|  |  |  |  |
| --- | --- | --- | --- |
| **ASIGNATURA: ELEMENTOS DE INGENIERÍA DE**  **SOFTWARE** | ***Comisión*** | ***Trabajo Práctico*** | |
| **760 - Elementos de Ingeniería de Software - A** | **Guía de Trabajos Prácticos** | |
| ***Nro.*** | ***Fecha*** |
| **-** | **18/07/2020** |

**Guía de Trabajos Prácticos**

**Grupo N°1**

**Año 2020**

**Integrantes del equipo:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre y Apellido** | **N° Legajo** | **E-mail** | **Teléfono** |
| **Florencia Massey** |  | **florenciamssy@gmail.com** | **02227475160** |
| **Juan Manuel Gerez** |  | **juanmadaria92@gmail.com** | **1158077295** |
| **Pablo Gerez** |  | **pablogerez2000@gmail.com** | **1164081371** |
| **Joaquín Pettinari** |  | **joaquinpettinari@hotmail.com** | **1161511893** |
| **Tiago Cabrera** |  | [**tiagocabrera14@outlook.com**](mailto:tiagocabrera14@outlook.com) | **1144094733** |

**Docente:**

**Prof. Mg. Marisa Daniela Panizzi**

***Casos Prácticos***

[***Caso Práctico N°1: “La Biblioteca”*** 3](#_Toc45920890)

[***Caso Práctico N°2: “Aparato respiratorio”*** 3](#_Toc45920891)

[***Caso Práctico N°3: La farmacia y su entorno*** 3](#_Toc45920892)

[***Caso Práctico N°4: Tessei-plex S.A*** 3](#_Toc45920893)

[***Caso Práctico N°5: Observando sistemas, subsistemas y supersistemas*** 3](#_Toc45920894)

[***Caso Práctico N°6: Modelizando el sistema lavadero de autos. Análisis de caja negra.*** 3](#_Toc45920895)

[***Caso Práctico N°7: Revisando Conceptos.*** 4](#_Toc45920896)

[***Caso Práctico N°8: Guías de Lectura*** 4](#_Toc45920897)

[***Caso Práctico N°9: Buscando Sistemas Informáticos*** 10](#_Toc45920898)

[***Caso Práctico N°10: Guía de lectura*** 11](#_Toc45920899)

[***Caso Práctico N°11: Aplicando Técnicas de modelado: Caso de Uso.*** 14](#_Toc45920900)

[***Caso Práctico N°12: Aplicando Técnicas de modelado: Caso de Uso.*** 16](#_Toc45920901)

[***Caso Práctico N°13: Aplicando Técnicas de Modelado: Historias de Usuario*** 18](#_Toc45920902)

[***Caso Práctico N°14: Aplicando Técnicas de Modelado: Historias de Usuario*** 18](#_Toc45920903)

[***Caso Práctico N°15: Comprendiendo el dominio de la aplicación: Modelo de dominio.*** 18](#_Toc45920904)

[***Caso Práctico N°16: Comprendiendo el dominio de la aplicación: Modelo de dominio.*** 19](#_Toc45920905)

[***Caso Práctico N°17: Investigando las Metodologías Ágiles: Sus aportes.*** 19](#_Toc45920906)

[***Caso Práctico N°18: Aplicando Técnicas de estimación: Puntos de caso de uso.*** 19](#_Toc45920907)

[***Caso Práctico N°19: Aplicando Técnicas de estimación: Puntos de caso de uso.*** 21](#_Toc45920908)

[***Caso Práctico N°20: Lectura de un artículo de un Congreso Nacional: Clasificación de métricas*** 23](#_Toc45920909)

[***Caso Práctico N°21: Aplicando Técnicas de estimación: Planning Pocker.*** 24](#_Toc45920910)

[***Caso Práctico N°22: Aplicando Técnicas de estimación: Planning Pocker.*** 24](#_Toc45920911)

[***Caso Práctico N°23. Comparando Técnicas de Estimación: Puntos de casos de uso & Puntos de historias de usuarios.*** 25](#_Toc45920912)

[**Caso Práctico N°24: Diagrama estructural del sistema “INMOBILIARIA NORTE”: Diagrama de clases.** 25](#_Toc45920913)

[***Caso Práctico N°25: Diagrama estructural del sistema “ESCUELA SIGLO XXI”: Diagrama de clases.*** 26](#_Toc45920914)

[***Caso Práctico N°26: Construyendo el modelo de análisis - Diagrama de comunicación.*** 26](#_Toc45920915)

[***Caso Práctico N°27: Construyendo el modelo de análisis - Diagrama de comunicación.*** 27](#_Toc45920916)

# ***Caso Práctico N°1: “La Biblioteca”***

1. Una biblioteca puede ser considerada un sistema porque posee una función y es un conjunto de elementos interdependientes. Cumple con las características de sistema abierto como para poder ser considerada como tal.
2. El entorno de la biblioteca serían los estudiantes, docentes y no docentes.
3. La función de una biblioteca es almacenar y llevar un inventario de libros, pudiendo ser prestados a los estudiantes que lo precisen.

# ***Caso Práctico N°2: “Aparato respiratorio”***

1. Si, ya que el todo como un sistema, en este caso respiratorio, depende de sus partes, estando las mismas interconectadas y afectando con su comportamiento a las demás.
2. Obtener oxígeno a partir del ambiente externo para posteriormente transferirlo a la sangre y eliminar del organismo el dióxido de carbono.
3. Partes del aparato respiratorio:
   1. Fosas nasales
   2. Faringe.
   3. Laringe.
   4. Tráquea.
   5. Pulmones.
   6. Bronquios, bronquiolos y alvéolos.

# ***Caso Práctico N°3: La farmacia y su entorno***

* El ejemplo A no cumple las condiciones necesarias para ser considerado un sistema, ya que, al tener dos entradas y ninguna salida, se saturaría.
* El ejemplo B tampoco cumple las condiciones, pero en este caso tiene dos salidas y carece de entradas.

# ***Caso Práctico N°4: Tessei-plex S.A***

* La función del sistema de compras es adquirir materia prima, basándose en los datos que recibe de otras partes del sistema (stock, demanda proyectada, etc.).
* El límite se encuentra desde que se realiza la compra hasta que le llega la materia prima.
* Los subsistemas restantes que lo rodean, es decir, la empresa en su totalidad.

# ***Caso Práctico N°5: Observando sistemas, subsistemas y supersistemas***

1. Ejemplo 1:

* Sistema: Sistema Solar
* Subsistemas: Planetas
* Sistema mayor: Universo

1. Ejemplo 2:

* Sistema: Persona
* Subsistemas: Órganos
* Sistema mayor: Sociedad

# ***Caso Práctico N°6: Modelizando el sistema lavadero de autos. Análisis de caja negra.***

1. El entorno del lavadero son los clientes. La relación que existe entre el sistema y su entorno es trabajar con el vehículo que lleva el cliente.
2. **Entrada:** automóvil sucio.

