UNIVERSIDAD DE EL SALVADOR
FACULTAD MULTIDISCIPLINARIA ORIENTAL
DEP. INGENIERIA Y ARQUITECTURA

Puntos de Casos de Uso

Asignatura: Diseño de Sistemas II

Docente: Ing. Ligia Astrid Bonilla Hernández

Presentan:

García Meléndez, Yenifer Zuleyma GM15002
Hernández Hernández, Stefany Magaly HH14012
Iraheta Medrano, Luis José IM15005

Ciudad Universitaria - Ciclo II
Lunes 15 de Octubre de 2018.

Introducción

Este presente trabajo se desarrolla el diseño de un sistema de puntos de caso de uso en tres plataformas diferentes para lo cual se tiene que desarrollar diversos procesos.

Su principal ventaja es su rápida adaptación a empresas que ya estén utilizando la técnica de Casos de Uso. Por lo cual se requiere de un sistema que automáticamente realice las operaciones sin necesidad de molestarnos en estar haciendo muchos cálculos, con el objetivo de minimizar el tiempo al realizar todo este proceso.

Se analizarán los requerimientos necesarios especificados por el usuario para tener una mejor visión de lo que desea y presentarle un diseño óptimo. De esta manera buscamos que sea fácil y rápido el manejo de este sistema, que sea funcional y práctico para el usuario.

Se especificarán el modelo de ciclo de vida, paradigma de programación, la estructura del software y el patrón o anti patrón de dicha aplicación con el fin de tener un mejor control y conocimiento de lo que contendrá el sistema en su etapa de desarrollo. Tomando en cuenta los factores de costo, tiempo y adaptabilidad se analizan las ventajas y los beneficios que tiene que contar con este sistema para ahorrar tiempo en hacer las operaciones.

Contenido

I. Planteamiento del problema

1.1. Situación problemática

Uno de los principales problemas a los que nos enfrentamos los desarrolladores de software al momento de planear nuestros proyectos es la estimación. Existen distintas técnicas que nos permiten estimar proyectos de software, cada una de ellas con sus ventajas y desventajas, pero la mayoría de ellas no ofrecen la flexibilidad de estimar software orientado a objetos, y se basan prácticamente en la experiencia del equipo de desarrollo. La técnica de estimación con puntos de caso de uso nos permite realizar estimaciones a partir de modelos orientados a objetos con una precisión bastante aceptable.

Además de esto no existen muchas herramientas para determinar los puntos de caso de uso, aunque se puede hacer una plantilla en Excel, pero software no existe ninguna según lo investigado, por ello se pretende desarrollar un software que permita nada más ingresar valores y automáticamente darnos la respuesta que deseamos sin necesidad que gastar mucho tiempo realizándolo a mano. Con esto se pretende ahorrar mucho tiempo para los usuarios. Y guardar el resultado de todas estas operaciones para uso posterior.

Es por eso que surge la problemática de conocer todos estos valores en corto tiempo y de manera exacta y no contando con un software para conocerlo. Es por ello el desarrollo del software que de forma fácil y sin muchos procesos se conozca el resultado final.

1.2. Objetivos

General

- Desarrollar los puntos de caso de uso mediante consola, formulario y una aplicación para dispositivos móviles.

Específicos

- Establecer las clases y métodos a utilizar para el desarrollo en cada una de las formas de desarrollo.
- Calcular de manera precisa cada uno de los puntos de caso de uso
- Estimar de manera consciente las importancias que incurre cada punto en el desarrollo de los programas a diseñar.
- Mostrar pasados puntos de casos de uso ha calculados.

1.3. Justificación

Los Puntos de caso de uso son un método de estimación de esfuerzo para proyectos de software, a partir de sus casos de uso, se basan en el método de punto de función, y supervisado.

El método utiliza los actores y casos de uso relevados para calcular el esfuerzo que significará desarrollarlos. A los casos de uso se les asigna una complejidad basada en transacciones, entendidas como una interacción entre el usuario y el sistema, mientras que a los actores se les asigna una complejidad basada en su tipo, es decir, si son interfaces con usuarios u otros sistemas. También se utilizan factores de entorno y de complejidad técnica para ajustar el resultado.

Este proyecto busca realizar las esas evaluaciones que se realizan a la hora de desarrollar un sistema, y así evaluar cada uno de los puntos necesarios que se calculan en los puntos de caso de uso. De forma que el desarrollador facilite y agilice estos cálculos para un proyecto actual, también se analizan o comparan resultados anteriores de proyectos anteriores; evaluando la dinámica del sistema se eligen diferentes modelos, arquitecturas o paradigmas que se ajusten a las necesidades de desarrollo del sistema ya sea en consola, formulario o ya sea una aplicación. Para esto el proyecto está desarrollado a base del modelo de ciclo de vida en cascada.

El modelo en cascada es un enfoque clásico en el desarrollo de software que describe un método de desarrollo lineal y secuencial. Consta de cinco a siete fases, cada fase está definida por diferentes tareas y objetivos, por lo que la totalidad de las fases describe el ciclo de vida del software hasta su entrega. Una vez finalizada una fase, sigue el siguiente paso de desarrollo y los resultados de la fase anterior pasan a la siguiente fase.

Un ejemplo de una metodología de desarrollo en cascada es:

- Análisis de requisitos.
- Diseño del sistema.
- Diseño del programa.
- Codificación.
- Pruebas.
- Implementación o verificación del programa.
- Mantenimiento.

Para la arquitectura de software usaremos el Modelo-vista-controlador (MVC) que es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones.

Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

Muchos sistemas informáticos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos que debe utilizar la aplicación; en líneas generales del MVC dicha gestión corresponde al modelo.

La unión entre capa de presentación y capa de negocio conocido en el paradigma de la Programación por capas representaría la integración entre la Vista y su correspondiente Controlador de eventos y acceso a datos, MVC no pretende discriminar entre capa de negocio y capa de presentación pero si pretende separar la capa visual gráfica de su correspondiente programación y acceso a datos, algo que mejora el desarrollo y mantenimiento de la Vista y el Controlador en paralelo, ya que ambos cumplen ciclos de vida muy distintos entre sí.

En el desarrollo interno de la aplicación y dentro de los paradigmas de programación se utiliza la Programación Orientada a Objetos (POO) es un paradigma de programación que viene a innovar la forma de obtener resultados. Los objetos manipulan los datos de entrada para la obtención de datos de salida específicos, donde cada objeto ofrece una funcionalidad especial.

Muchos de los objetos prediseñados de los lenguajes de programación actuales permiten la agrupación en bibliotecas o librerías, sin embargo, muchos de estos lenguajes permiten al usuario la creación de sus propias bibliotecas.

Está basada en varias técnicas, incluyendo herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento.

En diseño de software, el patrón de diseño Simple Factory donde uno de los que devuelve una instancia de una de varias clases posibles, según los datos que se le proporcionen. Esto implica que las clases que devuelve tienen la misma clase primaria y métodos, pero cada uno de ellos realiza la tarea de manera diferente para diferentes tipos de datos.

II. Marco Teórico

Aplicaciones Móviles

Una aplicación móvil, aplicación o app, es una aplicación informática diseñada para ser ejecutada en teléfonos inteligentes, tabletas y otros dispositivos móviles. Las aplicaciones permiten al usuario efectuar un conjunto de tareas de cualquier tipo profesional, de ocio, educativas, de acceso a servicios, etc., facilitando las gestiones o actividades a desarrollar. El acortamiento inglés app suele ser incorrectamente pronunciado por los hispanohablantes como /apepé/, tratándolo incorrectamente como una sigla.

Por lo general, se encuentran disponibles a través de plataformas de distribución, operadas por las compañías propietarias de los sistemas operativos móviles como Android, iOS, BlackBerry OS, Windows iPhone, entre otros. Existen aplicaciones móviles gratuitas u otras de pago, donde en promedio el 20 a 30 % del coste de la aplicación se destina al distribuidor y el resto es para el desarrollador. El término app se volvió popular rápidamente, tanto que en 2010 fue listada como la palabra del año de la American Dialect Society.

Al ser aplicaciones residentes en los dispositivos están escritas en algún lenguaje de programación compilado, y su funcionamiento y recursos se encaminan a aportar una serie de ventajas tales como:

- Un acceso más rápido y sencillo a la información necesaria sin necesidad de los datos de autenticación en cada acceso.
- Un almacenamiento de datos personales que, a priori, es de una manera segura.
- Una gran versatilidad en cuanto a su utilización o aplicación práctica.
- La atribución de funcionalidades específicas.
- Mejorar la capacidad de conectividad y disponibilidad de servicios y productos (usuario-usuario, usuario-proveedor de servicios, etc.).

Un sistema operativo es un programa o conjunto de programas informáticos que gestiona el hardware de un dispositivo y administra el servicio de aplicaciones informáticas (Windows, iOS, Android, etc.).

En los últimos años, los servicios de informática distribuida han permitido que las organizaciones, incluidas las educativas, puedan gestionar sus procesos, actividad y aplicaciones informáticas a través de empresas que ofrecen comercialmente software como servicio (SaaS) alojado en un centro de datos o en servicios en la nube, y grandes redes de ordenadores pueden formar una "malla" que representa una potencia considerable (Google, Amazon, Microsoft).

React Native: desarrollando apps nativas usando JavaScript

React Native es un framework desarrollado por Facebook que permite desarrollar apps nativas iOS y Android usando JavaScript. Desarrolla apps nativas usando JavaScript, sin Objective-C o Java de por medio. React Native lleva como un “puente” donde su función es la de traducir el código React Native en Objective-C, para el caso de iOS, y Java en Android.

