MEMORIA TÉCNICA DE ESTADÍA

DESARROLLO DE UN SISTEMA DE CONTROL DE ALMACÉN PARA ESCRITORIO Y ANDROID

PRESENTADO POR:
MIGUEL ÁNGEL VITE HERNÁNDEZ

PARA OBTENER EL TÍTULO DE:
INGENIERO EN SOFTWARE

ASESORADO POR:
«ASESOR»

ZEMPOALA, HIDALGO. MES-AÑO

0.0.

Agradecimientos

En toda mi vida académica estuvieron conmigo muchas personas con quienes me fui familiarizando, hasta poder llamarlos mis amigos. Con ellos compartí mucho momentos que recordaré por mucho tiempo y muchas enseñanzas que me han ayudado a tomar las mejores decisiones en mi vida. Y no solo mi amigos, sino también mis padres, hermanas, tíos y abuelos. Y quiero dedicarle a todos ellos este trabajo en el que puse todo mi esfuerzo y dedicación.

A mi madre, quien siempre estuvo ahí apoyándome incondicionalmente en todo lo que me hiciera falta, aconsejándome con sus sabias palabras, motivándome a seguir adelante con todo lo que me proponga y que siempre me brindó su cariño en todo momento, y que no solo es una madre sino una gran amiga.

A mi padre, quien siempre vio por mi y cuidó de que nada me hiciera falta en todo momento, quien siempre estuvo motivándome y aconsejándome con sus sabiduría, comprensión y su apoyo incondicional, que gracias a él, este proyecto es posible. Y no solo es un padre, sino un gran amigo.

A mis hermanas, quienes siempre estuvieron junto a mi, me alegraban y apoyaban en todo momento, que a pesar de mis frustraciones siempre me daban un motivo por seguir adelante.

A mis tíos, que siempre creyeron en mi.

A mis abuelos que siempre me aconsejaban y me llenaban de paz, tranquilidad y amor.

A mis amigos Daniel, Juan Antonio, David, Monse, Tania, Raúl, Misael, Elias, Ramiro, Said, Severa, Yuli, Rubi y Juan David, con quienes compartí de mis mejores en mi vida académica y quienes que me apoyaron y motivaron a seguir adelante.

A mis profesores, por llenarme de conocimiento y aprendizaje que siempre tendré en cuenta.

A mi asesor, que sin el, este trabajo no sería posible, y quien me fue guiando en todo momento.

0.0.

Resumen

0.0.

Índice general

Índice de figuras

0.0. ÍNDICE DE FIGURAS

Índice de tablas

Capítulo 1

Introducción

Grupo Impresor Criterio S.A. de C.V. es una empresa de con sede en la ciudad de Pachuca de Soto, Hidalgo. Su giro principal es el servicio de comunicación y publicación de información político-económico, así como de interés general por medio de periódico, revistas y redes sociales. Dentro de la misma existen distintas áreas como fotografía, deportes, publicidad, bodega, sistemas, contabilidad, recursos humanos, web, contabilidad.

Actualmente la empresa trabaja con SAI ERP, llevando a cabo las funciones de facturación, compras, gastos, cuentas por pagar, proveedores, nómina y almacén. Este sistema se ha estado utilizando desde 4 años. Durante este periodo ha presentado ciertas inconsistencias en el área de almacén, ya que para subir al sistema se tienen que llenar formatos a mano para luego ser subidas al sistema. Este proceso desperdicia mucho tiempo y puede ser optimizado.

1.1. Antecedentes

Actualmente existen una gran cantidad de ERP's que son capaces de adaptarse a las necesidades de cualquier organización, además de mejorar el desempeño de las actividades que se realizan dentro de una organización.

1.2. Problemática

Grupo Impresor Criterio S.A. de C.V. es una empresa con el giro de servicios, la cual hace uso de SAI ERP como sistema principal, concentrando las principales actividades de facturación, compras, gastos, cuentas por pagar, proveedores, nómina y almacén, además de la generación de reportes. En el área de almacén se encargan principalmente de recibir todo el material que la empresa utilizara, además de cambiarlo de almacén según requieran el material. Existen ciertos inconvenientes cuando se trata de subir los datos al sistema, ya que cuando llegan revistas o periódicos, se tiene que llenar un formato a mano, para posteriormente subirlo al sistema. Esta actividad puede llevar menos tiempo del que normalmente necesita.