**Salida:** automóvil limpio.

1. Realizar la correcta limpieza del vehículo.

# ***Caso Práctico N°7: Revisando Conceptos.***

* **Dato**: Representación elemental de la información.
* **Información**: Cualquier clase de conocimiento o mensaje que puede usarse para mejorar o posibilitar una decisión o acción.
* **Software:** Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.
* **Sistema de Información:** Sistema de información se entiende como el conjunto de tecnologías, procesos, aplicaciones de negocios y software disponibles para las personas dentro de una organización.
* **Software de sistemas:** Es el conjunto de instrucciones que permiten el manejo de la computadora. Una computadora sin software de sistema se hace inmanejable. Consiste en un software que sirve de soporte o base para controlar e interactuar con el hardware y otros programas.
* **Ingeniería de Software:** Es una disciplina formada por un conjunto de métodos, herramientas y técnicas que se utilizan en el desarrollo de los programas informáticos.
* **Proceso de software:** Proceso de continuo [aprendizaje](https://www.monografias.com/trabajos5/teap/teap.shtml) mediante el cual una [organización](https://www.monografias.com/trabajos6/napro/napro.shtml) mejora y se mejora a través de procesos adquiridos y/o sus propios procesos.
* **Modelo:** Los modelos de información son una representación formal y abstracta de los tipos de entidades que existen en un dominio. Dichas entidades pueden ser representaciones de objetos reales u objetos propios de los sistemas de software y de los procesos del negocio.
* **Metodología:** Conjunto de actividades que logran un objetivo.Una metodología de desarrollo de software se refiere a un framework (entorno o marco de trabajo) que es usado para estructurar, planear y controlar el proceso de desarrollo en sistemas de información. Una Metodología para el Desarrollo de Sistemas de Información es un conjunto de actividades llevadas a cabo para desarrollar y poner en marcha un Sistema de Información.

Los Objetivos de las Metodologías de Desarrollo de Sistemas de Información son:

* Definir actividades a llevarse a cabo en un Proyecto de S.I.
* Unificar criterios en la organización para el desarrollo de S.I.
* Proporcionar puntos de control y revisión.

Fuentes

* <https://es.wikipedia.org/wiki/Software>
* <https://es.wikipedia.org/wiki/Sistema_de_informaci%C3%B3n>
* <https://es.wikipedia.org/wiki/Software_de_sistema>
* <https://definicion.de/ingenieria-de-software/>
* <https://www.monografias.com/trabajos96/procesos-de-software/procesos-de-software.shtml>
* <https://www.monografias.com/trabajos90/metodologia-desarrollo-sistema-informacion/metodologia-desarrollo-sistema-informacion.shtml>
* <https://es.wikipedia.org/wiki/Metodolog%C3%ADa_de_desarrollo_de_software>
* <https://es.wikipedia.org/wiki/Modelo_de_Informaci%C3%B3n>

# ***Caso Práctico N°8: Guías de Lectura***

**Capítulo 1 – Ingeniería del Software de Ian Sommerville**

1. ¿Por qué se dice que la Ingeniería del Software es “abstracta e intangible”? ¿Es positivo o negativo este concepto? ¿Por qué?

Porque no está restringido por materiales, o gobernado por leyes físicas o por procesos de manufactura. Dependiendo del punto de vista puede ser tanto positivo como negativo porque ya no existen limitaciones físicas del potencial del software, pero a su vez se hace más difícil de comprender.

1. ¿Por qué no es aplicable un enfoque informal en la Ingeniería del Software?

Un enfoque informal no es aplicable en la Ingeniería del Software ya que no es efectivo para producir software de alta calidad, el cual requiere técnicas sistemáticas y organizadas.

1. Justifique la siguiente afirmación: “No hay un enfoque ideal en la Ingeniería del Software”.

La amplia diversidad de diferentes de tipos de sistemas y organizaciones que usan estos sistemas significa que necesitamos una diversidad de enfoques al desarrollo de software.

No existe una forma correcta de desarrollar un determinado software, todo se tiene que adaptar al producto que se requiere.

1. ¿Qué agrega el autor a la definición de software tradicional?

“Yo prefiero una definición más amplia donde el software no son sólo programas, sino todos los documentos asociados y la configuración de datos que se necesitan para hacer que estos programas operen de manera correcta”.

1. Establecer la diferencia entre Productos de Software Genéricos y Productos de Software personalizados. Dar ejemplos de cada tipo.

El software genérico es un sistema que se desarrolla para un mercado abierto. Se venden a clientes a los que le sea posible comprarlos. Ejemplo: procesador de textos (Word), herramienta de gestión de proyectos (Enterprise architect).

Un software personalizado es un sistema en específico, requerido por un cliente en particular. Un contratista de software desarrolla el software especialmente para ese cliente. Ejemplo: sistema de turnos online para una determinada clínica.

1. Definir Ingeniería del Software

La ingeniería del software es una disciplina de la ingeniería que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de éste después de que se utiliza.

1. Enumere las etapas de un proceso de software. Indique si la siguiente afirmación es verdadera o falsa y justifique su elección: “Las etapas de un proceso de software se desarrollan siempre en orden. La próxima etapa se inicia cuando la anterior ha concluido.”
   * 1. Especificación del software: donde los clientes e ingenieros definen el software a producir y las restricciones sobre su operación.
     2. Desarrollo del software: donde el software se diseña y programa.
     3. Validación del software: donde el software se valida para asegurar que es lo que el cliente requiere.
     4. Evolución del software: donde el software se modifica para adaptarlo a los cambios requeridos por el cliente y el mercado.

“Las etapas de un proceso de software se desarrollan siempre en orden. La próxima etapa se inicia cuando la anterior ha concluido.”

Esta frase es verdadera porque para que un sistema se realice de forma eficiente hay que seguir las etapas, como si fuese una receta. Como una tortilla de papas.

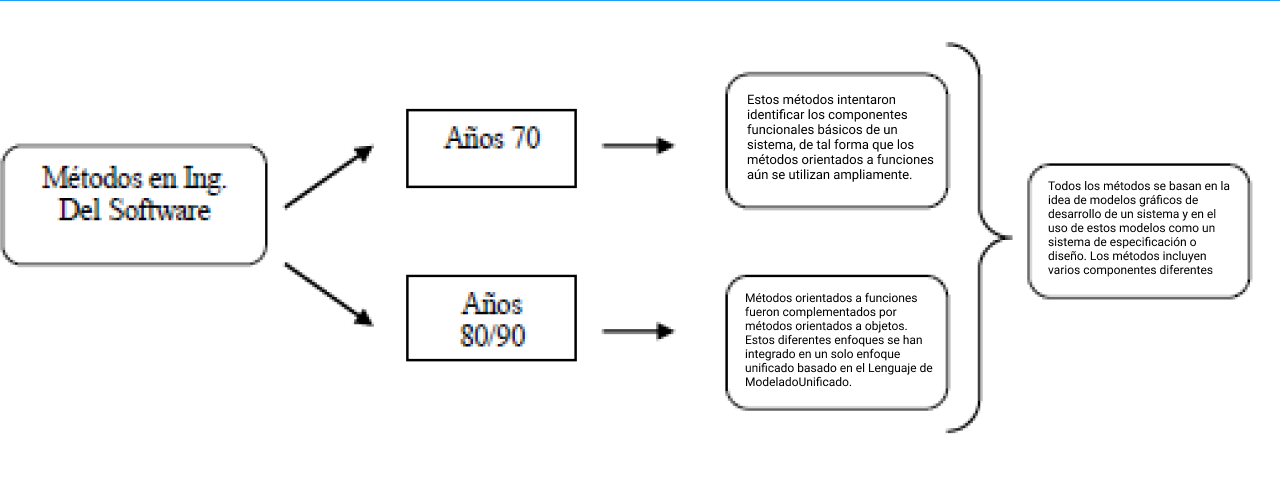
1. ¿Cuáles son los tres modelos generales de desarrollo de software?