Es algo fácil que permite a los programadores web poder desarrollar apps nativas sin tener que aprender nuevos lenguajes de programación muy específicos para cada una de las plataformas. A parte que la sintaxis de React Native es bastante clara y sencilla, además de heredar el mismo diseño que React, aportando flexibilidad y reaprovechamiento en el código. Faltaría añadir por último que el código fuente que desarrollemos, en nuestra primera app, será un 100% compartido con iOS y Android. Esto quiere decir que el código JavaScript que hagamos nos servirá tanto para la app en iOS como en Android.

Lo primero es que React Native compila código JS en el correspondiente código nativo directamente. Pero es bastante dura de realizar, ya que Java y Objective C/Swift son lenguajes fuertemente tipados mientras que JavaScript no lo es. En lugar de eso, RN hace algo más inteligente: React Native es, en esencia, un conjunto de componentes React, donde cada uno de ellos tiene su correspondiente equivalente en views y componentes nativos.

NetBeans

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos (Actualmente Sun Microsystems es administrado por Oracle Corporation).

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de softwarellamados *módulos*. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

El NetBeans IDE soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Ant, control de versiones y refactoring.

NetBeans IDE 6.5.2, la cual fue publicada el 19 de noviembre de 2008, extiende las características existentes del Java EE (incluyendo Soporte a Persistencia, EJB 3 y JAX-WS). Adicionalmente, el NetBeans Enterprise Pack soporta el desarrollo de Aplicaciones empresariales con Java EE 5, incluyendo herramientas de desarrollo visuales de SOA, herramientas de esquemas XML, orientación a web servicies (for BPEL), y modelado UML. El NetBeans C/C++ Pack soporta proyectos de C/C++, mientras el PHP Pack, soporta PHP 5.

Java (Lenguaje de Programación)

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como *WORA*, o "*write once, run anywhere*"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos diez millones de usuarios reportados.

La primera característica, orientado a objetos ("OO"), se refiere a un método de programación y al diseño del lenguaje. Aunque hay muchas interpretaciones para OO, una primera idea es diseñar el software de forma que los distintos tipos de datos que usen estén unidos a sus operaciones. Así, los datos y el código (funciones o métodos) se combinan en entidades llamadas objetos. Un objeto puede verse como un paquete que contiene el "comportamiento" (el código) y el "estado" (datos).

El principio es separar aquello que cambia de las cosas que permanecen inalterables. Frecuentemente, cambiar una estructura de datos implica un cambio en el código que opera sobre los mismos, o viceversa. Esta separación en objetos coherentes e independientes ofrece una base más estable para el diseño de un sistema software. El objetivo es hacer que grandes proyectos sean fáciles de gestionar y manejar, mejorando como consecuencia su calidad y reduciendo el número de proyectos fallidos.

III. Análisis

3.1. Análisis de requerimientos

3.1.1. Requerimientos Funcionales

- Calcular los puntos de Casos de Uso
- Se deben crear 3 proyectos en plataformas diferentes consola, formulario y una aplicación móvil.

3.1.2. Requerimientos no funcionales

- Interfaz tiene que ser amigable.
- La ventana que se ajuste a la pantalla.
- Que sea de fácil manejo para el usuario a la hora de los cálculos.
- No debe instalarse un servidor para ejecutar los programas.

3.2. Diseño lógico

3.2.1. Modelo de ciclo de vida

El término ciclo de vida del software describe el desarrollo de software, desde la fase inicial hasta la fase final. El propósito de este programa es definir las distintas fases intermedias que se requieren para validar el desarrollo de la aplicación, es decir, para garantizar que el software cumpla los requisitos para la aplicación y verificación de los procedimientos de desarrollo: se asegura de que los métodos utilizados son apropiados.

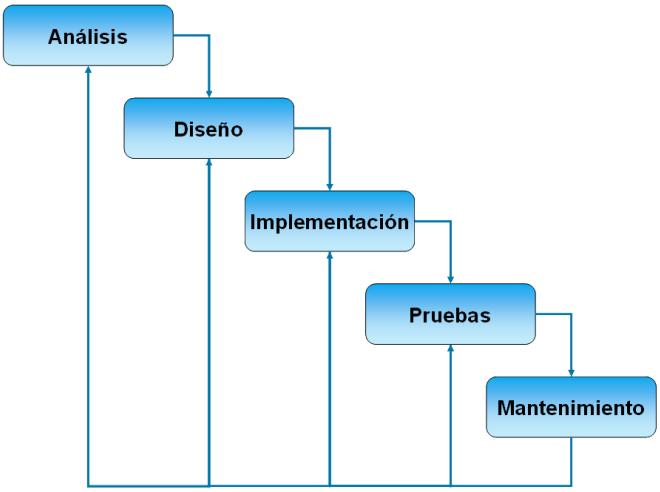
Estos programas se originan en el hecho de que es muy costoso rectificar los errores que se detectan tarde dentro de la fase de implementación. El ciclo de vida permite que los errores se detecten lo antes posible y, por lo tanto, permite a los desarrolladores concentrarse en la calidad del software, en los plazos de implementación y en los costos asociados.

	CASCADA	SCRUM	PROTOTIPO
RH	8	4	9
Tamaño del proyecto	9	7	8
Tiempo reducido	8	5	9
Costo	10	10	10
Documentación	10	7	10
Tipo de proyecto	9	8	7
Total	54	41	53

El modelo de ciclo de vida que se presentará para la aplicación será el siguiente:

Modelo de proceso basado en cascada

El modelo en cascada es un enfoque clásico en el desarrollo de software que describe un método de desarrollo lineal y secuencial. Consta de cinco a siete fases, cada fase está definida por diferentes tareas y objetivos, por lo que la totalidad de las fases describe el ciclo de vida del software hasta su entrega. Una vez finalizada una fase, sigue el siguiente paso de desarrollo y los resultados de la fase anterior pasan a la siguiente fase.



Beneficios / Desventajas

Algunas ventajas y desventajas del modelo en cascada:

Ventajas

- Debido a la estructura lógica del modelo, a menudo se pueden evitar errores conceptuales.
- El modelo conduce a una extensa documentación técnica, que es un alivio para los nuevos programadores y desarrolladores y también es útil en la fase de prueba.
- El progreso del proyecto puede ser monitoreado usando metas.
- El coste total puede estimarse con relativa precisión si no hay conflictos.

Desventajas

Los conflictos, bugs y errores de programación a veces conducen a un aumento de los costes y a una cantidad considerable de tiempo. Lo mismo se aplica si los clientes no están satisfechos.

Las especificaciones que se hacen inicialmente son a menudo difíciles de entender para los clientes porque son más abstractas de lo que se supone que el software debe hacer. Especialmente en proyectos subcontratados, esto puede ser una desventaja decisiva, ya que la fecha de lanzamiento debe posponerse y el mercado puede haber cambiado durante este tiempo.

La entrega del software lleva más tiempo porque los departamentos no trabajan simultáneamente y cada fase sólo puede comenzar cuando se ha completado la fase anterior.

Importancia para la programación

El modelo en cascada es uno de los modelos de proceso más conocidos en el desarrollo de software. Se ha utilizado con éxito durante décadas, pero ahora sólo se utiliza para proyectos más pequeños en los que las especificaciones son claras. Los inconvenientes antes mencionados, sin embargo, también llevaron a los analistas y desarrolladores a diseñar modelos alternativos llamados desarrollo ágil de software. El principal problema del modelo en cascada es que los cambios y revisiones no están necesariamente previstos por las secuencias lógicas.

La retroalimentación de los clientes, testers o probadores e ingenieros durante el desarrollo, está en parte ausente, y la integración del software en un sistema existente tiene lugar en un big bang. Estos inconvenientes pueden evitarse modificando las fases del proyecto, como es el caso del Modelo en Espiral. Pero desde hace algunos años, los métodos ágiles que utilizan otros elementos estructurales son mucho más populares (por ejemplo, los roles y sprints con Scrum o los principios extremos de programación). Por regla general, son más económicos, conducen a resultados más rápidos y son más transparentes para los clientes.

3.2.2. Paradigma de programación

Existe una infinidad de definiciones de lo que es un paradigma. Un paradigma es un determinado marco desde el cual miramos el mundo, lo comprendemos, lo interpretamos e intervenimos sobre él. Abarca desde el conjunto de conocimientos científicos que imperan en una época determinada hasta las formas de pensar y de sentir de la gente en un determinado lugar y momento histórico.

En nuestro contexto, el paradigma debe ser concebido como una forma aceptada de resolver un problema en la ciencia, que más tarde es utilizada como modelo para la investigación y la formación de una teoría. También, el paradigma debe ser concebido como un conjunto de métodos, reglas y generalizaciones utilizadas conjuntamente por aquellos entrenados para realizar el trabajo científico de investigación, los paradigmas de programación nos indican las diversas formas que, a lo largo de la evolución de los lenguajes, han sido aceptadas como estilos para programar y para resolver los problemas por medio de una computadora.

	Lógica	POO	Eventos
Manejo al aplicar el paradigma	**4**	**9**	**6**
Facilidad a la hora de implementar	**5**	**8**	**7**
Compatibilidad al uso de patrones	**2**	**8**	**7**
Adaptabilidad al proyecto	**2**	**9**	**8**
Total	**13**	**34**	**28**

Programación orientada a objetos

Una clase se puede definir de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ella. La Herencia es la facilidad mediante la cual la clase D hereda en ella cada uno de los atributos y operaciones de C, como si esos atributos y operaciones hubiesen sido definidos por la misma D. Por lo tanto, puede usar los mismos métodos y variables registrados como "públicos" o public en C. Los componentes registrados como "privados" o private también se heredan, pero se mantienen escondidos al programador y sólo pueden ser accedidos a través de otros métodos públicos. Para poder acceder a un atributo u operación de una clase en cualquiera de sus subclases, pero mantenerla oculta para otras clases es necesario registrar los componentes como "protegidos" (protected), de esta manera serán visibles en C y en D, pero no en otras clases.