El alcance del presente trabajo es analizar SAI ERP y desarrollar una solución viable, mediante una actualización del sistema o desarrollando un sistema nuevo. Se trabajará bajo la metodología XP (eXtreme Programming) en los lenguajes de C# para su versión de escritorio y Java para su versión móvil y para el web service, y se usarán las normas IEEE 830 para los requisitos y UML para los diagramas de casos de uso, diagramas de componentes y diagramas de clases.

1.3. Objetivo general

Desarrollar un sistema de control de almacén en los lenguajes de C# para escritorio y Java para la versión móvil, asi como el Web Service, con el fin de mejorar y agilizar las distintas actividades que se realizan en dicho almacén, automatizar la generación de reportes, agilizar la cobranza y mantener la integridad de los datos almacenados.

1.4. Objetivos específicos

- Conocer distintos ERP similares.
- Definir los requerimientos de software.
- Diseñar la base de datos.
- Diseñar la interfaz del sistema.

1.5. Justificación

SAI ERP es un sistema robusto que cuenta con una gran cantidad de herramientas, y con una interfaz que puede resultar difícil de comprender para el usuario, además de que se puede reducir el tiempo en el que se realizan ciertas tareas como cobranza, captura de datos a través de un dispositivo móvil.

Capítulo 2

Estado del arte

2.1. ERP

Actualmente existen distintos sistemas de información para cada organización de acuerdo a las funciones que desarrollan y a sus necesidades. Los sistemas de planificación de recursos empresariales (ERP por sus siglas en inglés) es un software que permite gestionar los planes de negocio de una organización en una forma integrada Chiesa (2004). Este tipo de sistemas cuentan con una grán cantidad de módulos. Los más comunes son:

- Recursos humanos.
- Ventas.
- Contabilidad y finanzas.
- Compras.
- Producción.

Estos módulos brindan información cruzada e integrada de todos los procesos de negocio. Este tipo de software debe de estar adaptado para responder a las necesidades especificas de cada organización. Este software permite administrar la información de todas la áreas, simular distintos escenarios y obtener información consolidada en tiempo real.

2.2. ERP actuales

Tabla 2.1: Comparación de los principales ERP del mercado.