* Un modelo de flujo de trabajo.
* Un modelo de flujo de datos o de actividad.
* Un modelo de rol/acción.

1. Mencione al menos tres cuestiones relacionadas con la determinación de los costos en la Ingeniería del Software.

* Proceso utilizado
* Tipo de software
* Tiempo requerido

1. Complete el siguiente esquema:



1. Leer el texto “Responsabilidad profesional y ética”. Realice una síntesis del mismo.

Los ingenieros deben mantener un comportamiento ético para ser considerados como profesionales. No se deben utilizar las capacidades y habilidades para comportarse de forma deshonesta. Existen áreas donde los estándares de comportamiento aceptable se rigen por sentido común, como, por ejemplo:

* Confidencialidad.
* Competencia.
* Derechos de propiedad intelectual.
* Uso inapropiado de las computadoras.

Las sociedades e instituciones profesionales tienen que desempeñar un papel importante en el establecimiento de estándares éticos. Como por ejemplo la ACM, el IEEE (Instituto de Ingenieros Eléctricos y Electrónicos) y la British Computer Society.

**Capítulo 4 – Ingeniería del Software de Ian Sommerville**

1. ¿Qué es un proceso de software? ¿Por qué se dice que "los procesos de software son complejos”?

Un proceso del software es un conjunto de actividades que conducen a la creación de un producto software. Se dice que los procesos de software son complejos porque el software se amplía, modifica o integra otros sistemas externos. También porque dependen de las personas que toman decisiones y juicios.

1. ¿Cuáles son las actividades comunes a todo proceso de software? Describir brevemente.

* Especificación del software: Se debe definir la funcionalidad del software y las restricciones en su operación.
* Diseño e implementación del software: Se debe producir software que cumpla su especificación.
* Validación del software: Se debe validar el software para asegurar que hace lo que el cliente desea.
* Evolución del software: El software debe evolucionar para cubrir las necesidades cambiantes del cliente.

1. Nombrar y describir los tres modelos de procesos de software genéricos.

* El modelo en cascada: considera las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución, y los representa como fases separadas del proceso, tales como la especificación de requerimientos, el diseño del software, la implementación, las pruebas.
* Desarrollo evolutivo: este enfoque entrelaza las actividades de especificación, desarrollo y validación. Un sistema inicial se desarrolla rápidamente a partir de especificaciones abstractas. Este se refina basándose en las peticiones del cliente para producir un sistema que satisfaga sus necesidades.
* Ingeniería del software basada en componentes: este enfoque se basa en la existencia de un número significativo de componentes reutilizables. El proceso de desarrollo del sistema se enfoca en integrar estos componentes en el sistema más que en desarrollarlos desde cero.

1. Indique si la siguiente frase es Verdadera o Falsa y justifique su respuesta: "Los modelos de procesos de software se utilizan siempre de manera aislada"

**Falsa**. Los modelos de procesos de software se pueden utilizar juntos, especialmente en el desarrollo de sistemas grandes.

1. Enuncie las etapas del Modelo en Cascada. Describir brevemente cada una de ellas.

* Análisis y definición de requerimientos. Los servicios, restricciones y metas del sistema se definen a partir de las consultas con los usuarios. Entonces, se definen en detalle y sirven como una especificación del sistema.
* Diseño del sistema y del software. El proceso de diseño del sistema divide los requerimientos en sistemas hardware o software. Establece una arquitectura completa del sistema. El diseño del software identifica y describe las abstracciones fundamentales del sistema software y sus relaciones.
* Implementación y prueba de unidades. Durante esta etapa, el diseño del software se lleva a cabo como un conjunto o unidades de programas. La prueba de unidades implica verificar que cada una cumpla su especificación.
* Integración y prueba del sistema. Los programas o las unidades individuales de programas se integran y prueban como un sistema completo para asegurar que se cumplan los requerimientos del software. Después de las pruebas, el sistema software se entrega al cliente.
* Funcionamiento y mantenimiento. Por lo general, ésta es la fase más larga del ciclo de vida. El sistema se instala y se pone en funcionamiento práctico. El mantenimiento implica corregir errores no descubiertos en las etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema y resaltar los servicios del sistema una vez que se descubren nuevos requerimientos.

1. Diferenciar ventajas y desventajas del Modelo en Cascada.

Ventajas

* La documentación se produce en cada fase.
* Cuadra con otros modelos del proceso de ingeniería.

Desventajas

* Es inflexible al momento de dividir el proyecto en distintas etapas.
* Solo para requerimientos que no cambien durante el desarrollo del sistema.

1. ¿Cuál es el planteo que propone el desarrollo evolutivo?

El desarrollo evolutivo se basa en la idea de desarrollar una implementación inicial, exponiéndose a los comentarios del usuario y refinando a través de las diferentes versiones hasta que se desarrolla un sistema adecuado.

Las actividades de especificación, desarrollo y validación se entrelazan en vez de separarse, con una rápida retroalimentación entre éstas.

1. Establezca ventajas y desventajas del desarrollo evolutivo. ¿En qué tipos de sistemas es mejor aplicarlo?

Ventajas

* Satisface las necesidades inmediatas de los clientes.
* La especificación se puede desarrollar de forma creciente.

Desventajas

* El proceso no es visible, los administradores tienen que hacer entregas regulares para medir el progreso. Si los sistemas se desarrollan rápidamente, no es rentable producir documentos que reflejen cada versión del sistema.
* A menudo los sistemas tienen una estructura deficiente, los cambios continuos tienden a corromper la estructura del software. Incorporar cambios en él se convierte cada vez más en una tarea difícil y costosa.

Es mejor aplicarlo en sistemas pequeños y de medianos.

1. ¿Qué metodología sugiere la bibliografía para sistemas de gran tamaño?

Para sistemas de gran tamaño, se recomienda un proceso mixto que incorpore las mejores características del modelo en cascada y del desarrollo evolutivo.

1. ¿A qué se llama Ingeniería de software basada en componentes? ¿Cuál es su principal diferencia respecto de otras metodologías?