Los Objetos son las Instancias de una clase. Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos), los mismos que consecuentemente reaccionan a eventos. Se corresponden con los objetos reales del mundo que nos rodea, o con objetos internos del sistema (del programa).

Los Métodos son los algoritmos asociados a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje". Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema.

En el Evento Es un suceso en el sistema (tal como una interacción del usuario con la máquina, o un mensaje enviado por un objeto). El sistema maneja el evento enviando el mensaje adecuado al objeto pertinente. También se puede definir como evento la reacción que puede desencadenar un objeto; es decir, la acción que genera.

El problema con la abstracción de datos es que no hay ninguna distinción entre las propiedades generales y las particulares de un conjunto de objetos. Expresar esta distinción

y aprovecharla es lo que define a la OOP a través del concepto de herencia. El paradigma de la programación orientada a objetos es, entonces:

a) Definir que clases se desean

b) Proporcionar un conjunto completo de operaciones para cada clase

c) Indicar explícitamente lo que los objetos de la clase tienen en común empleando el concepto de herencia.

En algunas áreas las posibilidades de la OOP son enormes. Sin embargo, en otras aplicaciones, como las que usan los tipos aritméticos básicos y los cálculos basados en ellos, se requiere únicamente la abstracción de datos y/o programación por procedimientos, por lo que los recursos necesarios para apoyar la POO podrían salir sobrando.

3.2.3. Arquitectura del software

Arquitectura de Software

La programación por capas es un modelo de desarrollo software en el que el objetivo primordial es la separación (desacoplamiento) de las partes que componen un sistema software o también una arquitectura cliente-servidor: lógica de negocios, capa de presentación y capa de datos. De esta forma, por ejemplo, es sencillo y sostenible crear diferentes interfaces sobre un mismo sistema sin requerirse cambio alguno en la capa de datos o lógica.

	Monolítica	2 Capas	3 Capas
Experiencia de uso	5	6	9
Recurso Humano	6	7	8
Tiempo	7	8	9
Tamaño del proyecto	5	8	10
Adaptabilidad del proyecto	7	8	10
Tipo de proyecto	7	8	8
Total	37	45	54

1. Capa de presentación: la que ve el usuario (también se la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.

2. Capa de negocio: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las

reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

3. Capa de datos: es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Todas estas capas pueden residir en un único ordenador, si bien lo más usual es que haya una multitud de ordenadores en donde reside la capa de presentación (son los clientes de la arquitectura cliente/servidor). Las capas de negocio y de datos pueden residir en el mismo ordenador, y si el crecimiento de las necesidades lo aconseja se pueden separar en dos o más ordenadores. Así, si el tamaño o complejidad de la base de datos aumenta, se puede separar en varios ordenadores los cuales recibirán las peticiones del ordenador en que resida la capa de negocio.

Si, por el contrario, fuese la complejidad en la capa de negocio lo que obligase a la separación, esta capa de negocio podría residir en uno o más ordenadores que realizarían solicitudes a una única base de datos. En sistemas muy complejos se llega a tener una serie de ordenadores sobre los cuales corre la capa de negocio, y otra serie de ordenadores sobre los cuales corre la base de datos.

En una arquitectura de tres niveles, los términos "capas" y "niveles" no significan lo mismo ni son similares.

El término "capa" hace referencia a la forma como una solución es segmentada desde el punto de vista lógico:

- Presentación. (Conocida como capa Web en aplicaciones Web o como capa de usuario en Aplicaciones Nativas)
- Lógica de Negocio. (Conocida como capa Aplicativa)
- Datos. (Conocida como capa de Base de Datos)

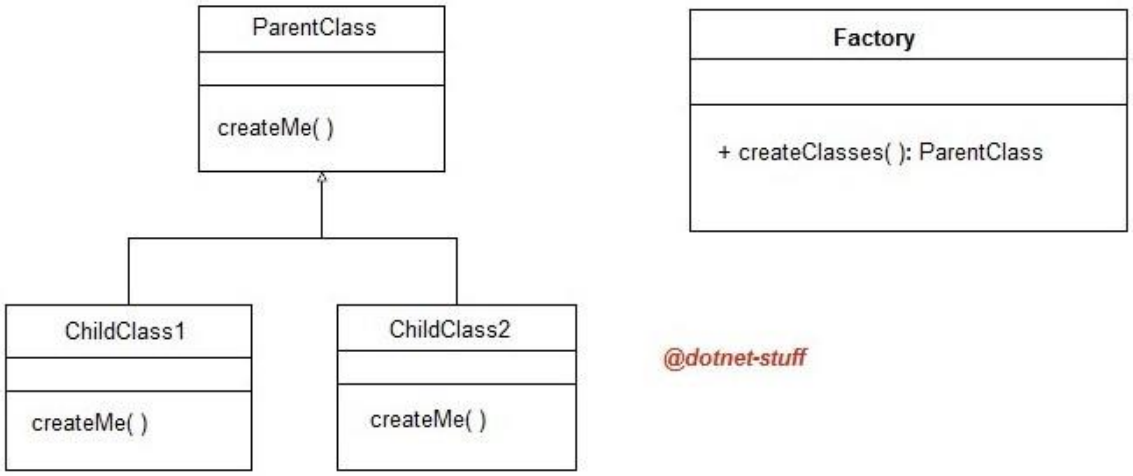
En cambio, el término "nivel" corresponde a la forma en que las capas lógicas se encuentran distribuidas de forma física. Por ejemplo:

Una solución de tres capas (presentación, lógica del negocio, datos) que residen en un solo ordenador (Presentación+lógica+datos). Se dice que la arquitectura de la solución es de tres capas y un nivel.

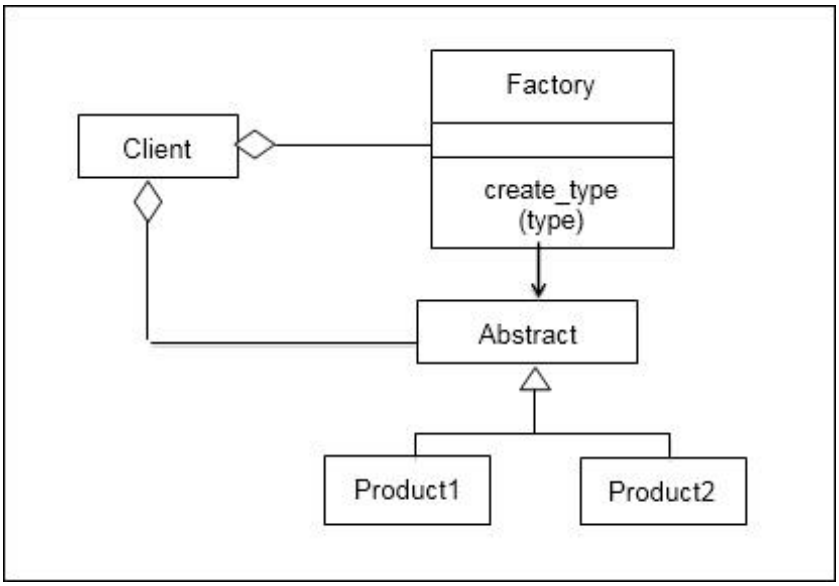
Una solución de tres capas (presentación, lógica del negocio, datos) que residen en dos ordenadores (Presentación+lógica por un lado; lógica+datos por el otro lado). Se dice que la arquitectura de la solución es de tres capas y dos niveles.

3.2.4. Patrones de diseño

Simple Factory Design Pattern no pertenece a las Gangs of Four. Un patrón de fábrica simple es uno de los que devuelve una instancia de una de varias clases posibles, según los datos que se le proporcionen. Esto implica que las clases que devuelve tienen la misma clase primaria y métodos, pero cada uno de ellos realiza la tarea de manera diferente para diferentes tipos de datos.



Se observa que hay dos subclases ChildClass1 y ChildClass2 tienen el mismo padre ParentClass, y hay una clase separada Factory. Clase de fábrica un método createClass con el tipo de retorno ParentClass. Esto implica que Factory puede devolver cualquier instancia de clases hijo. El método de clase infantil puede tener una implementación diferente dependiendo del requisito. El método en la lase Factory tiene la lógica para decidir qué clase debe devolver. Podría ser una lógica muy compleja o puede ser muy simple.



	Abstract Factory	Simple	Command
Dominio del patrón	8	8	7
Eficiencia	7	7	5
Facilidad	7	9	5
Encaje al sistema	7	7	6
Tipo de proyecto	8	9	7
Total	37	40	30

3.2.5. Bases de Datos

Una base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido; una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. Actualmente, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos están en formato digital, siendo este un componente electrónico, por tanto se ha desarrollado y se ofrece un amplio rango de soluciones al problema del almacenamiento de datos.

SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña biblioteca escrita en C. SQLite es un proyecto de dominio público creado por D. Richard Hipp.

A diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite 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. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción.