Nombre	Características	Beneficios	Gratuito	Precio	Nube	Gestión móvil
Sage 200cloud	▪ Control de finanzas. ▪ Almacenes. ▪ Posventa. ▪ Gestión comercial. ▪ CRM. ▪ Bussiness intelligence. ▪ Gestión de producción. ▪ Soluciones conectadas. ▪ Gestión de proyectos. ▪ Gestión de empleados. ▪ Multilenguaje.	▪ Aumento de productividad. ▪ Fiscalidad. ▪ Rentabilidad. ▪ Integración con Office 360. ▪ Oficina en movilidád. ▪ Conexión y productividad. ▪ Actualización.	No	▪ $215 USD estándar ▪ $430 USD extra	Si	Si
ECount INC	▪ Basado en web. ▪ Gestión de servidores. ▪ Gestión de seguridad. ▪ Rastreo de datos. ▪ Mensajero corporativo. ▪ Multilenguaje.	▪ Precio asequible ▪ Informes personalizables. ▪ Respaldo de datos. ▪ Actualizaciónes gratuitas	No	▪ $55 USD / mes ▪ $600 USD / año	Si	Si
FinancialForce					Si	Si
Oracle	▪ Planificación. ▪ Análisis financiero. ▪ Gestión empresarial. ▪ Gestión de riesgos. ▪ Obligaciónes de riesgos. ▪ Compras. ▪ Inventario. ▪ Cadenas de suministros. ▪ Cierre financiero. ▪ Planificación de proyectos. ▪ Ciclo de vida del producto.	▪ Agilizar procesos de negocio. ▪ Proporciona múltiples herramientas. ▪ Soporta múltiples libros contables.	No	Cotizar con la empresa.	Si	
Infor	▪ Sectores de servicios. ▪ Gestión de compras y contratos. ▪ Gestión de activos y de efectivo. ▪ Contabilidad de proyectos.	▪ Instalaciones rápidas. ▪ 80 % de reducción del tiempo de ejecución. ▪ Asesoramiento.	No	Cotizar con la empresa.	Si	No
Odoo	▪ Gestión de proyectos. ▪ Ventas. ▪ Gestión de contactos. ▪ Eventos. ▪ Chat. ▪ Apartado web.	▪ Amplia cantidad de herramientas. ▪ Interfaz intuitiva. ▪ Integración de Odoo Studio.	No	Cotizar con la empresa.	Si	Si
Syspro	▪ Transferencia electrónica de fondos. ▪ Gestión de inventario. ▪ Gestión de informes. ▪ Ad-hoc para facilitar navegación.	▪ Variedad de funciones. ▪ GUI facil de usar. ▪ Compatibilidad con SQL.	No	Cotizar con la empresa.	Si	No
Epicor	▪ Gestión de la relación con el cliete. ▪ Gestion de contenidos empresariales. ▪ Administración de preoyectos. ▪ Analitica e inteligencia de negocios. ▪ Administración de la produccion. ▪ Planeación y producción.	▪ Expansible. ▪ Simplifica el negocio.	No		Si	Si
Netsuite	▪ Compras. ▪ Gestión financiera. ▪ Gestión de pedidos. ▪ Producción. ▪ Almacenes. ▪ Gestión de la cadena de suministro.	▪ Agilización de los procesos. ▪ Inteligencia empresarial integrada. ▪ Se añaden nuevas funciones.	No		Si	No
SAP ERP	▪ Finanzas. ▪ Logistica. ▪ Recursos humanos.	▪ Consulta de información en tiempo real. ▪ Capacidad de integrarse con otros componentes.	No	Cotizar con la empresa.	Si	Si
Acumatica ERP						
Microsoft Dynamics						
Openbravo						
Dolibarr						
SAI ERP						
Datisa ERP						

Capítulo 3

Marco contextual

Los sistemas de información han sido de gran utilidad en estos últimos años, obteniendo ciertos beneficios como automatizar el manejo de los datos, generación automática de reportes y desarrollar sus actividades de forma más eficiente y productiva. Cada organización cuenta con algún sistema de información de acuerdo a sus necesidades y actividades.

No obstante, existe ciertos problemas dentro de un sistema de información, ya sea por su grado de dificultad, problemas de diseño, consistencia e integridad de los datos almacenados.

3.1. Ingeniería de software

Sommerville (2005) define a la ingeniería de software como una disciplina de ingeniería que se interesa por todos los aspectos de la producción de software, desde las primeras etapas de la especificación de sistema hasta el mantenimiento del mismo después de que se pone en operación. Sommerville (2005) destaca dos frases clave:

1. **Disciplina de ingeniería:** Los ingenieros aplican técnicas y métodos de ingeniería para hacer que las cosas funciones, aun cuando no existen teorías y métodos aplicables, además de trabajar bajo ciertas restricciones organizacionales y económicas.

2. **Todos los aspectos de la producción de software:** La ingeniería de software no solo incluye los procesos de desarrollo de software, si no que también comprende la administración del proyecto de software y el uso de herramientas y teorías que puedan ser de utilidad en la producción de software.

El enfoque sistemático que comúnmente se usa en la ingeniería de software se conoce comúnmente como proceso de software. Un proceso de software es una secuencia de actividades que conducen a la elaboración de un producto de software. (Sommerville, 2005) menciona cuatro actividades fundamentales de este proceso:

1. **Especificación del software:** Los clientes y los programadores llegan a un acuerdo sobre qué software se producirá y bajo qué restricciones.
2. **Desarrollo del software:** Se diseña y programa el software.
3. **Validación del software:** Se verifica si es lo que acord´desarrollar y que funcione correctamente.
4. **Evolución del software:** Se modifica el software para ajustarse a los requerimientos del cliente y del mercado.