La Ingeniería de software basada en componentes es un enfoque asentado en la reutilización que se compone de una gran base de componentes software reutilizables y de algunos marcos de trabajo de integración para éstos. Algunas veces estos componentes son sistemas por sí mismos que se pueden utilizar para proporcionar una funcionalidad específica, como dar formato al texto o efectuar cálculos numéricos.

La principal diferencia respecto de otras metodologías está en sus etapas intermedias.

1. ¿A qué se llama iteración de procesos? ¿Cuáles son los métodos basados en este concepto?

Se llama iteración de procesos a las actividades que se repiten regularmente a medida que un sistema va evolucionando en respuesta a requerimientos de cambios.

Los métodos basados en este concepto son los siguientes:

* Entrega incremental. La especificación, el diseño y la implementación del software se dividen en una serie de incrementos, los cuales se desarrollan por turnos.
* Desarrollo en espiral. El desarrollo del sistema gira en espiral hacia fuera, empezando con un esbozo inicial y terminando con el desarrollo final del mismo.

1. Enuncie ventajas y desventajas de los procesos iterativos.

Ventajas

* La especificación se desarrolla junto con el software.

Desventajas

* Crea conflictos con el modelo de obtención de muchas organizaciones donde la especificación completa del sistema es parte del contrato de desarrollo del mismo.

1. Describa las principales características, ventajas y desventajas del modelo de entrega incremental.

La entrega incremental es un enfoque intermedio que combina las ventajas del modelo de desarrollo en cascada y el modelo de desarrollo evolutivo. En un proceso de desarrollo incremental. los clientes identifican, a grandes rasgos, los servicios que proporcionará el sistema. Identifican qué servicios son más importantes y cuáles menos. Entonces, se definen varios incrementos en donde cada uno proporciona un subconjunto de la funcionalidad del sistema.

Una vez que un incremento se completa y entrega, los clientes pueden ponerlo en servicio.

Ventajas

* Los clientes no tienen que esperar hasta que el sistema completo se entregue para sacar provecho de él. El primer incremento satisface los requerimientos más críticos de tal forma que pueden utilizar el software inmediatamente.
* Los clientes pueden utilizar los incrementos iniciales como prototipos y obtener experiencia sobre los requerimientos de los incrementos posteriores del sistema.
* Existe un bajo riesgo de un fallo total del proyecto.
* Puesto que los servicios de más alta prioridad se entregan primero, y los incrementos posteriores se integran en ellos, es inevitable que los servicios más importantes del sistema sean a los que se les hagan más pruebas. Esto significa que es menos probable que los clientes encuentren fallos de funcionamiento del software en las partes más importantes del sistema.

Desventajas

* Los incrementos deben ser relativamente pequeños y cada uno debe entregar alguna funcionalidad del sistema.
* Dificultad en la adaptación de los requerimientos del cliente a incrementos de tamaño apropiado.
* Muchos de los sistemas requieren un conjunto de recursos que se utilizan en diferentes partes del sistema.

1. ¿En qué consiste el modelo de desarrollo en espiral? ¿Cuál es el manejo que hace respecto del riesgo de errores?

Cada ciclo en la espiral representa una fase del proceso del software. Así. el ciclo más interno podría referirse a la viabilidad del sistema, el siguiente ciclo a la definición de requerimientos, el siguiente ciclo al diseño del sistema, y así sucesivamente.

Un ciclo de la espiral empieza con la elaboración de objetivos, como el rendimiento y la funcionalidad. Entonces se enumeran formas alternativas de alcanzar estos objetivos y las restricciones impuestas en cada una de ellas. Cada alternativa se evalúa contra cada objetivo y se identifican las fuentes de riesgo del proyecto. El siguiente paso es resolver estos riesgos mediante actividades de recopilación de información como la de detallar más el análisis, la construcción de prototipos y la simulación. Una vez que se han evaluado los riesgos, se lleva a cabo cierto desarrollo seguido de una actividad de planificación para la siguiente fase del proceso.

# ***Caso Práctico N°9: Buscando Sistemas Informáticos***

Sistema de Gestión: **Inmosoft** -<https://www.inmosoft.com.ar>

Dominio de la aplicación: **Inmobiliarias**

Requisitos funcionales:

* Contratos de alquiler
* Impresión de recibos
* Facturación electrónica AFIP Gestión de inmuebles
* Liquidaciones a propietarios
* Agenda y administración de personas
* Caja diaria y balance de ganancias
* Informes, análisis y toma de decisiones
* Administración de alquileres temporales
* Administración de consorcio

Requisitos no funcionales:

* El sistema es compatible solo con sistema operativo Windows.
* Para su correcto funcionamiento el sistema requiere acceso a Internet.

Sistema de Gestión: **Mercasoft** - <https://www.solinsur.net/mercasof/mercasof.aspx>

Dominio de la aplicación: **Supermercados**

Requisitos funcionales:

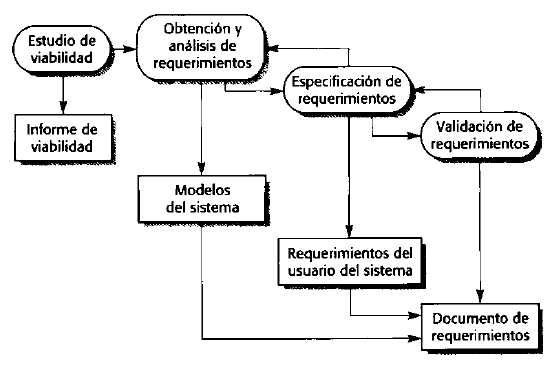
* Terminal de punto de venta
* Módulo de almacén
* Gestión de ficheros
* Configuración de seguridad
* Garantía de calidad
* Circuito completo de compras
* Gestión de promociones
* Manejo de estadísticas e informes
* Configuración de manejo de red
* Gestión de contabilidad

Requisitos no funcionales:

* El sistema es compatible solo con sistema operativo Windows.
* Mínimo de 512 MB de RAM para su instalación.

# ***Caso Práctico N°10: Guía de lectura***

1. Explicar brevemente el diagrama que representa los Procesos de Ingeniería de Requerimientos.

****

La meta del proceso de ingeniería de requerimientos es crear y mantener un documento de requerimientos del sistema. El proceso general corresponde cuatro subprocesos de alto nivel de la ingeniería de requerimientos. Estos tratan de la evaluación de si el sistema es útil para el negocio (estudio de viabilidad); el descubrimiento de requerimientos (obtención y análisis); la transformación de estos requerimientos en formularios estándar (especificación), y la verificación de que los requerimientos realmente definen el sistema que quiere el cliente (validación).

1. ¿Cuáles son las diferencias entre el enfoque general que describió en el punto anterior y el Modelo Espiral de Procesos de Ingeniería de Requerimientos?

* El modelo espiral de procesos comprende 3 etapas principales mientras que el enfoque general consta de 4.
* El modelo espiral contempla y considera explícitamente los posibles riesgos que puede presentar un modelo.

1. ¿A qué se llama Estudio de Viabilidad? ¿Qué intenta resolver? ¿Qué se obtiene en esta etapa? ¿Cuáles pueden ser las causas de que un sistema resulte no viable de ser desarrollado?

