

3.1.1 Plano Estrategia

Al implementar la estrategia se tiene una visión acerca de cuáles son los objetivos o que requiere el usuario, de esta manera se pretende:

- Definir lo que se quiere lograr para los usuarios.
- Facilitar la toma de decisiones en la implementación del diseño de interfaz basado en todos los aspectos de la experiencia del usuario.

Objetivos del Producto

Generar un diseño intuitivo el cual permita manipular directamente los objetos de la interfaz.

Captar la atención del usuario, y la libre exploración de la interfaz, sin miedo a consecuencias negativas.

Generar un modelo que cumpla con los principios del diseño de interfaz.

Captar las habilidades cognitivas y de percepción de los usuarios e implementarlas en la generación del diseño de interfaz.

Desarrollar una presentación visual clara.

Consistencia en la apariencia estética (íconos, fuentes, colores, organización).

Notificar al usuario indicaciones sobre el proceso que se está siguiendo.

Objetivos del Negocio

Brindar el servicio de reserva de cancha sintética dentro de la ciudad de Cuenca.

Ahorrar tiempo de búsqueda de canchas disponibles mediante aplicación móvil.

3.1.2 Plano Alcance

"Del lado del software el alcance es determinado conociendo las especificaciones funcionales del sistema, es decir un registro detallado de las características y operación.

Del lado de la información, en el plano de alcance se determinan los requerimientos de contenido.

En ambos casos el alcance depende enteramente del Plano de Estrategia. Debe de haber un balance entre los objetivos del sitio y las necesidades del usuario."(Área.com.mx)

En este plano se pretende convertir la estrategia en requerimientos, utilizando elementos funcionales para satisfacer las necesidades de los usuarios.

Del listado de requerimientos obtenidos en el plano estratégico se selecciona los que son viables para el presente proyecto, los cuales se detallan en el presente plano.

Posibles conflictos o irregularidades

- Ingreso erróneo de los horarios de reservación.
- Ingreso erróneo de los nombres de las canchas.
- Ingreso erróneo de los valores de costo de reservación de cancha.
- Reservación en cancha equivoca.
- Ingreso erróneo en datos personales de usuarios y propietarios.

Funcionalidad y Contenido

El objetivo de implementar este portal es permitir a los usuarios reservar canchas sintéticas dentro de la ciudad de Cuenca, de acuerdo a su elección, ya que podrán obtener la dirección en la que están ubicadas así como también su disponibilidad y precio.

Requerimientos Globales.

- Mostrar información requerida por los usuarios.
- Mostrar información requerida por los propietarios.
- Diseño correcto de la aplicación para que los usuarios y propietarios puedan manipular la misma de manera fácil y rápida.

Requerimientos Específicos

- Se implementara un diseño agradable a la vista del usuario, con la ayuda de herramientas que permitan que nuestra aplicación tenga la funcionalidad esperada.

- Permitir al usuario el ingreso y consulta de la información mediante formularios interactivos, con una iconicidad adecuada, distribuida de forma correcta.

Especificaciones Funcionales

- Permitir opción de registro a los usuarios.
- Los usuarios registrados podrán tener acceso a la siguiente información:

    1. Mantenimientos de reservas.
    2. Consultas sobre canchas disponibles.

- Permitir opción de registro a los propietarios de las canchas.
- Los propietarios registrados podrán tener acceso a la siguiente información:

    1. Mantenimiento de horarios de atención.
    2. Mantenimiento de canchas
    3. Mantenimiento de costos de cancha.
    4. Mantenimiento de información de canchas.
    5. Consulta de reservaciones por cancha.

Priorizar Requisitos

1. Diseño de ambiente de reservas.
2. Diseño de ambiente de registro de usuarios y propietarios.
3. Diseño de ambiente de creación de canchas y horarios.
4. Diseño de ambiente de consulta de reservaciones.
5. Detalle de información de cada una de las canchas.

3.1.3 Plano Estructura

"Define el lugar de los elementos de la interfaz, la estructura definirá cómo es que los usuarios pueden llegar a qué página, de donde a donde, etc."(Aceves)

Este plano contiene toda la organización, es decir el comportamiento que tendrá cada una de las piezas que conforman la aplicación.

La aplicación está basada en una estructura de tipo jerárquica es decir que se necesita cumplir necesariamente cada uno de los pasos y en orden de acuerdo a su prioridad:

1. Identificar las necesidades que tienen los usuarios.
2. Obtener los actores que son los que van a intervenir en la aplicación.
3. Realizar el diseño, codificación y pruebas de nuestra aplicación.
4. Realizar la prevención para evitar errores.
5. Realizar la corrección de errores.

Control de Errores

Una de las partes más importantes que debemos tener en cuenta al momento de la elaboración de una aplicación, es estar siempre al tanto de corregir errores; ya que por el contrario provocará que los usuarios se desinteresen por utilizar nuestra aplicación, es por esto que hemos tomado en cuenta lo siguiente:

Prevención

Cuando un usuario o un propietario se esté registrando se indicara de forma visual muy claro aquellos campos que son necesarios (obligatorios) mediante mensajes o símbolos que hacen que para el usuario sea un placer utilizar nuestro sistema y no una perturbación.

De igual forma cuando el usuario reserve una cancha se le indicara si el horario ya está ocupado o en su defecto si olvida ingresar la hora, el día, entre otros será informado previamente.
En el caso de los propietarios cuando estén creando canchas u horarios se les informará si no están ingresando toda la información requerida, de la misma manera se les indicara con mensajes cuando la información se ha guardado exitosamente.
Tanto para usuarios como para propietarios se les informará cuando quieran pasar de pantalla sin haber seleccionado una cancha, horario, entre otros.

Corrección

Al momento que se ingresan datos que no corresponden a los pedidos por el sistema se le informará al usuario, por ejemplo: si ingresa letras en el precio, valores incorrectos en las horas, etc.

Arquitectura de Información

Estructura de contenido

Utilizaremos una organización jerárquica de tal forma que las personas que accedan a la aplicación puedan utilizar cada una de las funcionalidades de la misma tanto para usuarios como para propietarios.
La arquitectura jerárquica le permitirá al usuario moverse en cada uno de los siguientes módulos:

- Registro de Usuarios.
- Información de las Canchas.
- Reservación de Canchas.
- Registro de Propietarios.
- Información de Reservas.
- Creación de Canchas.

- Creación de Horarios.

Le daremos al usuario o al propietario la facilidad de ir de un lugar a otro mediante pestañas que le indicarán en donde se encuentran y a donde pueden desplazarse.