3.2. Programación Extrema (eXtreme Programming)

3.2.1. Agilidád

Para comprender de una mejor manera este concepto, es necesario conocer el concepto de *agilidad* en el ámbito de la ingeniería en software. Jacobson et al. (2000) tiene la siguiente definición:

> Un equipo ágil es diestro y capaz de responder de manera apropiada a los cambios. El cambio es de lo que trata el software en gran medida. Hay cambios en el software que se construye, en los miembros del equipo, debidos a las nuevas tecnologías, de todas clases y que tienen un efecto en el producto que se elabora o en el proyecto que lo crea. Deben introducirse apoyos para el cambio en todo lo que se haga en el software; en ocasiones se hace porque es el alma y corazón de éste. Un equipo ágil reconoce que el software es desarrollado por individuos que trabajan en equipo, y que su capacidad, su habilidad para colaborar, es el fundamento para el éxito del proyecto.

Esto nos dice que el ingeniero de software debe de ir rápido si quiere ir adaptándose a los cambios. La agilidad también incluye actitudes y estructuras de equipo que hacen más fácil la comunicación entre los ingenieros y los gerentes.

La agilidad puede aplicarse casi para cualquier proceso de software, para lograr esto se tiene que diseñar de tal forma que se le permita al equipo adaptar las tareas y desarrollarlas con fluidez, concentrarse en los productos de trabajo esenciales y hacer énfasis en una estrategia de entrega incremental que haga trabajar al software tan rápido como sea posible para el cliente, según el tipo de producto y el ambiente de operación.

3.2.2. Programación Extrema

La Programación Extrema (XP) es una metodología ágil definida por Kent Beck. Según Canós and Letelier (2012) ésta metodología está centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo

de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

3.2.3. Proceso XP

- **Planeación:** Esta actividad comienza escuchando a los demás integrantes del equipo con el fin de recabar los requerimientos de software. Esto conlleva a la creación de *historias de usuario* que describen las salidas, caracterizaras y funciones del software, el cliente les asigna un valor o prioridad. Posteriormente es analizada por los demás integrantes del equipo y se le asigna un costo medido en semanas de desarrollo.

- **Diseño:**

- **Codificación:**

- **Pruebas:**

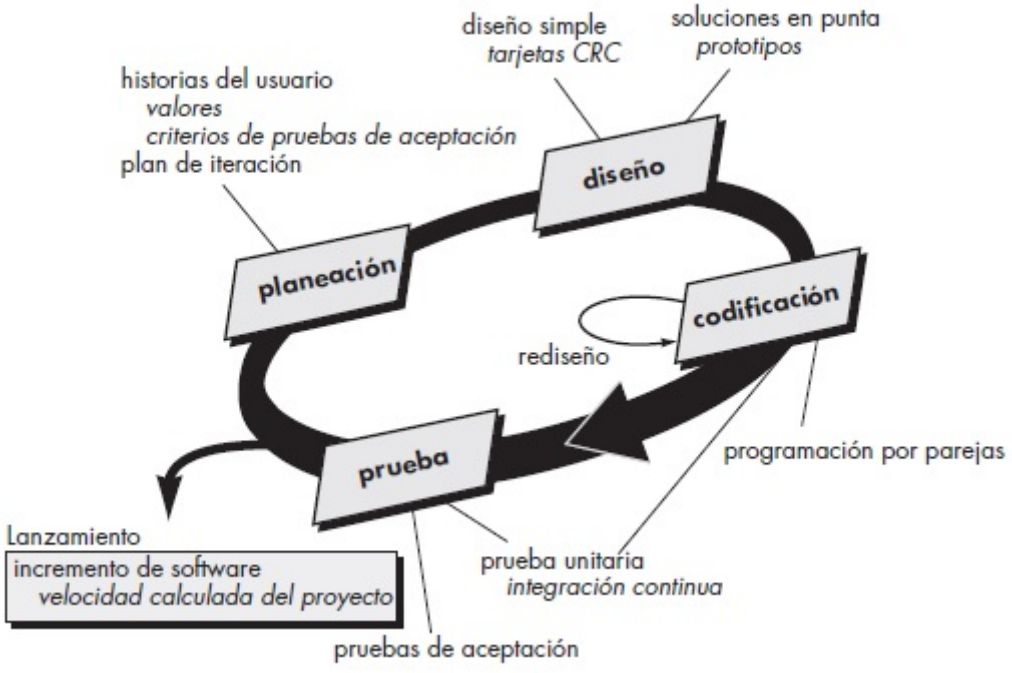

Figura 3.1: Proceso de la Programación Extrema (Pressman (2005)).