El estudio de viabilidad es el análisis que se realiza previo a la ejecución de un proyecto para evaluar su factibilidad. Comienza a partir de un conjunto de requerimientos de negocio preliminares, una descripción resumida del sistema y de cómo éste pretende contribuir a los procesos del negocio.

En un estudio de viabilidad se plantean los siguientes interrogantes:

* ¿Contribuye el sistema a los objetivos generales de la organización?
* ¿Se puede implementar el sistema utilizando la tecnología actual y dentro de las restricciones de coste y tiempo?
* ¿Puede integrarse el sistema con otros sistemas existentes en la organización?

Si un sistema no contribuye a los objetivos del negocio, es innecesario que sea desarrollado.

A partir de esta etapa se obtiene un informe que recomienda si merece o no la pena seguir con la ingeniería de requerimientos y el proceso de desarrollo del sistema.

1. ¿Con quiénes trabajan los Ingenieros de Software en la etapa de obtención y análisis de requerimientos? ¿Cuál es el término utilizado? Defínalos y dé ejemplos a partir de un caso concreto de sistema a construir.

Los ingenieros de software trabajan con los clientes y los usuarios finales del sistema para determinar el dominio de la aplicación, qué servicios debe proporcionar el sistema, el rendimiento requerido del sistema, las restricciones hardware, etc.

El término stakeholder es utilizado para referirse a cualquier persona o grupo que se verá afectado por el sistema, directa o indirectamente.

Ejemplo: Campus de la UNaHur.

Stakeholders: Alumnos, Profesores, Directivos, Desarrolladores de Moodle.

1. Indique las razones por las cuales es difícil trabajar en la obtención y comprensión de los requerimientos. Redacte un ejemplo al respecto.

* Los stakeholders a menudo no conocen lo que desean obtener del sistema informático excepto en términos muy generales; puede resultarles difícil expresar lo que quieren que haga el sistema o pueden hacer demandas irreales debido a que no conocen el coste de sus peticiones.

Ejemplo: solicitan que se modifique cierta pantalla para agregar un menú desplegable, sin saber que para implementarla se requiere de un framework que no está presente en el sistema y el cual añadirlo sería muy costoso o prácticamente imposible.

* Los stakeholders expresan los requerimientos con sus propios términos de forma natural y con un conocimiento implícito de su propio trabajo. Los ingenieros de requerimientos, sin experiencia en el dominio del cliente, deben comprender estos requerimientos.
* Diferentes stakeholders tienen requerimientos distintos, que pueden expresar de varias formas. Los ingenieros de requerimientos tienen que considerar todas las fuentes potenciales de requerimientos y descubrir las concordancias y los conflictos.
* Los factores políticos pueden influir en los requerimientos del sistema. Por ejemplo, los directivos pueden solicitar requerimientos específicos del sistema que incrementarán su influencia en la organización.
* El entorno económico y de negocios en el que se lleva a cabo el análisis es dinámico. Inevitablemente, cambia durante el proceso de análisis. Por lo tanto, la importancia de ciertos requerimientos puede cambiar. Pueden emerger nuevos requerimientos de nuevos stakeholders que no habían sido consultados previamente.

1. Justifique la siguiente afirmación: “El proceso de obtención y análisis de requerimientos es iterativo con retroalimentación de cada actividad al resto de las actividades”.

El ciclo del proceso comienza con el descubrimiento de requerimientos y termina con la documentación de los mismos. La comprensión de los requerimientos por parte del analista mejora con cada vuelta del ciclo (iteración).

1. ¿Qué criterio suele utilizarse para clasificar requerimientos? ¿Cómo se establecen las prioridades entre los requerimientos que se obtuvieron?

El criterio que suele utilizarse para clasificar requerimientos es el de un modelo de arquitectura del sistema, para identificar los subsistemas y asociar los requerimientos con cada subsistema.

Durante el proceso de clasificación y organización de requerimientos se deben organizar frecuentes reuniones o negociaciones con los stakeholders para que se pueda llegar a tener acuerdos, debido a que estos tienen diferentes opiniones sobre la importancia y prioridad de los requerimientos.

1. Mencione distintas técnicas utilizadas para “elicitar” (descubrir) requerimientos. Analice sus ventajas y desventajas.

* Entrevistas

|  |  |
| --- | --- |
| Ventajas | Desventajas |
| Sirven para obtener una comprensión general de la interacción que llevarían a cabo los stakeholders con el sistema. | No son útiles para la comprensión de los requerimientos del dominio de la aplicación. |
| Pueden ser la única fuente de información sobre los requerimientos del sistema. | No es una técnica eficaz para ganar conocimiento sobre los requerimientos y restricciones organizacionales debido a que existen intereses personales entre los stakeholders en la organización. |
|  | Tienden a omitir información esencial. |

* Escenarios

|  |  |
| --- | --- |
| Ventajas | Desventajas |
| Se basa en ejemplos de interacción con el software en la vida real. |  |
| Son útiles para agregar detalles a un boceto de la descripción de requerimientos. |  |

* Etnografía

|  |  |
| --- | --- |
| Ventajas | Desventajas |
| Permite entender los requerimientos sociales y organizacionales. | Debido a que se centra en el usuario final, no es adecuado para descubrir los requerimientos organizacionales o del dominio. |
| Ayuda a los analistas a descubrir los requerimientos implícitos que reflejan los procesos reales. | No siempre identifica nuevas características para agregar al sistema. |
| Es efectiva para descubrir los requerimientos que se derivan de la forma en la que la gente trabaja realmente y los requerimientos que se derivan de la cooperación y conocimiento de las actividades de la gente. | No es un enfoque completo para la obtención de requerimientos, es complementario a otros más apropiados. |

* Casos de uso

|  |  |
| --- | --- |
| Ventajas | Desventajas |
| Utiliza el modelo UML (estándar para el modelado orientado a objetos), por lo que cada vez son más utilizados. | No son eficaces para obtener restricciones y requerimientos de negocio. |
| Identifica el tipo de interacción y los actores involucrados. | No son eficaces para descubrir requerimientos del dominio. |
|  | No son eficaces para obtener requerimientos no funcionales de alto nivel. |

* Métodos de análisis estructurado
* Construcción de prototipos del sistema

1. ¿En qué consiste la “Validación de Requerimientos”? ¿Qué técnicas de validación pueden aplicarse?

La validación de requerimientos trata de mostrar que éstos realmente definen el sistema que el cliente desea. Coincide parcialmente con el análisis ya que éste implica encontrar problemas con los requerimientos. La validación de requerimientos es importante debido a que los errores en el documento de requerimientos pueden conducir a importantes costes al repetir el trabajo cuando son descubiertos durante el desarrollo o después de que el sistema esté en uso.