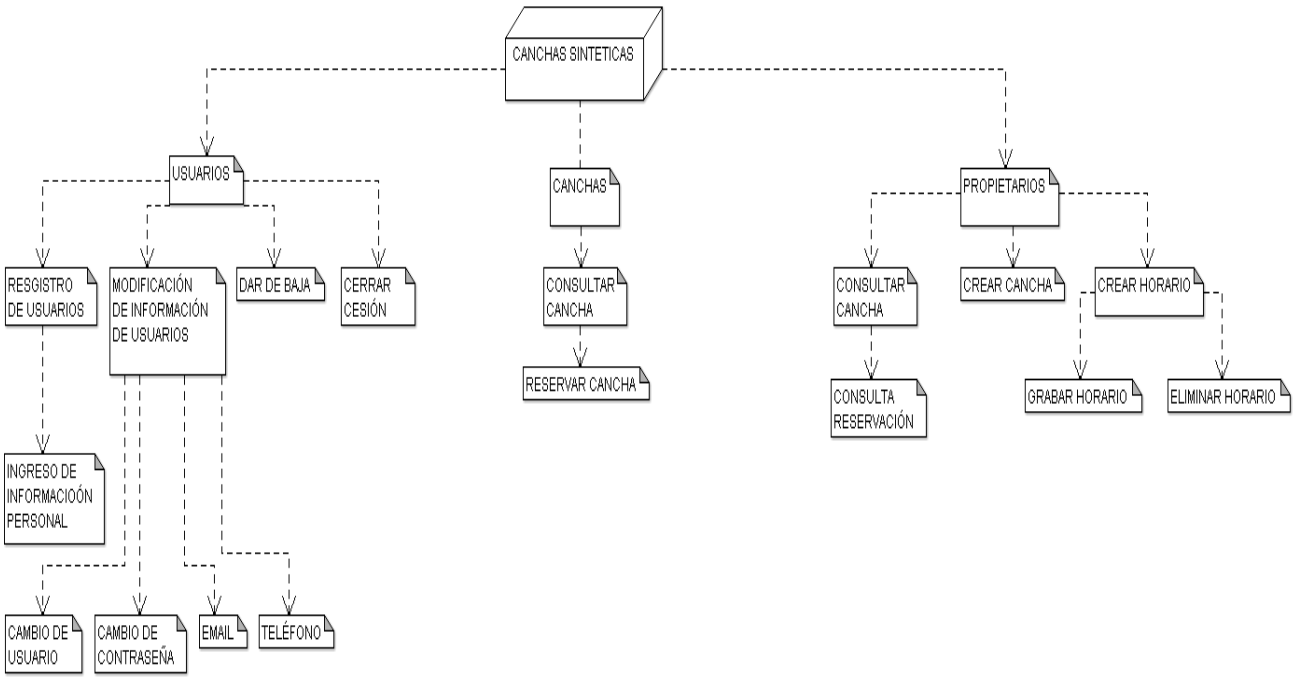

Gráfico 06 Estructura de contenido

Principios Organizacionales

La aplicación tendrá la siguiente organización

- Usuarios.
    1. Ingresos.
    2. Modificación.
    3. Eliminación.
- Canchas.
    1. Consultas.
    2. Reservación.
- Propietarios.
    1. Consultas.
    2. Creación de Canchas.
    3. Creación de Horarios.

3.1.4 Plano Esqueleto

En el plano del esqueleto del sitio, se define el lugar de los botones, las fotos, imágenes, bloques de texto, etc. El esqueleto está diseñado para optimizar el acomodo de los elementos que componen la aplicación para maximizar la eficiencia del uso o interacción de los usuarios con misma.(Aceves)

Diseño de Interfaz

- El logo de la aplicación de reserva de canchas sintéticas se pondrá para identificar la aplicación es decir estará solamente para acceder a la misma.
- En la parte inferior se situarán pestañas que son el menú de la aplicación.
- Todas las instancias o páginas además usaran los botones del dispositivo móvil para navegar por la aplicación.
- Los colores en general deberán matizar de una manera armónica para que tengan relación con los íconos elegidos para la aplicación.

El diseño de interfaz para el Inicio de sesión será el siguiente:
En esta pantalla el usuario deberá ingresar su nombre o pseudónimo, y su contraseña, para poder iniciar sesión, también podrá pasar a la pantalla de registro en caso de ser necesario.

Gráfico 07 Inicio de Sesión

Una de las primeras opciones es el registro como usuario, el diseño de usuarios posee el siguiente diseño:

En esta pantalla el usuario deberá ingresar la información requerida por la aplicación para poder iniciar sesión en un futuro.

Gráfico 08 Inicio de Sesión

Posterior a estar registrado en la aplicación y luego de haber pasado por el Inicio de sesión obtenemos el menú, el cual consta de dos partes:

1. Canchas

Se presenta una lista de las canchas que se tienen ingresadas previamente, así como sus datos principales, como son la dirección y el teléfono.

Gráfico 09 Selección de Canchas

Una vez seleccionada la cancha que se desea reservar y pasará a la pantalla en la cual se realiza la reservación:

Aquí se tienen los horarios y tarifa en los cuales se puede reservar la cancha anteriormente seleccionada.

Gráfico 10 Reservación de Canchas

2. Usuarios

La aplicación da la opción de modificar ciertos parámetros importantes de los usuarios, eliminar un usuario del sistema, cerrar cesión.

- Modificar

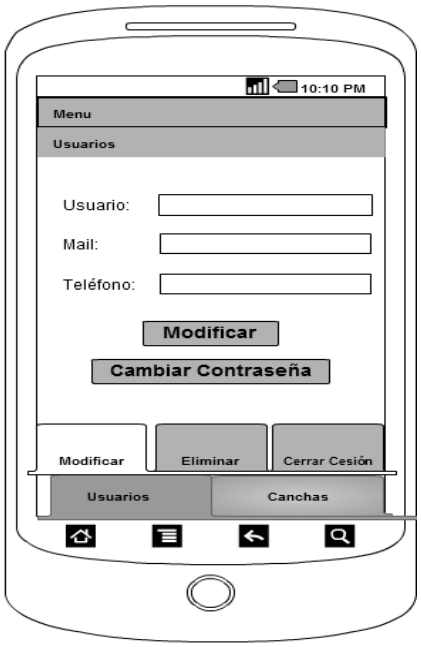

Gráfico 11 Modificación de Información de usuario

Cuando se modifican parámetros del usuario se tiene la opción de cambiar la contraseña, esta modificación se la hace verificando primero la contraseña actual y luego permitiendo ingresar la nueva contraseña.

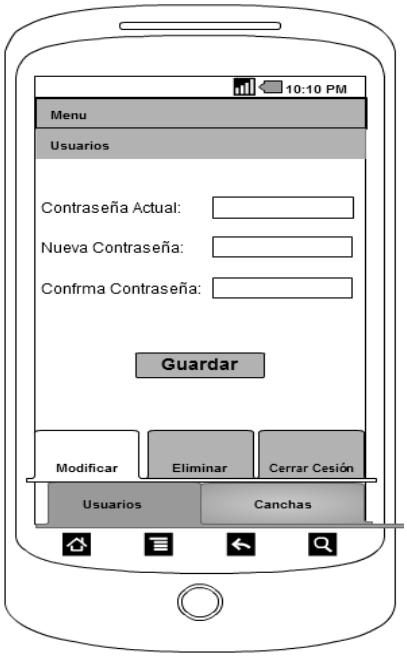

Gráfico 12 Cambio de Contraseña

- Eliminar

En esta pantalla se deberá ingresar el usuario para poder eliminarlo.

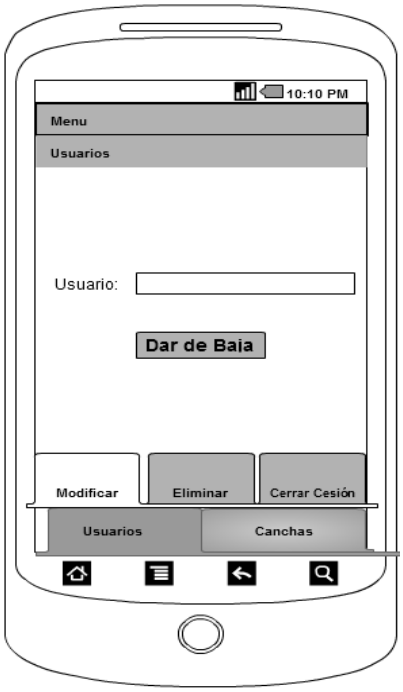

Gráfico 13 Eliminación de Usuario

- Cerrar Cesión

Esta pantalla permite cerrar la sesión con la cual un usuario realiza sus actividades dentro de la aplicación.