3.2.4. Historias de usuario

Es una técnica utilizada para la determinación de los requisitos de software. Son tarjetas de papel, en donde el cliente describe de forma breve y clara las características que el software debe contener, ya sean requisitos funcionales o no funcionales en un lenguaje no técnico (Canós and Letelier (2012)).

Tabla 3.1: Ejemplo de una historia de usuario.

1. Aplicación didáctica		
Como usuario quiero que la aplicación sea didáctica y de aprendizaje mediante ejercicios.		
Estimación: 7 semanas	**Valor:** 50	**Dependencias:**
Condiciones de satisfacción: ▪ La aplicación contendrá información selecta y bien explicada. ▪ Se incluirán algunos ejercicios los cuales deberán resolverse en la computadora.		

La esctructura de una historia de usuario consiste en los siguientes elementos:

- **Número de historia:** El numero que corresponde a la historia.
- **Nombre de la historia:** Debe ser breve y claro.
- **Descripción:** Es una explicación breve en lenguaje no técnico de la función que va a realizar el sistema.
- **Estimación:** El tiempo que llevará desarrollar dicha función.
- **Valor:** El numero de importancia que se le asigna a la historia.
- **Dependencias:** Indica la historias de usuario que dependen de esta misma.
- **Condiciones de satisfacción:** Son restricciones bajo las cuales se desarrollará la historia de usuario.

3.2.5. Roles en XP

Los roles que Beck (1999) propone son:

- **Programador:** Es el encargado de producir el código del sistema y llevar a cabo las pruebas unitarias.
- **Cliente:** Es el que escribe las historias de usuario y los requerimientos funcionales para su implementación, asigna la prioridad a las historias de usuario y decide cuales se implementaran en cada iteracion.

- **Encargado de pruebas (*tester*):** Es el encargado de ejecutar las pruebas y ayuda al cliente a escribir las pruebas funcionales.

- **Encargado de seguimiento:** Es quien verifica el grado de acierto en las estimaciones realizadas con el tiempo real que se utilizó, dando seguimiento para mejorar futuras estimaciones.

- **Entrenador (*coach*):** Es el responsable de todo e proceso, guiando al equipo con el fin de que se aplique el proceso de XP de forma correcta y eficiente.

- **Consultor:** Es un miembro externo al equipo con un conocimiento de un tema en especifico en el que pueda surgir algún problema.

- **Gestor (*Big boss*):** Es el enlace entre el cliente y los programadores, coordinando al equipo a que trabajen con las condiciones adecuadas de forma efectiva.

3.3. Lenguaje de Modelado Unificado (UML)

El Lenguaje de Modelado Unificado (UML por sus siglas en inglés) es un lenguaje de modelado no un proceso o un método. UML está compuesto por una notación muy especificas y reglas semánticas relacionadas para la construcción de un sistema de software. UML tiene un amplio conjunto de elementos de notación gráfica. Estos pueden ser clases, componentes, nodos, actividades, flujos de trabajo, casos de uso, objetos y estados. Además de como relacionar estos elementos entre si (Sparks (2000)).

3.4. Notación

3.4.1. Vistas

Las vistas muestran los diferentes aspectos del sistema que está siendo modelado. No es un gráfico, sino una abstracción que consiste en una serie de diagramas. Cada vista mostrará aspectos particulares del sistema, dando enfoque a ciertos ángulos y niveles, y con esto podría ser construida una imagen completa del sistema (Burgués (2014)).

3.4.2. Modelos

Los modelos representan una abstracción de una entidad en la que se esté trabajando (Burgués (2014)).