Las técnicas de validación que pueden aplicarse son las siguientes:

* Revisiones de requerimientos: los requerimientos son analizados sistemáticamente por un equipo de revisores.
* Construcción de prototipos: en este enfoque de validación, se muestra un modelo ejecutable del sistema a los usuarios finales y a los clientes. Estos pueden experimentar con este modelo para ver si cumple sus necesidades reales.
* Generación de casos de prueba: los requerimientos deben poder probarse. Si las pruebas para éstos se conciben como parte del proceso de validación, a menudo revela los problemas en los requerimientos. Si una prueba es difícil o imposible de diseñar, normalmente significa que los requerimientos serán difíciles de implementar y deberían ser considerados nuevamente. Desarrollar pruebas para los requerimientos del usuario antes de que se escriba código es una parte fundamental de la programación extrema.

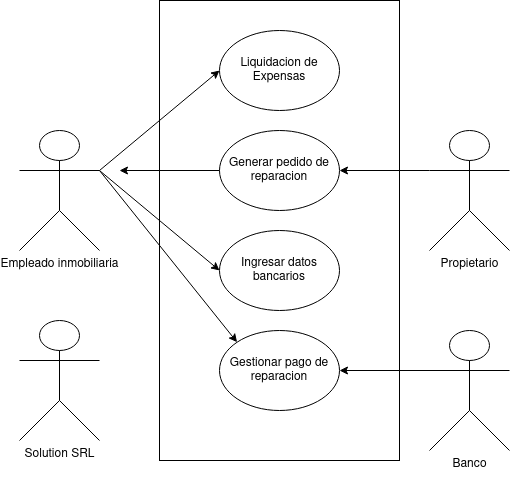
1. ¿A qué se refiere el concepto de “Gestionar Requerimientos”?

La gestión de requerimientos es el proceso de comprender y controlar los cambios en los requerimientos del sistema. Es necesario mantenerse al tanto de los requerimientos particulares y mantener vínculos entre los requerimientos dependientes de forma que se pueda evaluar el impacto de los cambios en los requerimientos.

# ***Caso Práctico N°11: Aplicando Técnicas de modelado: Caso de Uso.***

1. Requisitos funcionales y diagrama de casos de uso.

* Liquidación de expensas.
  + Entrada
    - Cada día 28 del mes.
  + Proceso
    - Verificar si tuve reparaciones.
    - Verificar si hay deuda de los propietarios.
    - Verificar los gastos bancarios.
  + Salida
    - Liquidación registrada + liquidación enviada.
* Generar pedido de reparación.
  + Entrada
    - Solicitud de reparación registrado.
  + Proceso
    - Verificar si es propietario.
  + Salida
    - Pedido de reparación.
* Gestionar pagos de reparación.
  + Entrada
    - Informe del costo de la reparación.
  + Salida
    - Inclusión en la próxima liquidación de expensas.
* Ingresar datos bancarios
  + Entrada
    - Recepción del listado de pagos de impuestos.
  + Salida
    - Inclusión en los gastos de expensas del edificio.





|  |  |
| --- | --- |
|  | Liquidación de expensas |
| Caso de Uso | Empleado de inmobiliaria |
| Nombre | Liquidación de expensas |
| Número | 1 |
| Tipo | Primario |
| Pre-Condición | Cada día 28 del mes. |
| Actor | Empleado de inmobiliaria |
| Descripción | El presente caso de uso permite realizar liquidación de expensas |
| Flujo Principal | 1. Verificar si tuvo reparaciones 2. Verificar si hay deuda de los propietarios 3. Verificar los datos bancarios 4. Calcular la liquidación |
| Flujo Alternativo | ---------- |
| Post-Condición | Liquidación registrada + liquidación enviada |
|  | Generar pedido  de reparación |
| Caso de Uso | Empleado de inmobiliaria |
| Nombre | Generar pedido de reparación |
| Número | 2 |
| Tipo | Secundario |
| Pre-Condición | Solicitud de reparación registrada |
| Actor | Empleado de inmobiliaria |
| Descripción | El presente caso de uso permite gestionar reclamos por parte de los  propietarios para reparaciones del inmueble. |
| Flujo Principal | 1. Verificar si la persona que hace el reclamo es propietario. |
| Flujo Alternativo | ---------- |
| Post-Condición | Pedido de reparación. |

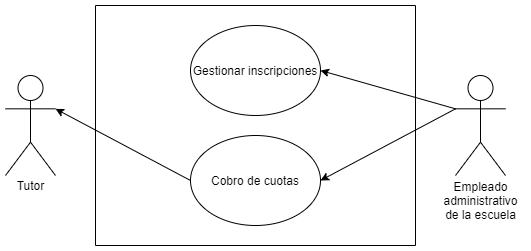
1. Requerimientos no funcionales

* El sistema deberá contar con un nivel alto de seguridad sobre la información de los propietarios y los pagos que se encuentre registrados o pendientes.
* El acceso al sistema no pueda ser provocada por nadie que no tenga los permisos necesarios y suficientes.
* La interfaz deberá ser simple, intuitiva y de fácil uso para que cualquier tipo de usuario pueda utilizar el sistema de forma correcta y rápida.
* El desempeño del sistema debe ser ágil, rápido y con una buena performance, de manera tal que la usabilidad pueda ser fluida y eficiente.
* Requerimientos de hardware:
  + Disco HDD: 1 (un) disco SATA 1 TB
  + Procesador: AMD A8
  + Memoria: 4 GB DDR3
  + Puerto USB: 2 (dos) de acceso frontal.
* Enlace de acceso a internet.

# ***Caso Práctico N°12: Aplicando Técnicas de modelado: Caso de Uso.***

1. Requisitos funcionales y diagrama de casos de uso.

* Gestión de inscripciones
* Cobro de cuota





|  |  |
| --- | --- |
|  | Gestión de inscripciones |
| Caso de Uso | Empleado administrativo de la escuela |
| Nombre | Gestión de inscripciones |
| Número | 1 |
| Tipo | Primario |
| Pre-Condición | Persona que se quiera inscribir en el colegio. |
| Actor | Empleado administrativo de la escuela |
| Descripción | El presente caso de uso permite realizar inscripciones. |
| Flujo Principal | 1. Verificar si tiene algún hermano en la institución o si se   Inscriben varios niños a la vez.   1. Verificar si se elige una actividad especial extra. 2. Registro de actividades seleccionadas. 3. Calcular el total de la cuota. 4. Pago de la primera cuota. 5. Generación de ticket. |
| Flujo Alternativo | 1. Aplicar un descuento del 50% |
| Post-Condición | Inscripción realizada + Ticket generado. |

|  |  |
| --- | --- |
|  | Cobro de cuota |
| Caso de Uso | Empleado administrativo de la escuela |
| Nombre | Cobro de cuota |
| Número | 2 |
| Tipo | Primario |
| Pre-Condición | Día 10 de cada mes |
| Actor | Empleado administrativo de la escuela |
| Descripción | El presente caso de uso permite realizar el cobro correspondiente a  las cuotas. |
| Flujo Principal | 1. Verificar regularidad de pago del alumno y cuotas pendientes. 2. Registrar pago. 3. Generar ticket. |
| Flujo Alternativo | 1. Si debe alguna cuota se da la posibilidad de un plan de pago. |
| Post-Condición | Cuota cobrada + Ticket generado. |

1. Requerimientos no funcionales

* Pool de 7(siete) direcciones IP públicas.
* Procesador: Dual Core Intel® Xeon TM serie 3000 Bus frontal de 2.40 GHz, 1066 MHz, caché de 4 MB
* El sistema deberá contar con un nivel alto de seguridad sobre la información de las prestaciones y los pagos de los clientes y que el acceso al sistema no pueda ser provocada por nadie que no tenga los permisos necesarios y suficientes.
* El sistema es compatible solo con sistema operativo Windows.
* Enlace de acceso a internet.