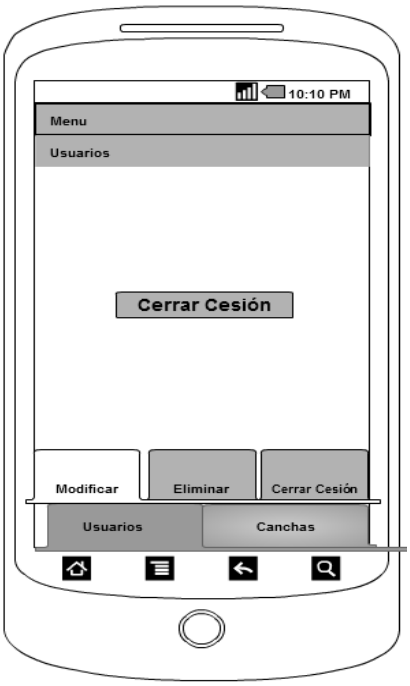

Gráfico 14 Cerrar Sesión

En la parte del Inicio de sesión se encuentra la opción “Propietario” la cual lleva a la siguiente pantalla:

Aquí se debe ingresar el nombre y contraseña de un usuario para poder iniciar una sesión como propietario.

Gráfico 15 Inicio de sesión de propietario

Al ingresar como propietario se puede ver en la pantalla las canchas que se posee.

Gráfico 16 Listado de canchas del propietario

El menú consta de tres partes que se describen a continuación:

1. Canchas

Los dueños ven la lista de las canchas de las cuales son propietarios

Seleccionan una cancha y obtienen una lista con todas las reservaciones, se mostrará la hora en la que los jugadores ingresarán y la hora en la que saldrán de la cancha, la tarifa por la que fue reservada, la fecha en la que se va a hacer uso de la cancha y por último el teléfono del usuario que hizo la reserva.

Gráfico 17 Reservaciones

2. Creación de Canchas

En la siguiente pantalla se indica cómo será el diseño para ingresar los datos cuando se necesite realizar la creación de una nueva cancha sintética.

Gráfico 18 Creación de canchas

3. Creación de Horarios

Esta opción nos permitirá ingresar o eliminar un horario dependiendo de la cancha que seleccionemos.

Gráfico 19 Creación y eliminación de horarios

Diseño de Navegación.

Parte importante de la organización de la estructura de la aplicación. Debe facilitar al usuario moverse con soltura y facilidad por las distintas pantallas de la aplicación, que encuentre lo que busca, que no se pierda yendo de un lugar a otro.(Gamboa)

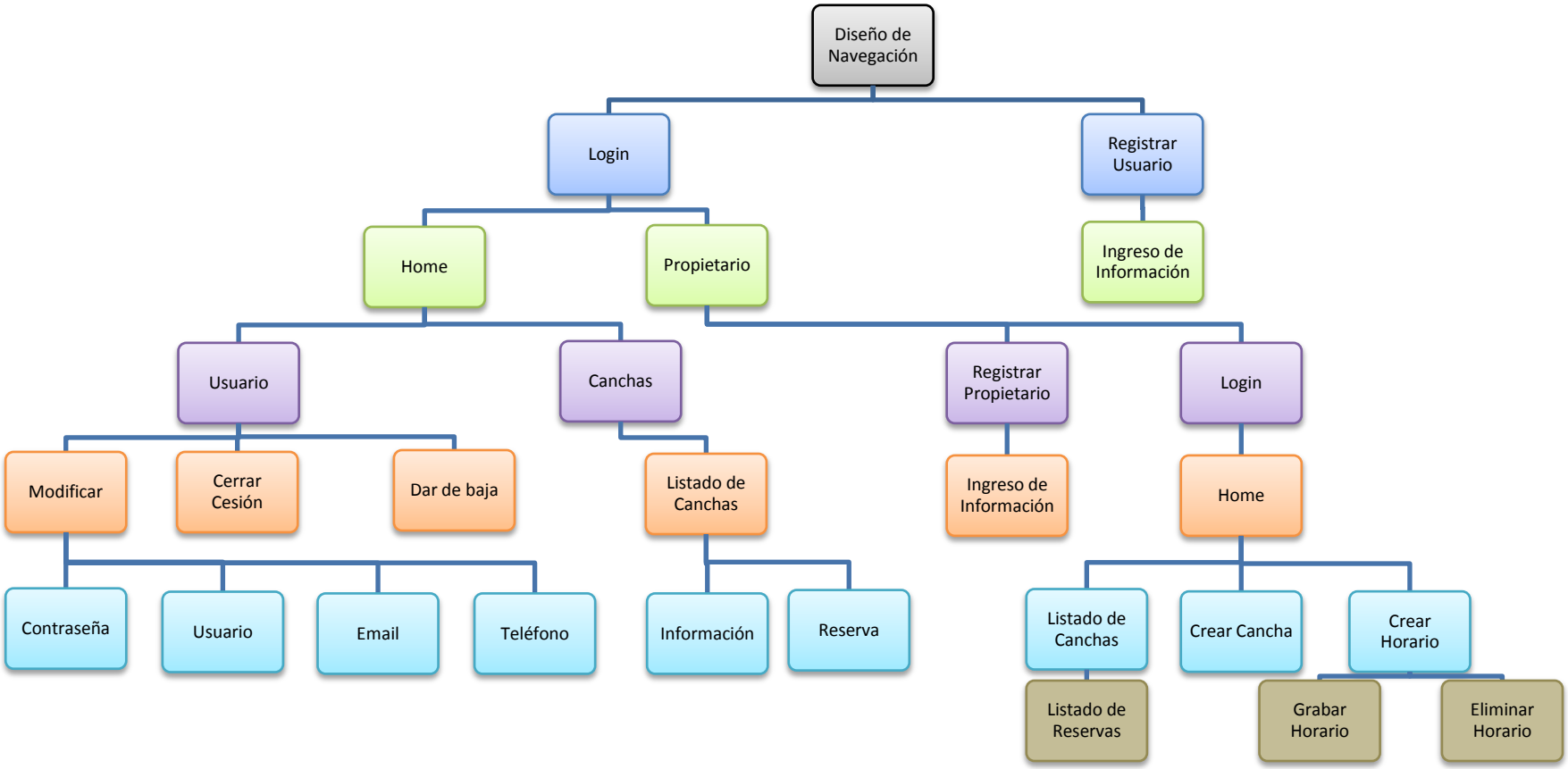

Gráfico 20 Diseño de navegación

Diseño de Información.

La información se presenta de forma clara para el usuario.
A continuación se detalla la información necesaria en cada una de las opciones con la que contará la aplicación:
Registro de Usuarios

1. Usuario
2. Contraseña
3. Nombres
4. Apellidos
5. Teléfono
6. Dirección
7. Correo

Registro de Propietarios

1. Usuario
2. Contraseña
3. Nombres
4. Apellidos
5. Teléfono
6. Dirección
7. Correo

Creación de Canchas

1. Nombre
2. Dirección
3. Teléfono

Creación de Horarios

1. Hora de Ingreso
2. Hora de Salida
3. Tarifa (precio)

Reserva de Canchas

1. Selección de Cancha.

2. Selección de Horario.
3. Selección de Fecha.

Verificación de Información

1. Selección de Cancha.

3.1.5 Plano Superficie

En la superficie de una aplicación, se aprecian imágenes y texto, en los cuáles, algunos son sólo de información, otros para realizar alguna función específica. Este plano hace referencia a la apariencia de la aplicación ante los usuarios.(Aceves)

Este es un esquema inicial de lo que será nuestra aplicación en el que se utiliza tipografía, colores, entre otros.