3.5. Elementos estructurales

Éstos son elementos estáticos y representan comúnmente cosas conceptuales o materiales. Alarcón (2000) describe los siguientes.

3.5.1. Clases

Una clase es una descripción de un conjunto de objetos que comparten los mismos atributos, relaciones, operaciones y semántica. Una clase puede implementar una o mas interfaces. Está representada como un rectángulo con sus nombre, atributos y operaciones.

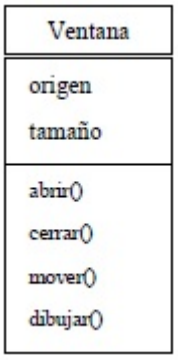

Figura 3.2: Representación gráfica de una clase.

3.5.2. Interfaz

Es una colección de operaciones que especifican un servicio de una determinada clase o componente. Una interfaz describe el comportamiento de varias operaciones pero nunca su implementación. La interfaz solo muestra los componentes visibles externos o solo una parte. Está representada por un circulo.

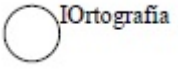

Figura 3.3: Representación gráfica de una interfaz.

3.5.3. Colaboración

Una colaboración define una interacción y es un conjunto de roles y otros elementos que colaboran para proporcionar un comportamiento cooperativo mayor que la suma de los comportamientos de sus elementos. Las colaboraciones tiene una dimensiones tanto estructural, como de comportamiento. Está representado por una elipse con borde discontinuo.

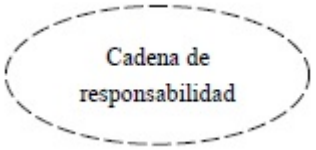

Figura 3.4: Representación gráfica de una colaboracion.

3.5.4. Casos de uso

Un caso de uso es la descripción de un conjunto de acciones que un sistema ejecuta y produce un determinado resultado que es de interés para un actor en general. Se usa principalmente para organizar el comportamiento de un modelo. Está representado por una elipse con borde continuo.

Figura 3.5: Representación gráfica de un caso de uso.

3.5.5. Actor

Un actor es una persona, sistema o despositivo que interactua con el sistema, iniciando, recibiendo resultados o participando en un caso de uso (Gil (2003)). Se representa de la siguiente manera:

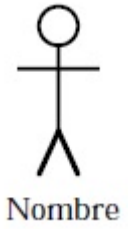

Figura 3.6: Representación gráfica de un actor.

3.5.6. Clase activa

Es una clase cuyos objetos tienen uno o mas hilos en ejecución y por lo tanto pueden dar lugar a actividades de control. Se representa de la misma forma que una clase, pero con bordes mas gruesos.

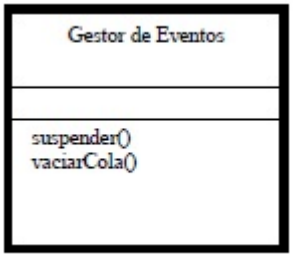

Figura 3.7: Representación gráfica de una clase activa.

3.5.7. Componentes

Un componente es una parte física y reemplazable de un sistema que proporciona interfaces y la implementación de dicho conjunto. Un componente representa comúnmente el empaquetamiento físico de diferentes elementos lógicos, como clases, interfaces y colaboraciones.

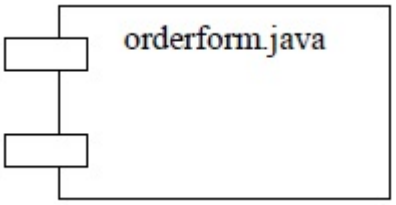

Figura 3.8: Representación gráfica de un componente.

3.5.8. Nodos

Es un elemento físico que existe en el tiempo de ejecución. Representa un recurso computacionál que dispone algo de memoria, y con frecuencia, de capacidad de procesamiento.

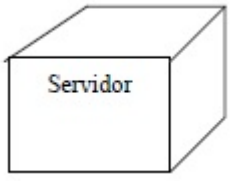

Figura 3.9: Representación gráfica de un nodo.

3.6. Elementos de comportamiento

Estos elementos son las partes dinámicas del modelo, y representan el comportamiento del tiempo y el espacio.