# ***Caso Práctico N°13: Aplicando Técnicas de Modelado: Historias de Usuario***

Caso Inmobiliaria Norte

* **Como** empleado de inmobiliaria **quiero** tener acceso al listado emitido por el banco con los pagos de impuestos del edificio **para** incluir los gastos en las liquidaciones.
* **Como** empleado de inmobiliaria **quiero** enviar una petición de reparación a la empresa correspondiente **para** registrar un pedido de reparación.
* **Como** empleado de inmobiliaria **quiero** gestionar pagos de reparación **para** saber los gastos efectuados por la empresa correspondiente.

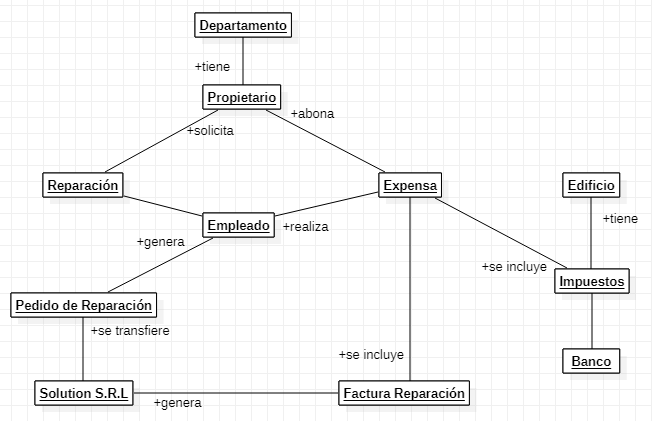
# ***Caso Práctico N°14: Aplicando Técnicas de Modelado: Historias de Usuario***

Caso Escuela Siglo XXI

* **Como** empleado administrativo de la escuela **quiero** saber los distintos niveles de educación **para** la gestión de inscripciones.
* **Como** empleado administrativo de la escuela **quiero** saber si la persona que se está inscribiendo tiene hermanos **para** aplicar el descuento correspondiente.
* **Como** empleado administrativo de la escuela **quiero** saber si el alumno tiene deudas **para** agregar lo adeudado en la cuota actual.

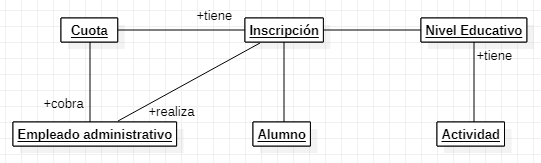
# ***Caso Práctico N°15: Comprendiendo el dominio de la aplicación: Modelo de dominio.***

Caso Inmobiliaria Norte



# ***Caso Práctico N°16: Comprendiendo el dominio de la aplicación: Modelo de dominio.***

Caso Escuela Siglo XXI



# ***Caso Práctico N°17: Investigando las Metodologías Ágiles: Sus aportes.***

* El Método Kanban es un medio para diseñar, administrar y mejorar los sistemas de flujo para el trabajo de conocimiento. También permite a las organizaciones comenzar con su flujo de trabajo existente e impulsar la evolución.
* El método Waterfall también denominado “cascada”, es el método que se ha utilizado tradicionalmente. Consiste en desarrollar un proyecto de forma secuencial, comenzando con las fases de análisis y diseño y terminando con las de testeo y puesta en producción.
* El método DevOps permite que los roles que antes estaban aislados se coordinen y colaboren para producir productos mejores y más confiables. Al adoptar una cultura de DevOps junto con prácticas y herramientas de DevOps, los equipos adquieren la capacidad de responder mejor a las necesidades de los clientes, aumentar la confianza en las aplicaciones que crean y alcanzar los objetivos empresariales en menos tiempo.
* El método Extreme Programming es un software de desarrollo ágil que apunta a la producción de software de alta calidad y la alta calidad de vida para el equipo de desarrollo.

# ***Caso Práctico N°18: Aplicando Técnicas de estimación: Puntos de caso de uso.***

Caso Inmobiliaria Norte

* **Caso de uso: Liquidación de expensas**
  + Transacciones: 3
  + Tipo: Simple (5)
  + Tipo de actor: Complejo (3)
* **Caso de uso: Generar pedido de reparación**
  + Transacciones: 1
  + Tipo: Simple (5)
  + Tipo de actor: Complejo (3)
* **Caso de uso: Ingresar datos bancarios**
  + Transacciones: 2
  + Tipo: Simple (5)
  + Tipo de actor: Complejo (3)
* **Caso de uso: Gestión pago de reparación**
  + Transacciones: 2
  + Tipo: Simple (5)
  + Tipo de actor: Complejo (3)

Peso total de los casos de uso sin ajustar:

|  |  |  |
| --- | --- | --- |
| Tipo de Caso de Uso | Peso | Cantidad |
| Simple | 5 | 4 |
| Medio | 10 | 0 |
| Complejo | 15 | 0 |

UUCW = 5 \* 4 + 10 \* 0 + 15 \* 0 = **20**

Peso total de los actores:

|  |  |  |
| --- | --- | --- |
| Tipo de Actor | Peso | Cantidad |
| Simple | 1 | 0 |
| Medio | 2 | 0 |
| Complejo | 3 | 4 |

UAW = 3 \* 4 = **12**

Puntos de casos de uso sin ajustar:

UUCP = UUCW + UAW

UUCP = 20 + 12 = **32**

Factor de complejidad técnica:

|  |  |  |  |
| --- | --- | --- | --- |
| Factor | Descripción | Peso | Valor |
| T1 | Sistema distribuido | 2 | 2 |
| T2 | Objetivos de performance o tiempo de respuesta | 1 | 4 |
| T3 | Eficiencia del usuario final | 1 | 3 |
| T4 | Procesamiento interno complejo | 1 | 3 |
| T5 | El código debe ser reutilizable | 1 | 4 |
| T6 | Facilidad de instalación | 0.5 | 5 |
| T7 | Facilidad de uso | 0.5 | 5 |
| T8 | Portabilidad | 2 | 0 |
| T9 | Facilidad de cambio | 1 | 3 |
| T10 | Concurrencia | 1 | 4 |
| T11 | Incluye objetivos especiales de seguridad | 1 | 3 |
| T12 | Provee acceso directo a terceras partes | 1 | 0 |
| T13 | Se requieren facilidades especiales de entrenamiento a usuarios | 1 | 1 |