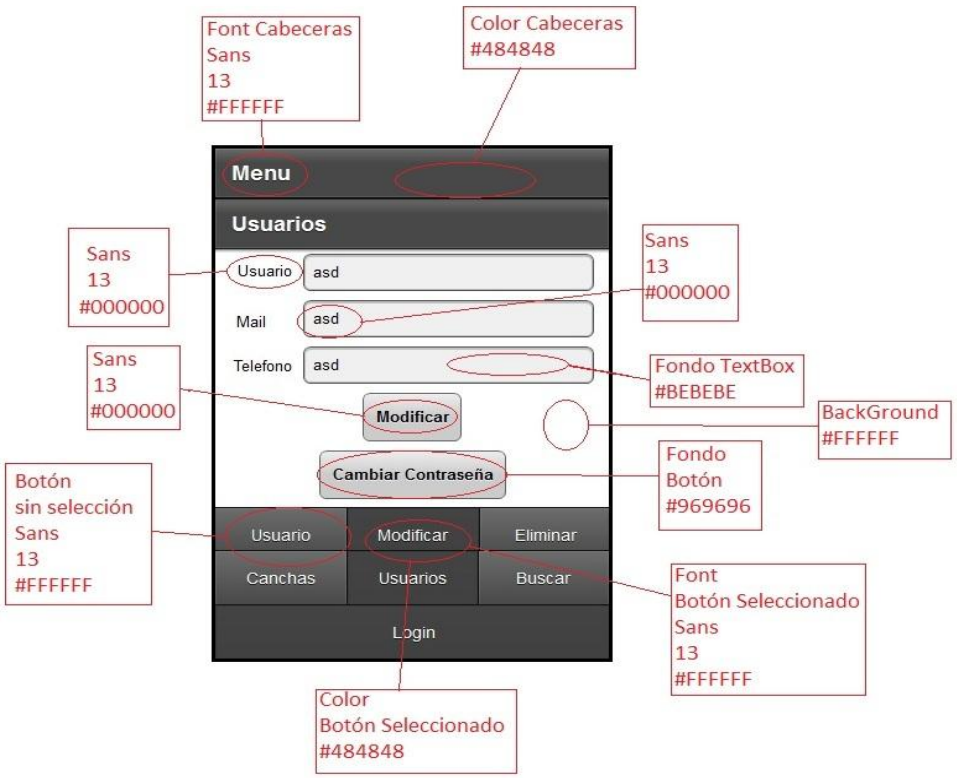

Gráfico 21Modboard

Paleta de Colores y Contraste.

Hemos determinado que los colores que se muestran a continuacion son los que vamos a utilizar para que el usuario tenga una buena impresión de la pagina y la interfaz sea amigable.

#BEBEBE

Gráfico 22 Paleta de Colores (#BEBEBE)

#969696

Gráfico 23 Paleta de Colores (#969696)

#5E5F61

Gráfico 24 Paleta de Colores (#5E5F61)

#484848

Gráfico 25 Paleta de Colores (#484848)

#000000

Gráfico 26 Paleta de Colores (#000000)

#FFFFFF

Gráfico 27 Paleta de Colores (#FFFFFF)

3.2. Conexión a la base de datos.

Para nuestra aplicación utilizaremos SQLite que es un sistema para gestionar bases de datos relacionales, es compatible con ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad).

El motor de SQLite no es un proceso independiente con el que el programa principal se comunica ya que se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones lo cual reduce la latencia en el acceso a la base de datos por que las llamadas a funciones son más eficientes que la comunicación entre procesos.

SQLite usa un sistema de tipos inusual debido a que en lugar de asignar un tipo a una columna como en la mayor parte de los sistemas de bases de datos SQL, los tipos se asignan a los valores individuales. Por ejemplo, se puede insertar un string en una columna de tipo entero sin que exista ningún inconveniente como la mayoría de gestores de bases de datos.

Tal como se indica la conexión como tal se efectúa mediante programación dentro de nuestra aplicación.

3.3. Análisis de la Aplicación

Para el análisis de la aplicación se utiliza el Lenguaje de Modelado Unificado (UML: Unified Modeling Language) es la sucesión de una serie de métodos de análisis y diseño orientadas a objetos que aparecen a

fines de los 80's y principios de los 90s.UML es llamado un lenguaje de modelado, no un método. Los métodos consisten de ambos de un lenguaje de modelado y de un proceso. El UML incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño orientados a objetos. El lenguaje de modelado es la notación (principalmente gráfica) que usan los métodos para expresar un diseño. El proceso indica los pasos que se deben seguir para llegar a un diseño.(Gonzalez Cornejo)

Dentro del UML se utiliza:

- Diagrama de casos de uso: Representa la forma en cómo un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso).(Gonzalez Cornejo)
- Diagrama de secuencia: Describen la interacción entre los objetos de una aplicación y los mensajes recibidos y enviados por los objetos.(Altova)
- Modelo conceptual: Se le define como una especificación del dominio del problema a través de la representación mediante objetos.(Monar) Representa conceptos, relaciones y reglas de restricción del mundo real. (Universidad Politécnica de Catalunya)
- Modelo Mental: Representa la estructura del sistema, la representación del comportamiento y de la interacción.(Akner)
- Diagrama entidad relación: Este modelo representa a la realidad a través de un esquema gráfico empleando la terminología de entidades, que son objetos que existen y son los elementos principales que se identifican en el problema a resolver con el diagramado y se distinguen de otros por sus características particulares denominadas Atributos, el enlace que rige la unión de las entidades está representado por la relación del modelo.(Ecuared)

3.3.1. Diagrama de Casos de Uso

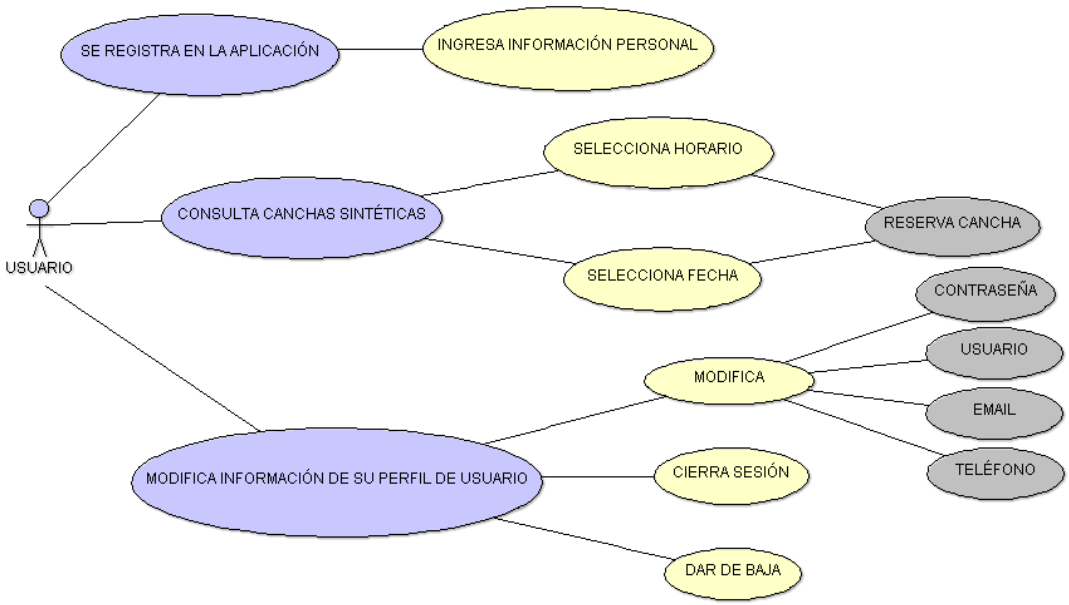

Gráfico 28 Diagrama de casos de uso (usuario)

Este diagrama representa las actividades del usuario normal dentro de la aplicación.