3.6.1. Interacción

Este comportamiento comprende una serie de mensajes intercambiados entre un conjunto de objetos, dentro de un contexto en particular para llegar a un resultado en específico. Puede involucrar muchos otros elementos, como mensajes secuencias de acción y enlaces.

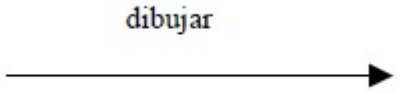

Figura 3.10: Representación gráfica de una interacción.

3.6.2. Máquinas de estado

Es un comportamiento que especifica los distintos estados por los que v pasando un objeto o las interacciones durante su vida en respuesta a eventos. Una máquina de estado puede involucrar a muchos otros elementos, como estados transiciones, eventos y actividades.

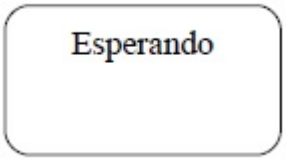

Figura 3.11: Representación gráfica de una maquina de estado.

3.7. Elementos de agrupación

El principal elemento de agrupación es el paquete, cuyo propósito es organizar los distintos elementos en grupos. Un paquete es solamente conceptual (existe solamente en el tiempo de desarrollo). Gráficamente se representa como una carpeta con su nombre, y a veces, su contenido.

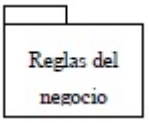

Figura 3.12: Representación gráfica de un paquete.

3.8. Elementos de anotación

Estos elementos son partes explicativas de los modelos UML. Son comentarios que se usan para describir, clasificar y hacer observaciones sobre cualquier elemento de un modelo.

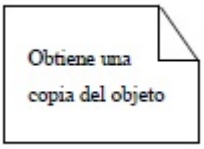

Figura 3.13: Representación gráfica de una nota.

3.9. Relaciones

En las relaciones se habla de una clase destino y de una clase origen. La clase origen es desde la que se realiza la acción de relacionar. Es decir, desde la que parte la flecha. La clase destino es la que recibe la flecha. Las relaciones se pueden modificar con estereotipos o con restricciones.

3.9.1. Dependencia

Es una relación semántica entre dos elementos en el cual, un cambio a un elemento (el elemento independiente) puede afectar a la semántica del otro (elemento dependiente). Se representa por una linea discontinua, posiblemente dirigida, y que puede incluir una etiqueta.

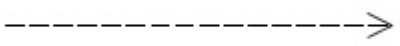

Figura 3.14: Representación gráfica de una relación de dependencia.

3.9.2. Asociación

Es una relación estructural que describe un conjunto de enlaces, los cuales son conexiones entre objetos. La agregación es u tipo de relación y representa una relación estructural entre un todo y sus partes. La asociación se representa con un alinea continua, posiblemente dirigida y que a veces incluye una etiqueta.

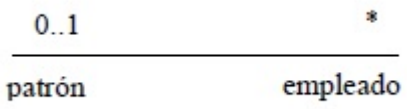

Figura 3.15: Representación gráfica de una relación de asociación.

3.9.3. Generalización

Es una relación de generalización/especialización, en la cual los objetos del elemento especializado (el hijo) pueden a los objetos del elemento general (el padre). De esta forma, el hijo comparte la estructura y comportamiento del padre. Se representa con una linea con una punta de flecha vacía.

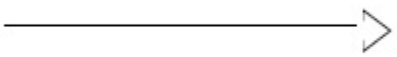

Figura 3.16: Representación gráfica de una relación de generalización.

3.9.4. Realización

Es una relación semántica entre clasificadores, donde un clasificador especifica un contrato que otro clasificador garantiza que cumplirá. Se pueden encontrar relaciones de realización en dos sitios: entre interfaces y las clases y componentes que las realizan, y entre los casos de uso y las colaboraciones que los realizan. Se representa con una linea discontinua con una punta de flecha vacía.

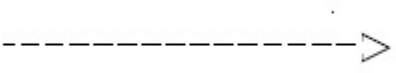

Figura 3.17: Representación gráfica de una relación de realización.