	FireBase	SQLite	MySQL
Experiencia de uso	6	8	9
Conexión	7	9	3
Relación al proyecto	7	9	5
Facilidad	6	8	8
Total	26	34	25

3.2.5. Diagramas

3.2.5.1. Diagrama de Casos de Uso

En el Lenguaje de Modelado Unificado, un diagrama de casos de uso es una forma de diagrama de comportamiento UML mejorado. El Lenguaje de Modelado Unificado (UML), define una notación gráfica para representar casos de uso llamada modelo de casos de uso. UML no define estándares para que el formato escrito describa los casos de uso, y así mucha gente no entiende que esta notación gráfica define la naturaleza de un caso de uso; sin embargo una notación gráfica puede solo dar una vista general simple de un caso de uso o un conjunto de casos de uso.

Los diagramas de casos de uso son a menudo confundidos con los casos de uso. Mientras los dos conceptos están relacionados, los casos de uso son mucho más detallados que los diagramas de casos de uso. En los conceptos se debe detallar más de un caso de uso para poder identificar qué es lo que hace un caso de uso.

- La descripción escrita del comportamiento del sistema al afrontar una tarea de negocio o un requisito de negocio. Esta descripción se enfoca en el valor suministrado por el sistema a entidades externas tales como usuarios humanos u otros sistemas.

- La posición o contexto del caso de uso entre otros casos de uso. Dado que es un mecanismo de organización, un conjunto de casos de usos coherentes y consistentes promueven una imagen fácil de comprender del comportamiento del sistema, un entendimiento común entre el cliente/propietario/usuario y el equipo de desarrollo.

En el diagrama de casos de uso de esta aplicación móvil muestra la interacción que tiene el usuario con la aplicación a desarrollar, explicando de manera gráfica y detallada los movimientos que realiza el usuario con el fin de cumplir el propósito de la aplicación que es mostrar la hora del país seleccionado.

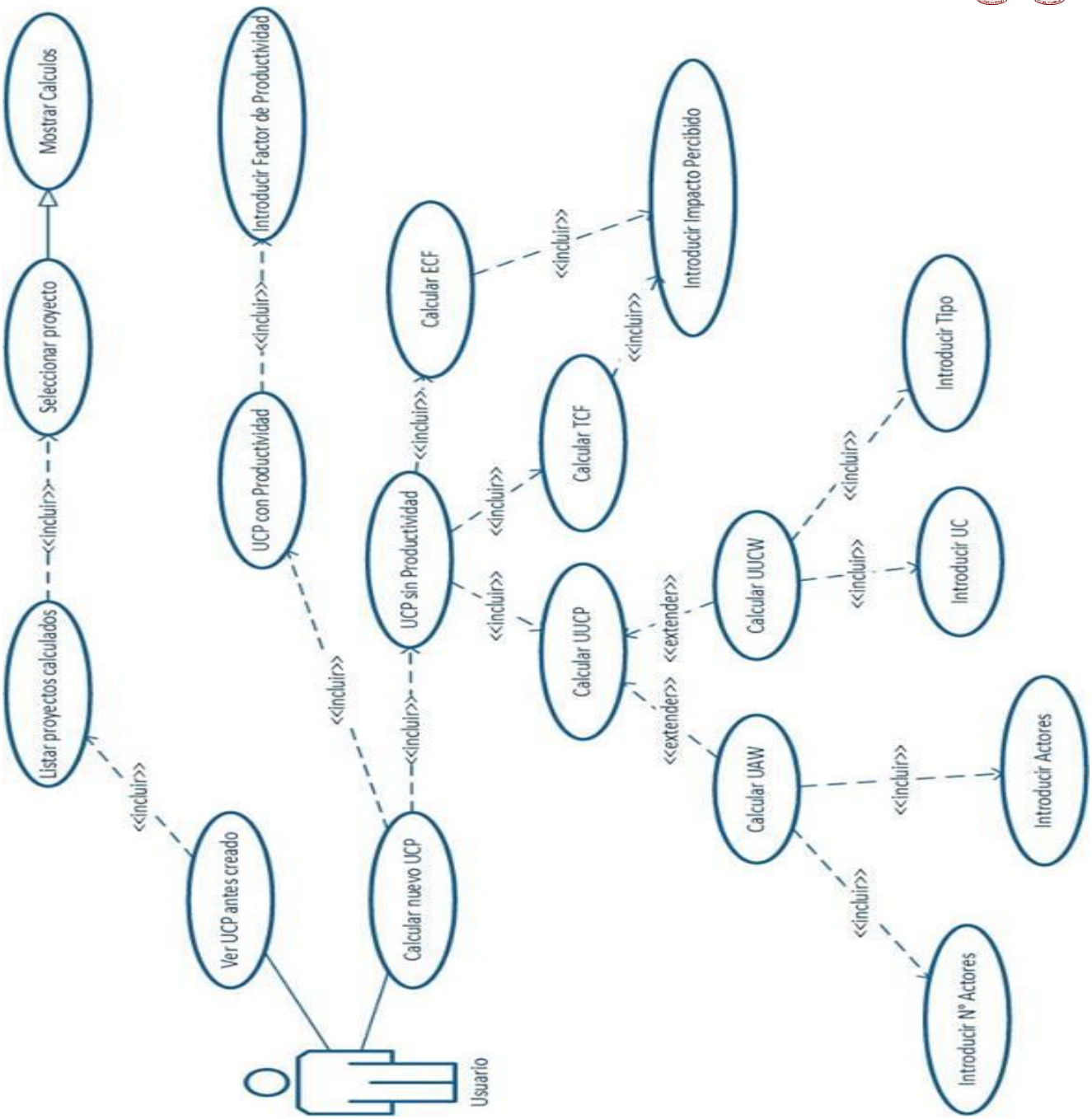


Diagrama 1: Casos de Uso

Sistema de puntos de caso de uso

Autores: Luis Iraheta, Yenifer García, Magaly Hernández

Fecha: lunes 1 de octubre de 2018

Versión: 1.0

Nombre de caso de uso:	Ver UCP antes creado	Caso de usos aplicado a la información del usuario.
ID del Caso de uso:	CU-0001	
Prioridad:	Alta	
Fuente:	Requerimiento CU-0001.R1	
Actor primario de negocios:	Usuario de sistema	
Otros actores participantes:	Ninguno	
Otros Involucrados interesados:	Ninguno	
Descripción:	Este caso de uso se muestra la opción si desea ver un UCP que se haya elaborado antes. Se le mostrara el listado y seleccionara el que desea ver. Y mostrara los cálculos de ese proyecto	

Sistema de puntos de caso de uso

Autores: Luis Iraheta, Yenifer García, Magaly Hernández

Fecha: lunes 1 de octubre de 2018

Versión: 1.0

Nombre de caso de uso:	Calcular nuevo UCP	Caso de usos aplicado a la información del usuario.
ID del Caso de uso:	CU-0002	
Prioridad:	Alta	
Fuente:	Requerimiento CU-0002.R3	
Actor primario de negocios:	Usuario de sistema	
Otros actores participantes:	Ninguno	
Otros Involucrados interesados:	Ninguno	
Descripción:	En este caso de uso se inicia toda la lógica del sistema para que el usuario pueda crear un nuevo UCP. Se pedirá de entrada el nombre del sistema para crear el UCP. Y también si va a crear un UCP con productividad o sin productividad.	

Sistema de puntos de caso de uso

Autores: Luis Iraheta, Yenifer García, Magaly Hernández

Fecha: lunes 1 de octubre de 2018

Versión: 1.0

Nombre de caso de uso:	Listar proyectos calculados	Caso de usos aplicado a la información del usuario.
ID del Caso de uso:	CU-0003	
Prioridad:	Alta	
Fuente:	Requerimiento CU-0003.R3	
Actor primario de negocios:	Usuario de sistema	
Otros actores participantes:	Ninguno	
Otros Involucrados interesados:	Ninguno	
Descripción:	Este caso de uso se listaran los proyectos ya creados para posteriormente el actor elija cual desea ver.	

Sistema de puntos de caso de uso

Autores: Luis Iraheta, Yenifer García, Magaly Hernández

Fecha: lunes 1 de octubre de 2018

Versión: 1.0

Nombre de caso de uso:	Seleccionar proyecto	Caso de usos aplicado a la información del usuario.
ID del Caso de uso:	CU-0004	
Prioridad:	Alta	
Fuente:	Requerimiento CU-0004.R1	
Actor primario de negocios:	Usuario de sistema	
Otros actores participantes:	Ninguno	
Otros Involucrados interesados:	Ninguno	
Descripción:	Este caso de uso el actor selecciona un proyecto que desee ver. El sistema buscara en la base de datos y lo mostrara.	

Sistema de puntos de caso de uso

Autores:
Luis Iraheta, Yenifer García, Magaly Hernández

Fecha: lunes 1 de octubre de 2018
Versión: 1.0

Nombre de caso de uso:	Mostrar cálculos	Caso de usos aplicado a la información del usuario.
ID del Caso de uso:	CU-0005	
Prioridad:	Alta	
Fuente:	Requerimiento CU-0005.R1	
Actor primario de negocios:	Usuario de sistema	
Otros actores participantes:	Ninguno	
Otros Involucrados interesados:	Ninguno	
Descripción:	Mostrará todos los cálculos y tablas de UCP que selecciono anteriormente.	

Sistema de puntos de caso de uso

Autores:
Luis Iraheta, Yenifer García, Magaly Hernández

Fecha: lunes 1 de octubre de 2018
Versión: 1.0

Nombre de caso de uso:	UCP con productividad	Caso de usos aplicado a la información del usuario.
ID del Caso de uso:	CU-0006	
Prioridad:	Alta	
Fuente:	--	
Actor primario de negocios:	Usuario de sistema	
Otros actores participantes:	Ninguno	
Otros Involucrados interesados:	Ninguno	
Descripción:	En este caso uso ponemos a disposición que si un proyecto ya tiene la productividad pues solo la ingrese para luego solo calcular el factor de productividad.	