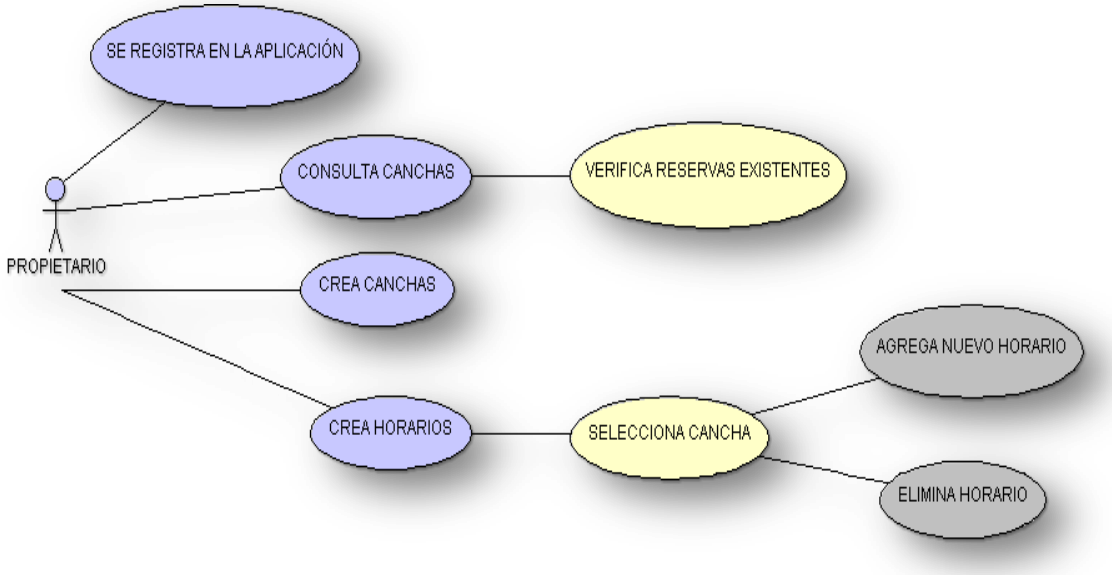

Gráfico 29 Diagrama de casos de uso (propietario)

Este diagrama representa las actividades del usuario propietario de una cancha dentro de la aplicación.

3.3.2. Diagrama de Secuencia

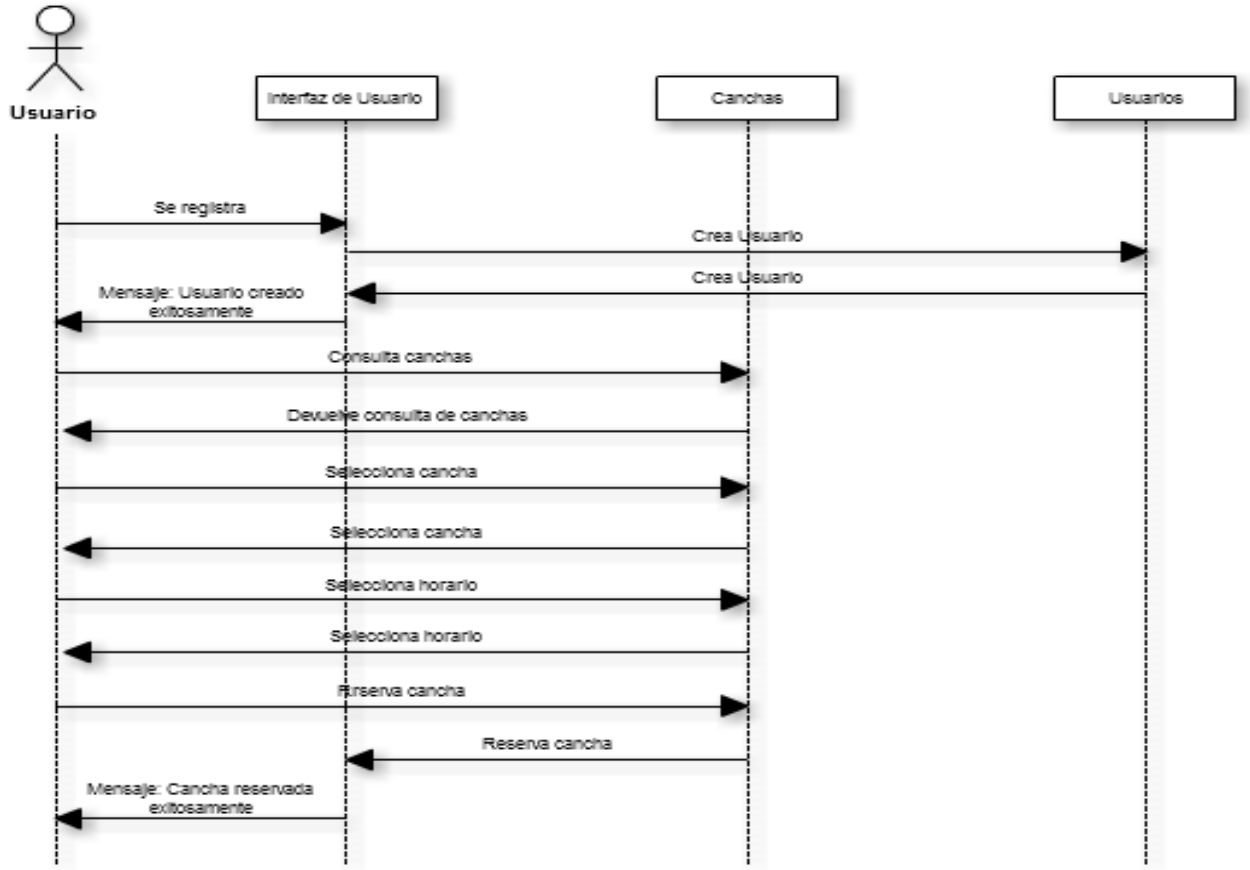

Gráfico 30 Diagrama de secuencia (usuario)