3.10. Diagramas

En UML existen distintos diagramas que muestran el comportamiento o la estructura de un sistema, mostrando distintas perspectivas del mismo.

3.10.1. Diagrama de clases

Este diagrama muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Cubren la vista de diseño estática.

3.10.2. Diagrama de secuencia y colaboración

Los diagramas de secuencia y los de colaboración son diagramas de interacción. Consisten en un conjunto de objetos y relaciones, incluyendo los mensajes que puedan tener de un objeto a otro. La diferencia entre ambos diagramas es que el diagrama de secuencia enfatiza el ordenamiento temporal de los mensajes, mientras que en los diagramas de colaboración se muestra la estructura de los objetos que envían mensajes.

3.10.3. Diagrama de casos de uso

3.10.4. Diagrama de estructura

3.11. Sistema de información

Para comprender de una mejor manera este concepto, primero abordaremos los conceptos de *sistema* e *infomación*.

3.11.1. Sistema

3.11.2. Información

3.12. Modelo Entidad-Relación

Capítulo 4

Desarrollo de Actividades

4.1. Metodología propuesta

La metodología propuesta para el desarrollo de este proyecto es Programación Extrema (XP).

4.1.1. Justificación

El uso de ésta metodología ágil ofrece múltiples ventajas en el desarrollo de proyectos de software en cuanto a la planeación y pruebas, ya que la tasa de errores es baja, facilita los cambios, fomenta una amplia comunicación entre los desarrolladores y el cliente. Además de que ésta metodología puede ser usada para cualquier lenguaje de programación.

4.2. Etapas del proyecto

4.3. Historias de usuario

4.4. Requerimientos

4.4.1. Requerimientos funcionales

- Establecer conexión simultanea desde el sistema de escritorio y android a la misma base de datos.
- Actualización automática de la información a la base de datos.
- Poner la información en espera si no se ha detectado alguna conexión a internet.
- Generación automática de reportes.

4.4.2. Requerimientos no funcionales

- Interfaz simple y comprensible para el usuario.
- Confirmaciones antes y después de actualizar la base de datos.
- Base de datos de respaldo.

4.5. Recursos

Se describen a continuación los recursos que se utilizaron para el desarrollo de este trabajo:

4.5.1. Recursos tecnológicos

- Visual Studio 2017
- Android Studio
- Enterprise Architect
- Balsamiq Mockups
- NetBeans
- MySQL

4.5.2. Recursos humanos

4.6. Diseño

El diseño de la interfaz de usuario para la versión de escritorio y android se realizó en Balsamiq Mockups.

4.6.1. Diseño del sistema de escritorio

4.6.2. Diseño del sistema de android

4.7. Codificación

4.8. Pruebas

Capítulo 5

Conclusiónes y perspectivas de trabajo

5.0. CAPÍTULO 5. CONCLUSIÓNES Y PERSPECTIVAS DE TRABAJO

Bibliografía

Alarcón, R. (2000). Diseño orientado a objetos con uml. *Grupo Eidos*.

Beck, K. (1999). Embracing change with extreme programming. *Computer*, 32(10):70–77.

Burgués, J. E. G. (2014). *Aprende a modelar aplicaciones con UML*. IT Campus Academy.

Canós, J. H. and Letelier, M. C. P. P. (2012). Metodologías ágiles en el desarrollo de software.

Chiesa, F. (2004). Metodología para selección de sistemas erp. *Reportes técnicos en ingeniería del software*, 6(1):17–37.

Gil, S. V. H. (2003). Representación de la arquitectura de software usando uml.

Jacobson, I., Booch, G., and Rumbaugh, J. (2000). El lenguaje unificado de modelado. *Manual de Referencia*.

Pressman, R. S. (2005). *Ingeniería del Software: Un enfoque práctico*. McGraw-Hill Interamericana,.

Sommerville, I. (2005). *Ingeniería del software*. Pearson educación.

Sparks, G. (2000). Una introduccion al uml. *El Modelo Lógico. Recuperado de: http://www. sparxsystems. com. es/downloads/whitepapers/El_Modelo_Logico. pdf*.

5.0. BIBLIOGRAFÍA

Anexos