Sistema de puntos de caso de uso

Autores: Luis Iraheta, Yenifer García, Magaly Hernández

Fecha: lunes 1 de octubre de 2018

Versión: 1.0

Nombre de caso de uso:	Introducir factor de productividad	Caso de usos aplicado a la información del usuario.
ID del Caso de uso:	CU-0007	
Prioridad:	Alta	
Fuente:	--	
Actor primario de negocios:	Usuario de sistema	
Otros actores participantes:	Ninguno	
Otros Involucrados interesados:	Ninguno	
Descripción:	Este caso de uso el actor introduce el factor de productividad para luego el sistema haga el procedimiento y arroje un resultado.	

Sistema de puntos de caso de uso

Autores:
Luis Iraheta, Yenifer García, Magaly Hernández

Fecha: lunes 1 de octubre de 2018
Versión: 1.0

Nombre de caso de uso:	UCP sin productividad	Caso de usos aplicado a la información del usuario.
ID del Caso de uso:	CU-0008	
Prioridad:	Alta	
Fuente:	--	
Actor primario de negocios:	Usuario de sistema	
Otros actores participantes:	Ninguno	
Otros Involucrados interesados:	Ninguno	
Descripción:	En este caso de uso el actor crea un UPC desde cero ingresando datos para tener un resultado.	

Sistema de puntos de caso de uso

Autores:
Luis Iraheta, Yenifer García, Magaly Hernández

Fecha: lunes 1 de octubre de 2018

Versión: 1.0

Nombre de caso de uso:	Calcular UUCP	Caso de usos aplicado a la información del usuario.
ID del Caso de uso:	CU-0009	
Prioridad:	Alta	
Fuente:	--	
Actor primario de negocios:	Usuario de sistema	
Otros actores participantes:	Ninguno	
Otros Involucrados interesados:	Ninguno	
Descripción:	Este caso de uso el sistema calculara la suma de los resultados de UAW y UUCW y mostrara el resultado.	

Sistema de puntos de caso de uso

Autores: Luis Iraheta, Yenifer García, Magaly Hernández

Fecha: lunes 1 de octubre de 2018

Versión: 1.0

Nombre de caso de uso:	Calcular UAW	Caso de usos aplicado a la información del usuario.
ID del Caso de uso:	CU-0010	
Prioridad:	Alta	
Fuente:	--	
Actor primario de negocios:	Usuario de sistema	
Otros actores participantes:	Ninguno	
Otros Involucrados interesados:	Ninguno	
Descripción:	Este caso de uso el actor ingresa el número de actores que interactúan con el sistema categorizándolo en simple, medio y complejo. El sistema realizara las operaciones matemáticas para posteriormente mostrar el resulta.	

Sistema de puntos de caso de uso

Autores:
Luis Iraheta, Yenifer García, Magaly Hernández

Fecha: lunes 1 de octubre de 2018

Versión: 1.0

Nombre de caso de uso:	Calcular UUAW	Caso de usos aplicado a la información del usuario.
ID del Caso de uso:	CU-0011	
Prioridad:	Alta	
Fuente:	--	
Actor primario de negocios:	Usuario de sistema	
Otros actores participantes:	Ninguno	
Otros Involucrados interesados:	Ninguno	
Descripción:	Este caso de uso el actor ingresa el número de casos de usos categorizándolos en simple, medio y complejo el sistema realizara las respectivas operaciones y guardándolos en la base de datos.	

Sistema de puntos de caso de uso

Autores:
Luis Iraheta, Yenifer García, Magaly Hernández

Fecha: lunes 1 de octubre de 2018
Versión: 1.0

Nombre de caso de uso:	Introducir numero de actores	Caso de usos aplicado a la información del usuario.
ID del Caso de uso:	CU-0012	
Prioridad:	Alta	
Fuente:	--	
Actor primario de negocios:	Usuario de sistema	
Otros actores participantes:	Ninguno	
Otros Involucrados interesados:	Ninguno	
Descripción:	En este caso uso se ingresa el número de actores según la categorización que emplea el método.	

Sistema de puntos de caso de uso

Autores: Luis Iraheta, Yenifer García, Magaly Hernández

Fecha: lunes 1 de octubre de 2018

Versión: 1.0

Nombre de caso de uso:	Introducir actores	Caso de usos aplicado a la información del usuario.
ID del Caso de uso:	CU-0013	
Prioridad:	Alta	
Fuente:	--	
Actor primario de negocios:	Usuario de sistema	
Otros actores participantes:	Ninguno	
Otros Involucrados interesados:	Ninguno	
Descripción:	Este caso de uso el actor introduce el tipo de actor que pide el método si es simple, medio y complejo. Se guardara en la base de datos.	

Sistema de puntos de caso de uso

Autores: Luis Iraheta, Yenifer García, Magaly Hernández

Fecha: lunes 1 de octubre de 2018

Versión: 1.0

Nombre de caso de uso:	Introducir UC	Caso de usos aplicado a la información del usuario.
ID del Caso de uso:	CU-0014	
Prioridad:	Alta	
Fuente:	--	
Actor primario de negocios:	Usuario de sistema	
Otros actores participantes:	Ninguno	
Otros Involucrados interesados:	Ninguno	
Descripción:	En este caso uso se ingresa el número de casos de uso que tiene el sistema según la categorización que emplea el método.	

Sistema de puntos de caso de uso

Autores: Luis Iraheta, Yenifer García, Magaly Hernández

Fecha: lunes 1 de octubre de 2018

Versión: 1.0

Nombre de caso de uso:	Introducir tipo	Caso de usos aplicado a la información del usuario.
ID del Caso de uso:	CU-0015	
Prioridad:	Alta	
Fuente:	--	
Actor primario de negocios:	Usuario de sistema	
Otros actores participantes:	Ninguno	
Otros Involucrados interesados:	Ninguno	
Descripción:	Este caso de uso el actor introduce el tipo de caso de uso que pide el método si es simple, medio y complejo. Se guardara en la base de datos.	

Sistema de puntos de caso de uso

Autores:
Luis Iraheta, Yenifer García, Magaly Hernández

Fecha: lunes 1 de octubre de 2018

Versión: 1.0

Nombre de caso de uso:	Calcular TCF	Caso de usos aplicado a la información del usuario.
ID del Caso de uso:	CU-0016	
Prioridad:	Alta	
Fuente:	--	
Actor primario de negocios:	Usuario de sistema	
Otros actores participantes:	Ninguno	
Otros Involucrados interesados:	Ninguno	
Descripción:	Este caso de uso el actor ingresa el impacto recibido de una serie de factores técnicos que influyen en el sistema. El sistema realizara las operaciones matemáticas para posteriormente mostrar el resultado.	

Sistema de puntos de caso de uso

Autores: Luis Iraheta, Yenifer García, Magaly Hernández

Fecha: lunes 1 de octubre de 2018

Versión: 1.0

Nombre de caso de uso:	Calcular ECF	Caso de usos aplicado a la información del usuario.
ID del Caso de uso:	CU-0017	
Prioridad:	Alta	
Fuente:	--	
Actor primario de negocios:	Usuario de sistema	
Otros actores participantes:	Ninguno	
Otros Involucrados interesados:	Ninguno	
Descripción:	Este caso de uso el actor ingresa el impacto recibido de una serie de factores ambientales que influyen en el sistema. El sistema realizara las operaciones matemáticas para posteriormente mostrar el resultado	

Sistema de puntos de caso de uso

Autores:
Luis Iraheta, Yenifer García, Magaly Hernández

Fecha: lunes 1 de octubre de 2018

Versión: 1.0

Nombre de caso de uso:	Introducir impacto recibido	Caso de usos aplicado a la información del usuario.
ID del Caso de uso:	CU-0018	
Prioridad:	Alta	
Fuente:	--	
Actor primario de negocios:	Usuario de sistema	
Otros actores participantes:	Ninguno	
Otros Involucrados interesados:	Ninguno	
Descripción:	En este caso uso el actor ingresara el impacto recibido de TCF y ECF.	

3.2.5.2. Diagrama de Modelo Conceptual

Modelo Conceptual, el cual nos muestra los conceptos presentes en el dominio del problema. Un concepto para este caso, en términos de la Programación Orientada a Objetos, es un objeto del mundo real; es decir, es la representación de cosas del mundo real y NO de componentes de software. En él no se definen operaciones (o métodos); en este modelo se pueden mostrar los conceptos, los atributos de los conceptos (opcionalmente) y la relación o asociación entre ellos. Informalmente podríamos decir que un concepto es una idea, cosa u objeto. Para descubrirlos debemos analizar los sustantivos en las descripciones textuales del dominio del problema, es decir, de la descripción del sistema, de los requerimientos y de los Casos de Uso