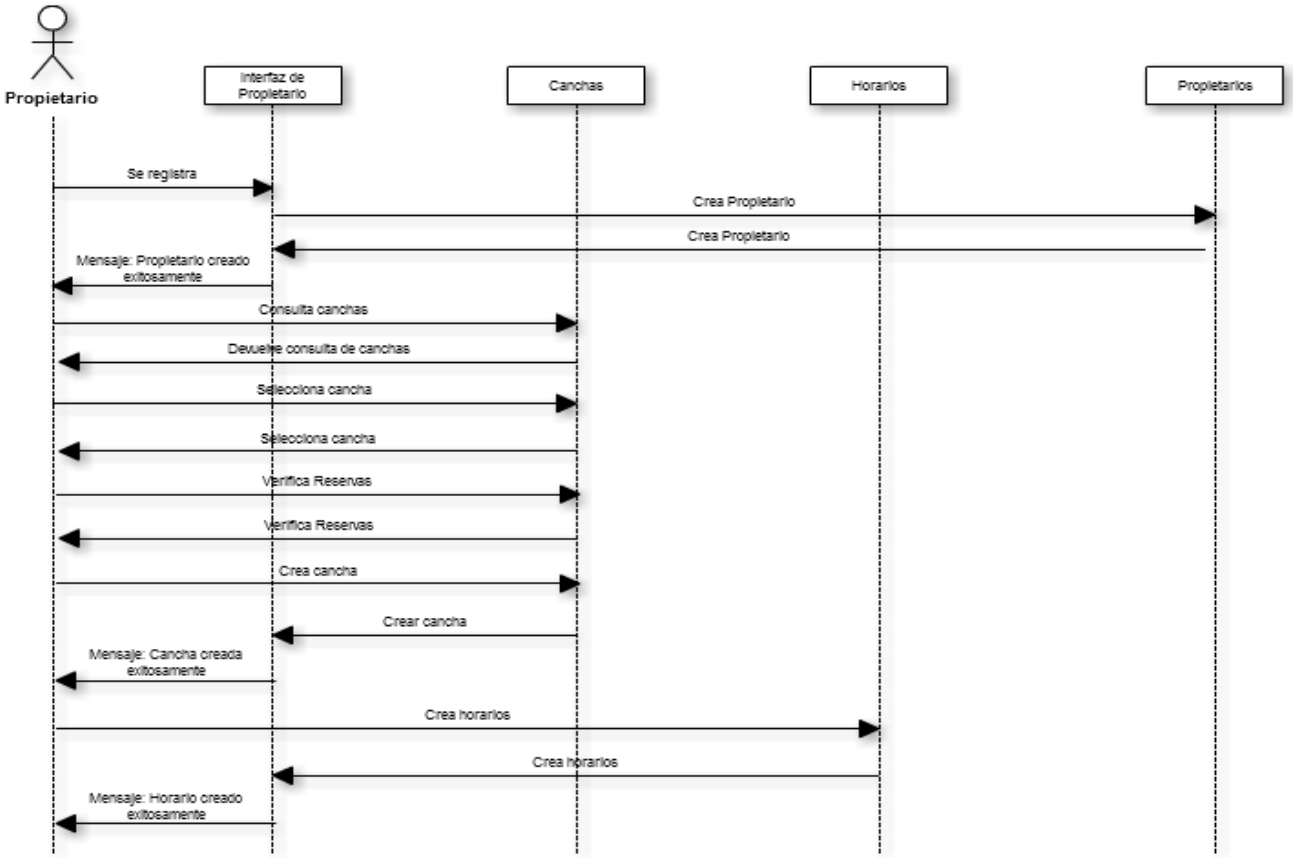

Gráfico 31 Diagrama de secuencia (propietario)