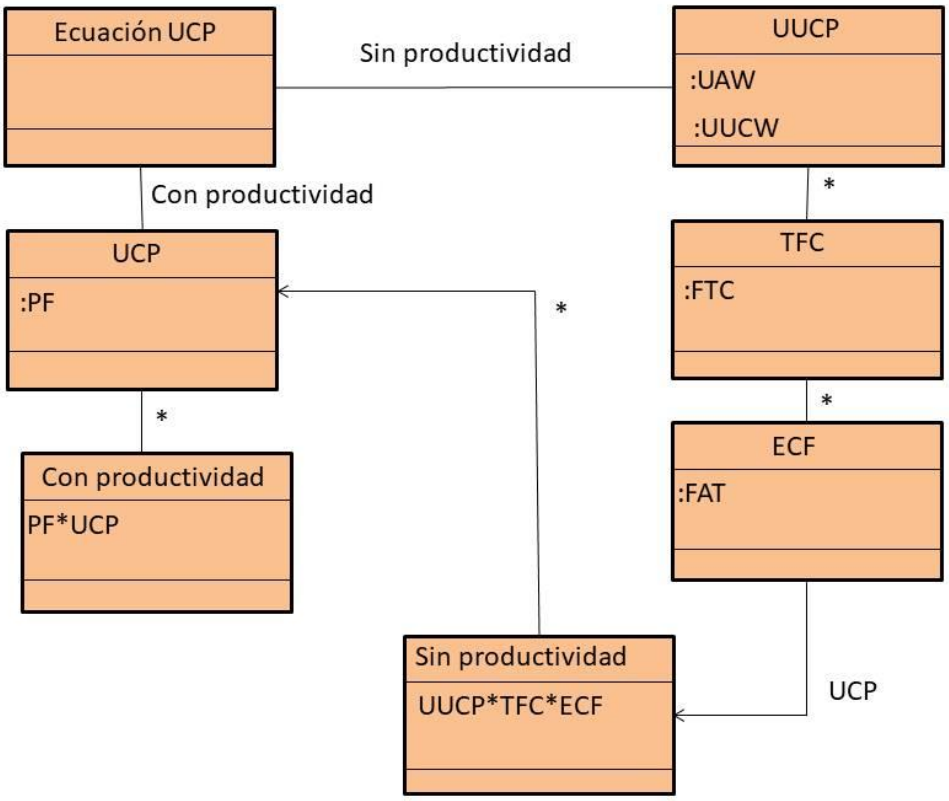
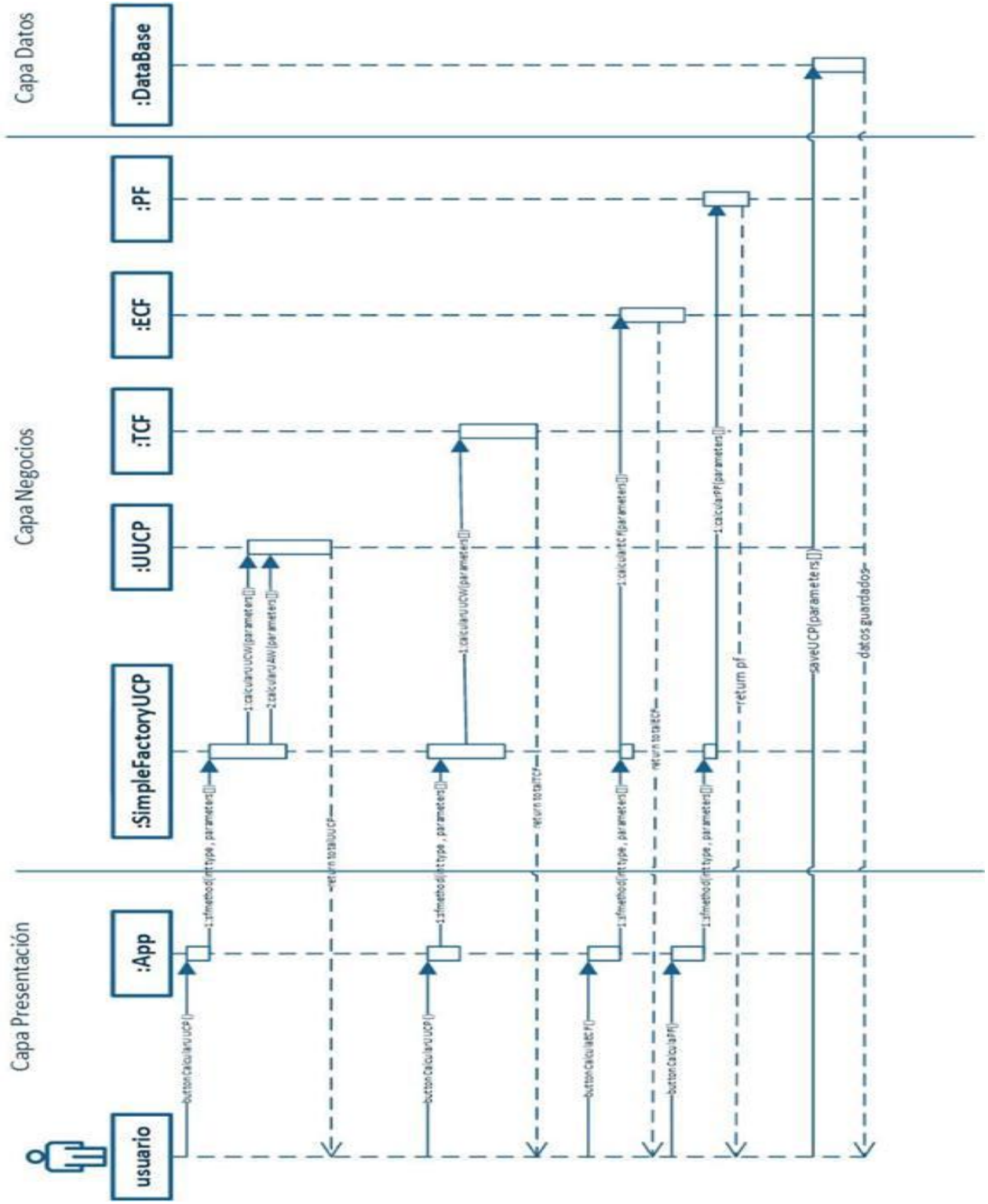
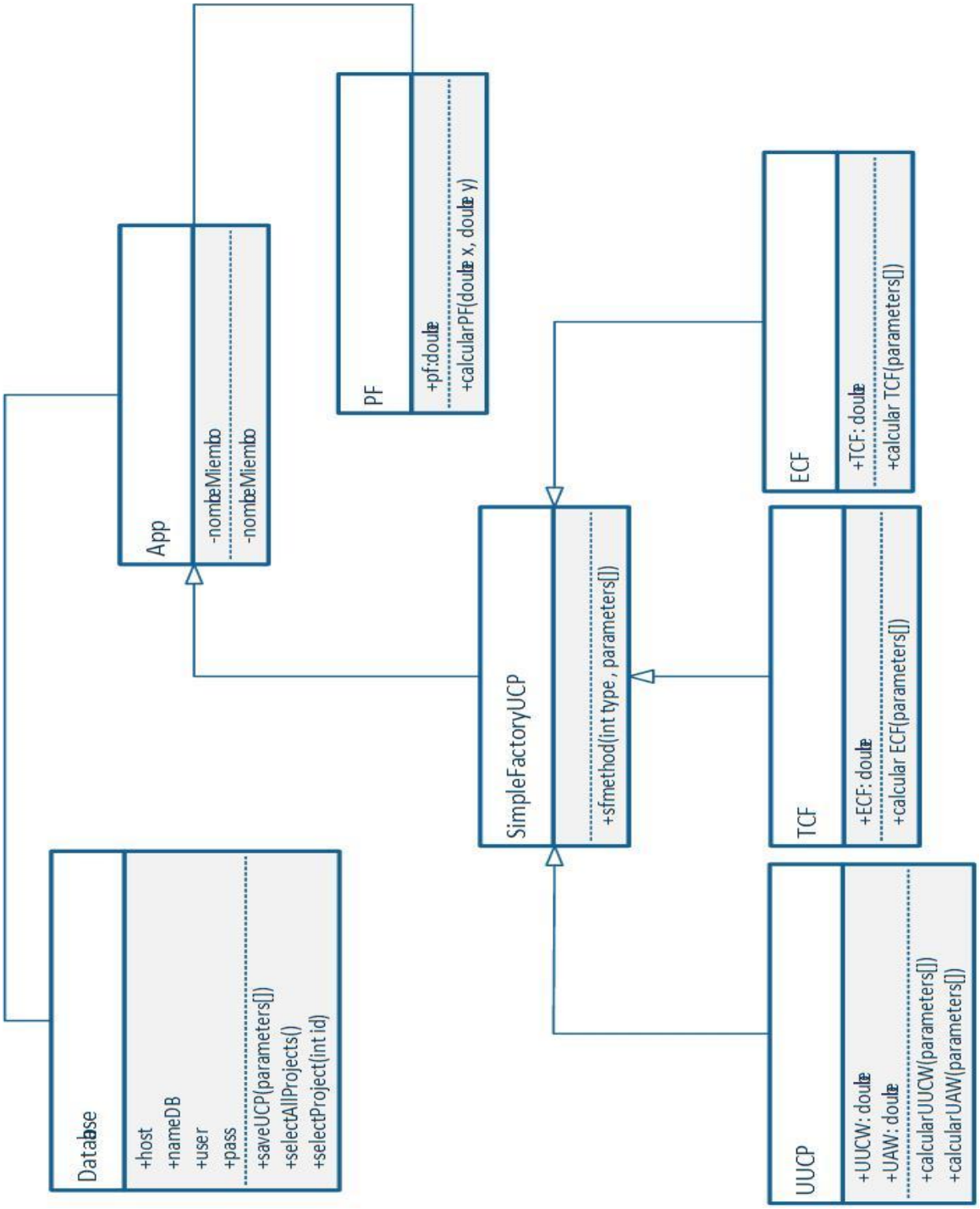


Diagrama Conceptual

3.2.5.3. Diagrama de Secuencia

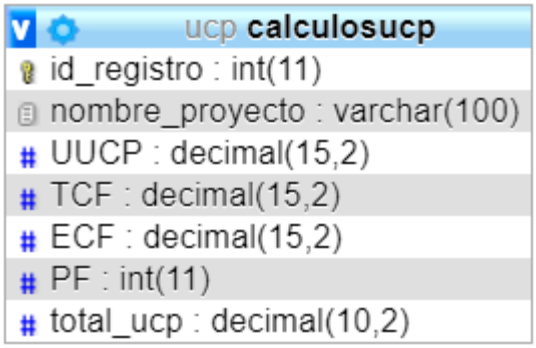


3.2.5.4. Diagrama de Clases



IV. Base de Datos

4.1. Modelo Entidad Relación



No existen ninguna relación, ya que solo se manejará una tabla para guardar información.

En el diseño de la base de datos se contempla una sola tabla la cual será la encargada de manejar los datos enviados por el usuario para que estos sean guardas, esta tabla retornada la información de los cálculos de los proyectos, para que el usuario ingrese el factor de productividad y se realice un nuevo cálculo de horas.

4.2. Diccionario de Datos

Cálculos ucp

Comentarios de la tabla: Se realizan los cálculos de para los UCP

Columna	Tipo	Nulo
id_registro *(Primaria)*	int(11)	No
UUCP	decimal(15,2)	No
TCF	decimal(15,2)	No
ECF	decimal(15,2)	No
PF	int(10)	No

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo
PRIMARY	BTREE	Sí	No	id_registro	0	A	No

V. Estudio de Viabilidades

5.1 Viabilidad Operacional

Es la menos técnica pero sí la más importante. Realizar el estudio para analizar si las necesidades del negocio pueden llegar a cumplirse a través de la idea propuesta. Además, medir en qué grado el sistema propuesto resuelve problemas y si se aprovecha de las oportunidades. Aquí se incluyen parámetros de diseño como la fiabilidad, compatibilidad del proyecto, facilidad de uso, facilidad de mantenimiento, accesibilidad y otros.

-En este caso la aplicación no se tiene la certeza de que será implementada después de desarrollada y por ende no tendrá mantenimiento de la aplicación.

-La aplicación se desarrollara en tres plataformas (consola, escritorio, aplicación móvil).

5.2 Viabilidad Economica

Permite a las empresas evaluar la viabilidad, estimando beneficios antes de asignar los recursos financieros y costes. Buenos resultados en esta área dan mucha fiabilidad al proyecto. Es un análisis generalizado de costes/beneficios.

Beneficios: la aplicación no tendrá ningún beneficio porque los costes para desarrollar la aplicación son muy altos debido a la plataforma en la que se está implementando.

Presupuesto de la aplicación se encuentra en anexos.

5.1. Puntos de Caso de Uso.

UUCW (Peso de los Casos de Uso sin Ajustar (1/2)

Categorías de los casos de uso sin ajustar

Categoría de los casos de uso	Descripción	Factor(Peso)
Simple	Transacciones=3 o menos Clases: menos de 5	5
Medio	Transacciones=4 a7 Clases:5 a 10	10
Complejo	Transacciones=más de 7 transacciones Clases: más de 10 clases	15

UAW (Pesos de los Casos de Uso sin Ajustar (2/2)

Formula: UUCW = Σ (cantidad de un tipo de caso de uso * Factor)

Categoría de los casos de uso	Descripción	Peso (Factor)	Número de Actores	Resultado
Simple	Transacciones=3 o menos Clases: menos de 5	5	0	0
Medio	Transacciones=4 a7 Clases:5 a 10	10	0	0
Complejo	Transacciones=más de 7 transacciones Clases: más de 10 clases	15	18	270
			Total UUCW	270

UAW (Peso de los actores sin Ajustar (1/2)

Evaluación de la complejidad de los actores con los que interactúa la aplicación.

Categoría de los casos de uso	Descripción	Factor(Peso)
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API).	1
Medio	Otro sistema interactuando mediante un protocolo o una persona interactuando a través de una interfaz en modo texto.	2
Complejo	Una persona que interactúa mediante una interfaz gráfica (GUI).	3

UAW (Pesos de los actores sin Ajustar (2/2)

Formula: UAW = Σ (cantidad de un tipo de Factor * Factor)

Tipo de Actor	Descripción	Peso (Factor)	Número de Actores	Resultado
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API).	1	2	2
Medio	Otro sistema interactuando mediante un protocolo o una persona interactuando a través de una interfaz en modo texto.	2	0	0
Complejo	Una persona que interactúa mediante una interfaz gráfica (GUI).Clases: más de 10 clases	3	0	0
			Total UAW	2

Calculo de los puntos de caso de uso sin ajustar

UUCP= UUCW + UAW

UUCP= 270 + 2

UUCP= 272

2. Factor de Complejidad Técnica (2/5)

Compuesto por 13 puntos que evalúan la complejidad de los módulos del sistema que se desarrolla.

Factor Técnico	Descripción	Peso Dado
T1	Sistema distribuido	2
T2	Rendimiento o tiempo de respuesta	1
T3	Eficiencia del usuario final	1
T4	Procesamiento interno complejo	1
T5	El código debe de ser reutilizable	1
T6	Facilidad de instalación	0.5
T7	Facilidad de uso	0.5
T8	Portabilidad	2
T9	Facilidad de cambio	1
T10	Concurrencia	1
T11	Características especiales de seguridad	1
T12	Provee acceso directo a terceras partes	1
T13	Se requiere facilidades especiales de entrenamiento a usuario	1

Factor de Complejidad Técnica (3/5)

Cada uno de los factores se debe evaluar según la siguiente escala:

Descripción	Valor
Irrelevante	0 a 2
Medio	3 a 4
Esencial	5

TCF (Factor de Complejidad Técnica (4/5)

Factor Técnico	Descripción	Peso	Impacto percibido	Factor calculado
T1	Sistema distribuido	2	3`	6
T2	Rendimiento o tiempo de respuesta	1	4	4
T3	Eficiencia del usuario final	1	5	5
T4	Procesamiento interno complejo	1	4	4
T5	El código debe de ser reutilizable	1	3	3
T6	Facilidad de instalación	0.5	3	1.5
T7	Facilidad de uso	0.5	4	2
T8	Portabilidad	2	0	0
T9	Facilidad de cambio	1	1	2
T10	Concurrencia	1	3	3
T11	Características especiales de seguridad	1	2	2
T12	Provee acceso directo a terceras partes	1	1	1
T13	Se requiere facilidades especiales de entrenamiento a usuario	1	2	2
		Factor Total Técnico		34.5

TCF=0.6 + (0.01*factor total técnico)

TCF=0.6+ (0.01*34.5)

TCF=0.945

3. ECF (Factor de Complejidad Ambiental) (2/4)

Factor Ambiental	Descripción	Peso
E1	Familiaridad con el modelo del proyecto utilizado Familiaridad con UML	1.5
E2	Personal tiempo parcial	-1
E3	Capacidad del analista líder	0.5
E4	Experiencia en la aplicación	0.5
E5	Experiencia en orientada a objetos	1
E6	Motivación	1
E7	Dificultad del lenguaje de programación	-1
E8	Estabilidad de los requerimientos	2

ECF (Factor de Complejidad Ambiental) (3/4)

Factor Ambiental	Descripción	Peso	Impacto percibido	Factor calculado
E1	Familiaridad con el modelo del proyecto utilizado Familiaridad con UML	1.5	4	6
E2	Personal tiempo parcial	-1	3	-3
E3	Capacidad del analista líder	0.5	5	2.5
E4	Experiencia en la aplicación	0.5	4	2
E5	Experiencia en orientada a objetos	1	5	5
E6	Motivación	1	5	5
E7	Dificultad del lenguaje de programación	-1	4	-4
E8	Estabilidad de los requerimientos	2	4	8
		Factor Ambiental Total		21.5

ECF= 1.4+ (-0.3*factor ambiental total)

ECF=1.4+ (-0.03*21.5)

ECF=0.755

Calculando los UCP

UCP = UUCP * TCF * ECF

Los valores obtenidos

UUCP= 272

TCF= 0.945

ECF= 0.755

UCP= 272 * 0.945 * 0.755

UCP= 194.07

Agregando la productividad

Factor de productividad

Total, de horas estimadas= UCP * PF

Total, de horas estimadas= 194.07 * 20

Total, de horas estimadas= 3881.304

Para el costo del desarrollo del proyecto se utiliza una tarifa de pago al programador $25 dólares por hora.

Total en mano de obra para el desarrollo = 3881.304 h * $25 / h = $97,032.60

5.2. Viabilidad Técnica.

Permite evaluar si los equipos, sistemas y software están disponibles y tienen las capacidades técnicas necesarias para cada propuesta de diseño planificado. También se analiza el factor humano, es decir, si el personal cuenta con la experiencia y conocimientos técnicos requeridos para el sistema que se propone. Es un análisis de los recursos técnicos disponibles en la organización.

Hardware

Para el desarrollo de los proyectos el cual van dirigido en el tema de los puntos de casos de uso se utilizan el dispositivo móvil para ejecutar la aplicación, posteriormente para ejecutar el programa en consola y en formulario se necesita una computadora sea de escritorio o portátil que cuente con requerimientos básicos.

Software

Se cuenta con la tecnología de software necesaria para desarrollar la aplicación como lo son el entorno de desarrollo que se especificó anteriormente React Native ya que es gratuita y se tiene el conocimiento para desarrollar en ella.

También para el almacenamiento de los datos que se utiliza un Gestor de Base de Datos SQLite. Y un entorno de desarrollo instalado en este caso Netbeans.

Recurso humano

- Analista de sistemas.
- Diseñador de sistemas.
- Programador.