3.3.3. Modelo Conceptual

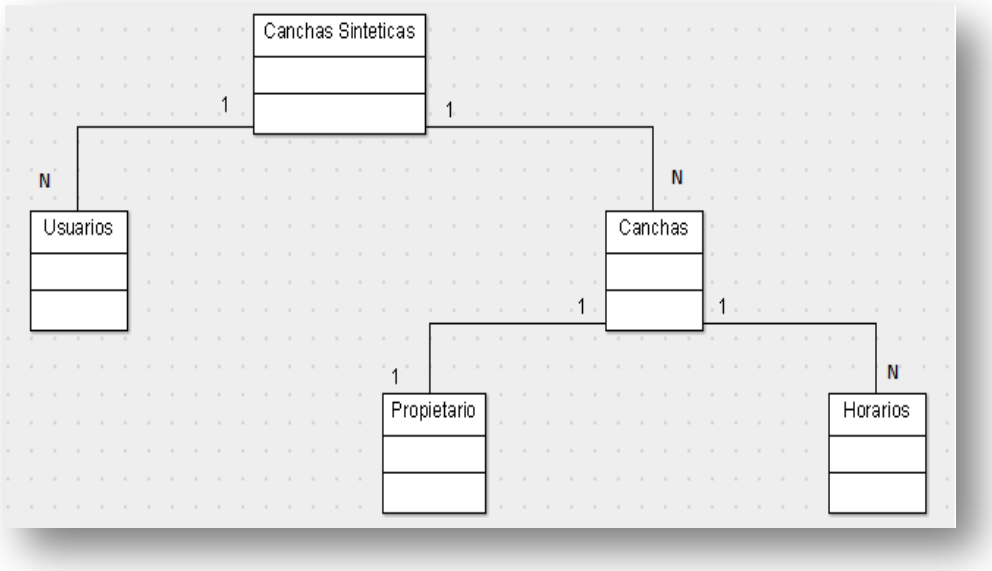

Gráfico 32 Modelo conceptual

3.3.4. Modelo Mental

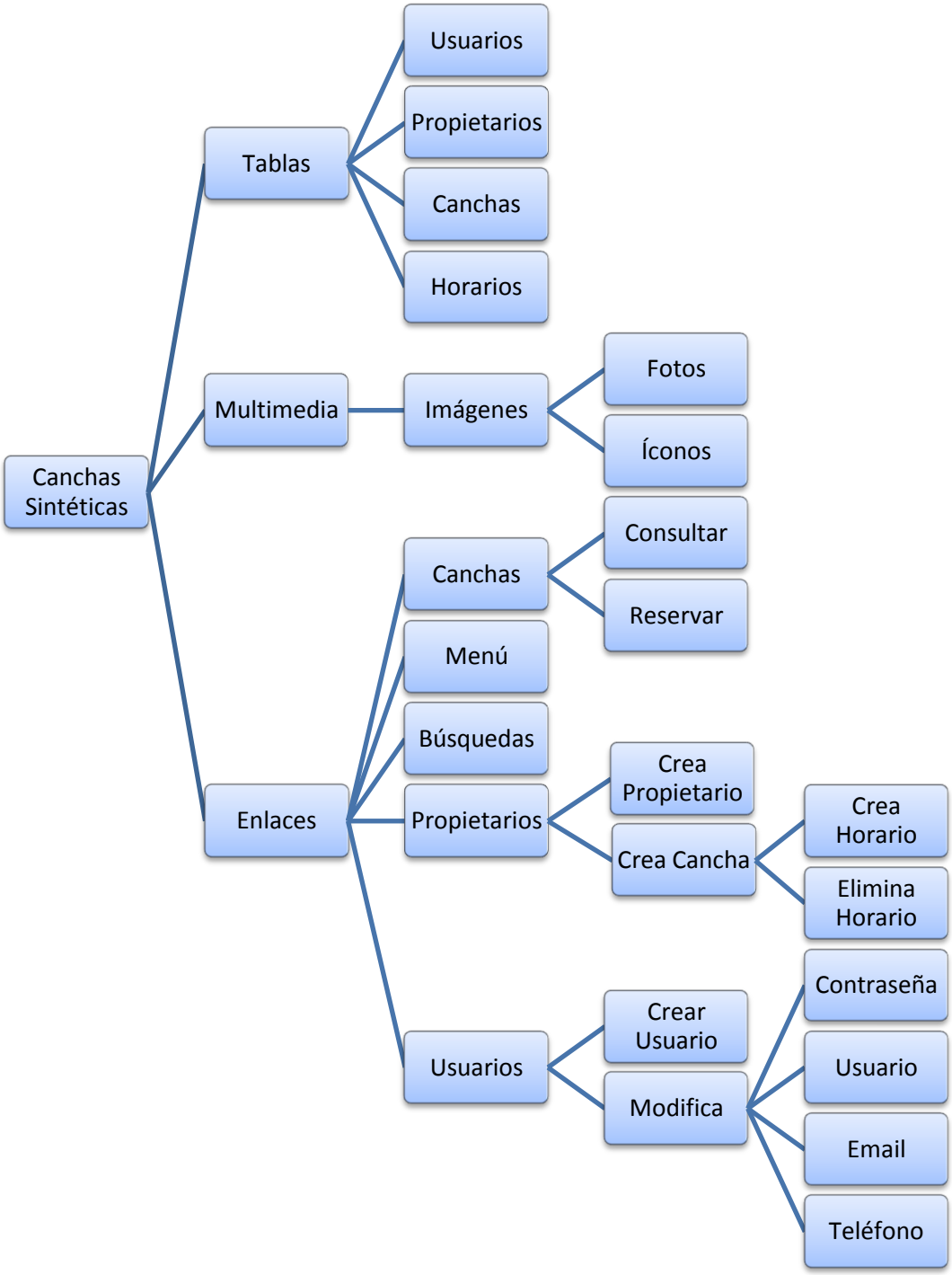

Gráfico 33 Modelo Mental

3.3.5. Diagrama Entidad Relación

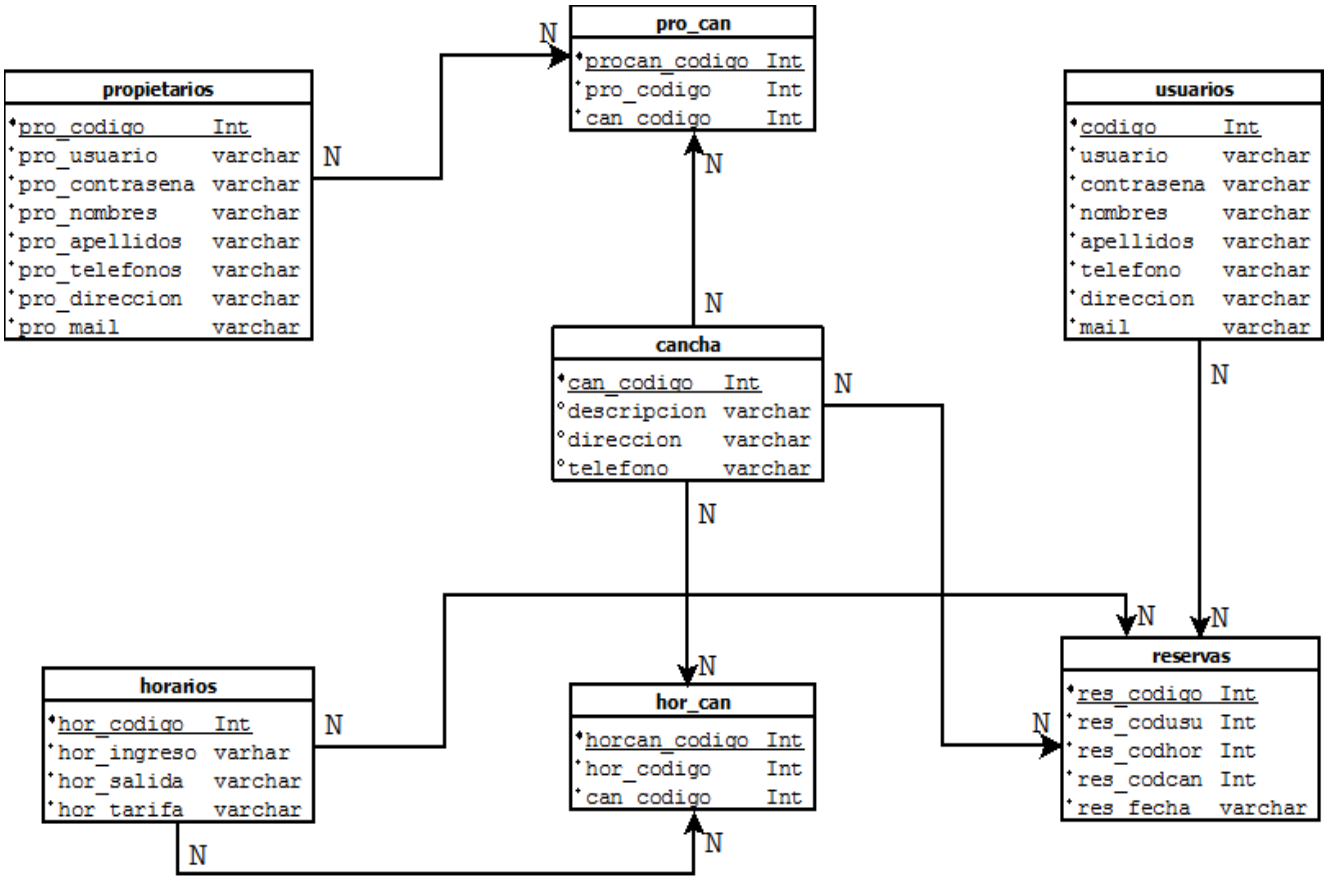

Gráfico 34 Diagrama entidad relación

3.4. Programación de la aplicación.

La programación la hemos realizado mediante el framework Adobe Flash Builder ya que fue el resultado del análisis realizado en el capítulo II en el que pudimos determinar mediante métricas que esta aplicación proporciona una serie de ventajas frente a las otras herramientas investigadas.

3.5. Pruebas de usabilidad y corrección de errores de la aplicación desarrollada.

Se realizó una serie de pruebas de usabilidad que nos permitieron hacer la respectiva corrección de errores a continuación detallamos en el presente cuadro las pruebas realizadas con sus respectivas correcciones y el número de usuarios que realizaron cada una de las pruebas.

PROBLEMA	SOLUCIÓN	NÚMERO DE USUARIOS QUE REALIZARON LAS PRUEBAS
Ingreso erróneo de la reservación	Se cuenta con un mantenimiento para la edición o eliminación de la reservación.	2
Ingreso erróneo del nombre de las canchas	Se cuenta con un mantenimiento para la edición de los datos de la cancha en el que se contempla no solo editar el (nombre, dirección y teléfono de la cancha).	5
Ingreso erróneo de los datos personales de usuarios y propietarios	Hay la posibilidad de editar los datos de usuario, contraseña y correo electrónico de los usuarios.	3
Elección erróneo de la fecha de reserva	Se realizan las validaciones respectivas para que la fecha y hora sean posteriores a la fecha actual cuando hacemos una reservación.	2

Tabla 21 Pruebas de usabilidad y corrección de errores

3.6. Conclusiones.

Mediante la realización del presente capítulo hemos podido obtener un conjunto de gráficos muy importantes para poder comprender de mejor manera el funcionamiento de la aplicación. También se ha tratado todo lo que se refiere al diseño de interfaz, paleta de colores, modboard, entre otros, componentes gráficos que son un aspecto fundamental de la parte visual.

De igual forma no hemos descuidado un tema importante como es la correcta conexión a la base de datos, programación y sobre todo corrección de errores.

CONCLUSIONES

- Para la definición de un modelo de calidad que permita evaluar frameworks es necesario un gran trabajo, ya que no existe un modelo establecido para este tipo de herramientas, la mayoría de los modelos de calidad están orientados a proyectos y productos software como sistema y aplicación más no están orientados a frameworks.
- Al momento de la selección de una herramienta para el desarrollo de una aplicación, no se debe considerar: que tan fácil es desarrollar el código fuente, si no, las prestaciones que brinda para que el resultado sea del desarrollo sea un buen producto, que no presenta problemas al momento de dar soporte, que se pueda acoplar a diferentes plataformas.
- El framework seleccionado brinda una gran facilidad al momento de desarrollar la aplicación, así mismo para aplicar los resultados del diseño y análisis de la aplicación.

RECOMENDACIONES

- Elaborar un modelo de calidad que permita hacer un análisis más completo de los pros y los contras del software que es orientado al desarrollo de aplicaciones para plataformas móviles.
- Definir métricas con las que se pueda analizar de manera mucho más clara todas las características que nos puede brindar un framework.
- Utilizar herramientas de software que nos permitan aplicar de manera más eficaz las métricas definidas en un determinado modelo de calidad

BIBLIOGRAFÍA

¿Qué es Adobe Flex? | TuFuncion. 14 de Julio de 2013 <http://www.tufuncion.com/flex>.

Aceves, Luis. 28 de 07 de 2013 <http://www.luiscarlosaceves.com/udem/dweb/theelementsofuserexperience.pdf>.

Adobe. Adobe - Flash Builder 4.5 Premium: Requisitos del sistema. 14 de julio de 2013 <http://www.adobe.com/store/es_es/popup/software/flashbuilder4_5/systemreqs.html>.

Adobe Flash Builder descarga gratis.15 de Juilo de 2013 <http://www.softpedia.es/programa-Adobe-Flex-Builder-120309.html>.

Akner. Mapa Mental -UML. 23 de 09 de 2009. 29 de 07 de 2013 <http://akner-akner.blogspot.com/2009/09/mapa-mental-uml.html>.

Altova. Diagramas de secuencia UML. 29 de 07 de 2013 <http://www.altova.com/es/umodel/sequence-diagrams.html>.