Analista de sistemas

Descripción
Los analistas de sistemas informáticos adaptan y diseñan sistemas de información para ayudar a las empresas trabajar de forma más rápida y eficiente. Trabajan en estrecha colaboración con personal de todas las categorías para averiguar los problemas que surjan en el sistema existente, y para cumplir con las expectativas del cliente a la hora de crear un nuevo sistema. Los analistas producen una especificación para un sistema que satisfaga las necesidades de la empresa.

Actividades laborales
Los analistas de sistemas utilizan tecnologías de la información y comunicación (TIC) para ayudar a las empresas a trabajar de forma más rápida y eficiente. Investigan un problema y luego diseñan o adaptan un sistema informático para mejorar el funcionamiento de la empresa.

Una vez la empresa ha aprobado el sistema, el analista comienza a trabajar en estrecha colaboración con los especialistas de TIC, diseñadores de sistemas y programadores para crear el sistema.

Cuando el proyecto está terminado, los analistas examinan cuidadosamente el nuevo sistema para asegurarse de que cumple con sus funciones y de que los usuarios están satisfechos con este.

Perfil profesional
Para ser analista de sistemas informáticos se necesita:
- Disfrutar a la hora de plantearse retos para la solución de problemas y de sopesar los pros y los contras de las diferentes soluciones.
- Capacidades lógicas, analíticas y de investigación, así como habilidades creativas.
- Conocimientos de informática y técnicas de programación.
- Fuertes habilidades de comunicación verbal y escrita.
- Saber escuchar y tener la capacidad de hacer las preguntas correctas.

Competencias
Analiza necesidades en software.
Capacidad de análisis.
Capaz de plantear preguntas con claridad.
Capaz de trabajar bajo presión.
Capaz de trabajar con vencimientos.
Conocimientos especializados en informática.
Creativo.
Destrezas en informática.
Diseña y adapta sistemas informáticos.
Habilidad para resolver problemas.
Planifica y realiza la introducción de nuevos sistemas.
Trabaja en equipo.

Diseñador de sistemas

Descripción:

El diseñador identifica y define las responsabilidades, operaciones, atributos y relaciones de los elementos de diseño. El diseñador se asegura de que el diseño sea coherente con la arquitectura de software, y que esté detallado hasta un punto en que pueda proceder la implementación.

Habilidades:

El diseñador debe tener conocimientos laborales sólidos de:

- Requisitos del sistema
- La arquitectura del sistema
- Técnicas de diseño de software, incluyendo técnicas de análisis y diseño orientado a objetos, y el Lenguaje unificado de modelado
- Tecnologías con las que se implementará el sistema
- Directrices de proyecto sobre cómo se relaciona el diseño con la implementación incluyendo el nivel de detalle esperado en el diseño antes de que proceda la implementación.
- Propuestas de asignación

A un diseñador se le puede asignar la responsabilidad de implementar una parte estructural del sistema (como un subsistema de implementación o de clases), o una parte funcional del sistema, como la ejecución de guiones de uso o sus características que cruza clases/subsistemas.

Programador

Descripción

Los programadores informáticos escriben y prueban los programas de ordenador. Escriben las instrucciones en un lenguaje informático que el ordenador puede leer, para llevar a cabo tareas tales como el control de stock en un almacén o de registro de ventas. Los programadores desarrollan los pasos lógicos necesarios para crear un programa, lo prueban y almacenan los registros de forma segura, para poder adaptar los programas en el momento que se necesite.

Actividades laborales

Los programadores informáticos escriben programas computacionales que dan instrucciones a un ordenador para que realice las tareas necesarias para almacenar la información introducida por los usuarios. Este trabajo permite, por ejemplo, controlar las acciones de la empresa, hacer cálculos salariales o mantener registros de personal.

Perfil profesional:

Para ser programador informático hay que tener las características siguientes:

-Tener conocimientos de programación.

-Ser analítico y lógico en el enfoque para la solución de problemas.

-Prestar atención a los detalles.

-Tener habilidades comunicativas y de trabajo en equipo.

-Concentrarse durante largos períodos de tiempo.

-Contar con habilidades de comunicación escrita para la compilación de informes y la elaboración de manuales.

-Administrar el tiempo de forma eficiente, priorizar tareas y trabajar bajo la presión de cumplir plazos determinados.

-Mantener registros exactos del trabajo realizado.

-Estar siempre al día sobre la evolución de los lenguajes de software y de programación, así como de las nuevas herramientas informáticas.

-Según el lugar de trabajo, se pueden requerir habilidades de negociación.

Competencias

-Adapta programas existentes.

-Analiza necesidades en software.

-Aptitudes para gestionar el tiempo.

-Aptitudes para llevar registros.

-Capacidad de análisis.

-Capacidad para concentrarse.

-Capacidad para priorizar tareas.

-Capacidad para trabajar en equipo.

-Capaz de mantenerse al día de los avances tecnológicos.

-Capaz de prestar atención al detalle.

-Capaz de trabajar bajo presión.

-Capaz de trabajar con vencimientos.

-Corrige defectos de software.

-Destrezas en informática.

-Escribe y desarrolla programas informáticos.

-Habilidad para la programación.

-Habilidad para resolver problemas.

-Trabaja en equipo.

-Utiliza códigos, herramientas y lenguajes de programación.

5.3. Viabilidad Ambiental.

En todo proyecto se evalúan los aspectos tanto sociales, económicos, culturales, legales y ambientales, tales que estos sean favorables y apunten a un buen desarrollo. Cada uno de estos aspectos debe ejecutarse de manera correcta a medida se obtengan resultados prometedores sin complicar o traer riesgos al proyecto o a la sociedad.

En el aspecto económico este proyecto es viable ya que contara con facilidades de tiempo y dinero debido a su agilidad en los procesos, también mencionar que es amigable con el ambiente ya que no se desechan materiales como papeles o tintas, de esta manera se mantiene un ambiente ecológico sin producir daños y mejorando el desarrollo.

Cabe destacar la importancia de la tecnología en el uso cotidiano y más, en el ambiente de trabajo donde se desechan o producen residuos que se acumulan con el paso del tiempo; mas con el uso de un dispositivo móvil o una computadora el uso de materiales físicos es cada día menos necesario.

VI. Conclusiones

Lo expuesto en este trabajo indica que la aplicación del método de Puntos de Caso de Uso propuesto por Frohnhoff y Engels, sobre doce casos de estudio reales, genera desviaciones excesivas en la estimación de esfuerzo de un proyecto de software. La clasificación cuantitativa y cualitativa otorgada por Frohnhoff y Engels a los casos de uso, no es parámetro convincente a la hora realizar una estimación de esfuerzo en una etapa preliminar del proyecto.

Frohnhoff y Engels enfocaron su mejora en los aspectos tecnológicos y de entorno. Los casos de uso son una herramienta útil e importante en el desarrollo de proyecto de software. Representan el punto de partida para la diagramación de un sistema orientado objetos, los puntos bases de la planificación del proyecto, y la documentación de entrada en la validación de requerimientos y en etapas de testeo.

Habiendo analizado todo esto, la necesidad de un software para todos estos procedimientos es de mucha importancia, por ello se realizarán tres programas en diferentes plataformas ya requerido por el cliente, con esto se pretende ahorrar tiempo para los usuarios.

VII. Recomendaciones

Para el desarrollo del proyecto se tomó en cuenta el primer principio SOLID, con el objetivo de tener un buen diseño de programación, de tal manera que el mantenimiento del software o añadir nuevas características no tenga que incurrir a modificar grandes porciones de código, sino que este pueda ser reutilizado.

S-Responsabilidad simple (Single responsibility)

Este principio trata de destinar cada clase a una finalidad sencilla y concreta. En muchas ocasiones estamos tentados a poner un método reutilizable que no tienen nada que ver con la clase simplemente porque lo utiliza y nos pilla más a mano. En ese momento pensamos "Ya que estamos aquí, para que voy a crear una clase para realizar esto. Directamente lo pongo aquí".

El problema surge cuando tenemos la necesidad de utilizar ese mismo método desde otra clase. Si no se refactoriza en ese momento y se crea una clase destinada para la finalidad del método, nos toparemos a largo plazo con que las clases realizan tareas que no deberían ser de su responsabilidad.

Con la anterior mentalidad nos encontraremos, por ejemplo, con un algoritmo de formateo de números en una clase destinada a leer de la base de datos porque fue el primer sitio donde se empezó a utilizar. Esto conlleva a tener métodos difíciles de detectar y encontrar de manera que el código hay que tenerlo memorizado en la cabeza.

VIII. Anexos

Referencias bibliografías

E. KENDALL, K. y. (2005). *Análisis y diseño de sistemas. Sexta edición.* México: PEARSON EDUCACIÓN.

Loira, R. d. (2005). *Ingenieria del Software*. Madrid: PEARSON EDUCATION SA.

Presuman, R. S. (2010). *INGENIERÍA DEL SOFTWARE. UN ENFOQUE PRÁCTICO.* México: McGraw-Hill.

Desing Patterns, Fecha de publicación original: 21 de octubre de 1994, *Erich Gamma, Richard Helm, John Vlissides, Ralph Johnson*, Editorial: *Addison-Wesley*

Presupuesto

A continuación, se detalla el presupuesto para el desarrollo del proyecto, los costos son financiados por cada uno de los estudiantes involucrados.

Descripción	Consola	Escritorio	Aplicación móvil	Precio
Entorno de desarrollo	NetBeans	NetBeans	Visual Studio Code	$0.00
Gestor de base de datos	SQLite			$0.00

Cronograma

Actividades por día desde viernes 22 de septiembre de 2018 a viernes 5 de octubre de 2018.

Actividades	1-3	4-9	10-15	16
Análisis de requisitos.				
Diseño del sistema.				
Codificación				
Pruebas				