Anónimo. FURPS. 29 de noviembre de 2008. 20 de mayo de 2013 <http://clases3gingsof.wetpaint.com/page/FURPS>.

—. Modelo De Calidad De Software. octubre de 2010. 20 de mayo de 2013 <http://www.buenastareas.com/ensayos/Modelo-De-Calidad-De-Software/993610.html>.

—. «Organización y Estructura de la Información.» 22 de mayo de 2013 <http://www.oei.eui.upm.es/Asignaturas/PInformaticos/ficheros/temario/PROYINF-2.pdf>.

—. «Universidad del Valle-Cali Colombia.» 28 de mayo de 2013 <http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/Metricas4.pdf>.

Área.com.mx. Los Elementos de la Experiencia de Usuario - Área. 04 de 02 de 2004. 22 de 07 de 2013 <http://www.area.com.mx/estrategia/los-elementos-de-la-experiencia-de-usuario.php>.

Basic4android - Free download and software reviews - CNET Download.com. 6 de Marzo de 2013. 15 de Julio de 2013 <http://download.cnet.com/Basic4android/3000-2247_4-75578603.html>.

Betzabeth Pereira, Farid Ayaach, Henry Quintero, Ismael Granadillo, Jomar Bustamante. «Universidad Simón Bolívar.» 27 de mayo de 2013 <http://ldc.usb.ve/~abianc/materias/ci4712/metricas.pdf>.

Camacho, Erika, Fabio Cardeso y Gabriel Nuñez. «Universidad Simón Bolivar.» 05 de junio de 2013 <http://prof.usb.ve/lmendoza/Documentos/PS-6116/Guia%20Arquitectura%20v.2.pdf>.

Civil, Dirección General de Servicio. «Index of /dgsc/indiceGestion/6TECINFOC/preg59seguridadplanutic.» 24 de mayo de 2013 <http://www.dgsc.go.cr/dgsc/indiceGestion/6TECINFOC/preg59seguridadplanutic/ModelocalidadsoftwareUTIC.pdf>.

Coimbra, Victor Hugo Gutierrez. Ingenieria de Software. 21 de abril de 2012. 01 de junio de 2013 <http://gutierrezcoimbra.blogspot.com/2012/04/modelos-de-calidad-en-el-software.html>.

Desarrolloweb.com. Métricas del software. 20 de mayo de 2013 <http://www.desarrolloweb.com/articulos-copyleft/articulo-metricas-de-software.html>.

DIVERTEKA. Programa tu Android en Basic. 14 de julio de 2013 <http://www.diverteka.com/?p=1258>.

Doria, Heidi González. «Colección de Tesis Digitales.» 22 de mayo de 2013 <http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/gonzalez_d_h/capitulo4.pdf>.

Eclipse. 15 de julio de 2013 <http://curso-sobre.berlios.de/introsobre/2.0.1/sobre.html/eclipse.html>.

Ecuared. Diagrama Entidad Relación - EcuRed. 29 de 07 de 2013 <http://www.ecured.cu/index.php/Diagrama_Entidad_Relaci%C3%B3n>.

Enterprises, Jelsoft. Noticia Basic4Android – Crea tu aplicación para Android con Visual Basic [Archivos] - HTCMania. 14 de julio de 2013 <http://www.htcmania.com/archive/index.php/t-298283.html>.

Fillottrani, Pablo R. «Departamento de Ciencias e Ingeniería de la Computación.» 01 de junio de 2013 <http://www.cs.uns.edu.ar/~prf/teaching/SQ07/clase6.pdf>.

Gallo, David. Prezi. 24 de abril de 2013. 12 de junio de 2013 <http://prezi.com/1hdb3-kv6lbq/iso-9126/>.

Gamboa, Salvador. DISEÑOS DE NAVEGACIÓN. 28 de 07 de 2013 <https://sites.google.com/site/comodisenarunapaginaweb/disenos-de-navegacion>.

García, José Alberto. Anónimo. 29 de mayo de 2013 <http://www.google.com.ec/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0CDQQFjAB&url=http%3A%2F%2Fdiaweb.usal.es%2Fdiaweb%2Farchivos%2F10011069INSA_Presentacion_CMMI.pdf&ei=yeGGUZ_rJ5TY8gSihIGQBQ&usg=AFQjCNGORvoumv6m7465i0haKvpR6wEUDg&bvm=bv.45960087,d.eWU>.

Garret, Jesse James. The Elements of User Experience.2010.

Gómez, David Alejandro. ISO 9126. 25 de noviembre de 2009. 20 de mayo de 2013 <http://alejandrogomeziso.blogspot.com/>.

Gonzalez Cornejo, José Enrique. El Lenguaje de Modelado Unificado (UML). 13 de 01 de 2010. 30 de 06 de 2013 <http://www.docirs.cl/uml.htm>.

Jiménez, José Luis. «José Luis Jiménez.» 02 de junio de 2013 <http://josejimenez.zzl.org/des/des5/me.pdf>.

Melendez, Karin y Abraham Dávila. «Pontificia Universidad Católica de Perú.» 23 de mayo de 2013 <http://inform.pucp.edu.pe/~edavila/publicaciones/calidadproductosoftware_ok.pdf>.

Mendoza, Gonzalo Mena. Anónimo. 27 de mayo de 2013 <http://mena.com.mx/gonzalo/maestria/calidad/presenta/iso_9126-3/>.

Monar, Oscar. El modelo conceptual del lenguaje UML contiene tres elementos. 29 de 07 de 2013 <http://www.oocities.org/es/bcontrerasrodriguez/AnalisisyDisenodeSistemas/foro/forouml.htm>.

Palazzolo, Cecilia. «No Quality Inside.» 2005. 02 de junio de 2013 <http://noqualityinside.com/nqi/nqifiles/CalidadDeSW_diap.pdf>.

Ruiz, José Joaquín. «Calidad y Medición de SI.» 23 de mayo de 2013 <http://alarcos.inf-cr.uclm.es/doc/cmsi/trabajos/Joaquin%20Ruiz.pdf>.

Sánchez, Victor Adrián. «Slideshare.» 06 de junio de 2013 <http://www.slideshare.net/hopdie/metricasmmm>.

Scalone, Fernanda. «Laboratorio de Sitemas Inteligentes.» 22 de mayo de 2013 <http://laboratorios.fi.uba.ar/lsi/scalone-tesis-maestria-ingenieria-en-calidad.pdf>.

Software, AnyWhere. Basic4android (Basic for Android) - Android programming with Gui designer.14 de julio de 2013 <http://www.basic4ppc.com/android/downloads.html>.

Systems, Sparx. Sparx Systems - Productos - MDG Integration for Eclipse - Requisitos de sistema. 14 de julio de 2013 <http://www.sparxsystems.com.ar/products/mdg/int/eclipse/sysreq.html>.

Tecnológoco de Monterrey. «slideshare.» 28 de mayo de 2013 <http://www.slideshare.net/JekittaB/calidad-del-producto-iso-9126>.

Tellez, Linda Luna. «scribd.» 28 de abril de 2012. 29 de mayo de 2013 <http://es.scribd.com/doc/91676203/Estandares-de-Calidad>.

Universidad Politécnica de Catalunya. Universidad Politécnica de Catalunya. 29 de 07 de 2013 <http://cursoo.files.wordpress.com/2009/03/reglastransformacionuml_espe.pdf>.

Valencia, Yeisson y Omar Alexus Velásquez. «Slideshare.» 30 de mayo de 2013 <http://www.slideshare.net/guest768516/modelo-de-calidad-de-desarrollo-de-software-cmmi-3212350>.

Viramontes Aguilar, Marisol, y otros. «scribd.com.» scribd.com. 10 de 06 de 2013 <http://es.scribd.com/doc/136281463/metricasingenieriadesoftware-100310182600-phpapp02>.