TF = 0.6 + (0.01 \* 33) = 0.6 + 0.33 = **0.93**

Factor del entorno:

|  |  |  |  |
| --- | --- | --- | --- |
| Factor | Descripción | Peso | Valor |
| E1 | Familiaridad con el modelo de proyecto utilizado | 1.5 | 4 |
| E2 | Experiencia en la aplicación | 0.5 | 3 |
| E3 | Experiencia en orientación a objetos | 1 | 5 |
| E4 | Capacidad del analista líder | 0.5 | 3 |
| E5 | Motivación | 1 | 2 |
| E6 | Estabilidad de los requerimientos | 2 | 3 |
| E7 | Personal Part-Time | -1 | 0 |
| E8 | Dificultad del lenguaje de programación | -1 | 0 |

EF = 1.4 + (-0.03 \* 22) = 1.4 – 0.66 = **0.74**

Puntos de casos de uso ajustados:

AUCP = UUCP \* TF \* EF

AUCP = 32 \* 1.02 \* 0.93 = **30.35**

Esfuerzo del proceso de desarrollo:

E = AUCP \* CF

E = 30.35 \* 20

**E = 607 HH**

# ***Caso Práctico N°19: Aplicando Técnicas de estimación: Puntos de caso de uso.***

Caso Escuela Siglo XXI

* **Caso de uso: Gestionar Inscripciones**
  + Transacciones: 3
  + Tipo: Simple (5)
  + Tipo de actor: Complejo (3)
* **Caso de uso: Cobro de cuotas**
  + Transacciones: 2
  + Tipo: Simple (5)
  + Tipo de actor: Complejo (3)

Peso total de los casos de uso sin ajustar:

|  |  |  |
| --- | --- | --- |
| Tipo de Caso de Uso | Peso | Cantidad |
| Simple | 5 | 2 |
| Medio | 10 | 0 |
| Complejo | 15 | 0 |

UUCW = 5 \* 2 + 10 \* 0 + 15 \* 0 = **10**

Peso total de los actores:

|  |  |  |
| --- | --- | --- |
| Tipo de Actor | Peso | Cantidad |
| Simple | 1 | 0 |
| Medio | 2 | 0 |
| Complejo | 3 | 2 |

UAW = 3 \* 4 = **6**

Puntos de casos de uso sin ajustar:

UUCP = UUCW + UAW

UUCP = 10 + 6 = **16**

Factor de complejidad técnica:

|  |  |  |  |
| --- | --- | --- | --- |
| Factor | Descripción | Peso | Valor |
| T1 | Sistema distribuido | 2 | 3 |
| T2 | Objetivos de performance o tiempo de respuesta | 1 | 2 |
| T3 | Eficiencia del usuario final | 1 | 2 |
| T4 | Procesamiento interno complejo | 1 | 2 |
| T5 | El código debe ser reutilizable | 1 | 5 |
| T6 | Facilidad de instalación | 0.5 | 5 |
| T7 | Facilidad de uso | 0.5 | 5 |
| T8 | Portabilidad | 2 | 3 |
| T9 | Facilidad de cambio | 1 | 2 |
| T10 | Concurrencia | 1 | 2 |
| T11 | Incluye objetivos especiales de seguridad | 1 | 3 |
| T12 | Provee acceso directo a terceras partes | 1 | 0 |
| T13 | Se requieren facilidades especiales de entrenamiento a usuarios | 1 | 0 |

TF = 0.6 + (0.01 \* 35) = 0.6 + 0.35 = **0.95**

Factor del entorno:

|  |  |  |  |
| --- | --- | --- | --- |
| Factor | Descripción | Peso | Valor |
| E1 | Familiaridad con el modelo de proyecto utilizado | 1.5 | 4 |
| E2 | Experiencia en la aplicación | 0.5 | 4 |
| E3 | Experiencia en orientación a objetos | 1 | 3 |
| E4 | Capacidad del analista líder | 0.5 | 2 |
| E5 | Motivación | 1 | 2 |
| E6 | Estabilidad de los requerimientos | 2 | 2 |
| E7 | Personal Part-Time | -1 | 2 |
| E8 | Dificultad del lenguaje de programación | -1 | 0 |

EF = 1.4 + (-0.03 \* 16) = 1.4 – 0.48 = **0.92**

Puntos de casos de uso ajustados:

AUCP = UUCP \* TF \* EF

AUCP = 16 \* 0.95 \* 0.92 = **13.98**

Esfuerzo del proceso de desarrollo:

E = AUCP \* CF

E = 13.98 \* 20

**E = 279.6 HH**

# ***Caso Práctico N°20: Lectura de un artículo de un Congreso Nacional: Clasificación de métricas***

1. ¿A qué se define métrica y a qué se define indicador? Mencionar los autores de dichas definiciones.

Según Pressman las métricas son escalas de unidades sobre las cuales puede medirse un atributo cuantificable.

Lascano define indicador como una métrica o la combinación de varias de ellas. Este indicador proporciona información objetiva que, sumados a criterios de decisión definidos, permiten a los participantes de un proyecto medir la ejecución y realizar ajustes tanto en el producto como en los procesos que se emplean.

1. Enumerar y explicar la clasificación de métricas según Basso y Kan.

En Kan y en Basso se propone una de clasificación de métricas, la cual se basa en el contexto o

dominio de aplicación y las características o atributos del software. Dicha clasificación define:

* **Métricas del producto:** describen características del producto tales como tamaño, complejidad, características de diseño, rendimiento y nivel de calidad.
* **Métricas del proceso:** pueden ser utilizadas para mejorar el proceso de desarrollo y mantenimiento del software. Algunos ejemplos son: la efectividad de la remoción de defectos durante el desarrollo y el tiempo de respuesta en el proceso de corrección de defectos.
* **Métricas del proyecto:** describen las características y ejecución del proyecto. Algunos ejemplos son: el número de desarrolladores de software, el comportamiento del personal durante el ciclo de vida del proyecto, el costo, el cronograma y la productividad.

1. Enumerar y explicar la clasificación de métricas según la ISO/IEC 9126.

Según la norma ISO/IEC 9126 las métricas se pueden clasificar en tres categorías, según sea su

naturaleza:

* **Métricas básicas:** son métricas que se obtienen directamente del análisis del código o la ejecución del software. No involucra ningún otro atributo ni depende de otras métricas. Estas métricas se denominan directas. Entre las métricas básicas se tiene la cantidad de líneas de código del programa o de cada módulo, la cantidad de horas de desarrollo, la cantidad de fuentes de datos o tablas a utilizar, la cantidad de atributos y registros de una tabla, entre otras.
* **Métricas de agregación:** son métricas compuestas a partir de un conjunto definido de métricas básicas (o directas), generalmente mediante una suma ponderada.
* **Métricas derivadas:** son métricas compuestas por una función de cálculo matemático, que utiliza como variables de entrada el valor de otras métricas. Estas métricas se denominan indirectas. Entre las métricas derivadas se tiene la cantidad de líneas de código producidas por hora y por persona, el porcentaje de completitud del proyecto, el tamaño promedio de los módulos del software, el tiempo promedio que una persona dedica a corregir los defectos de un módulo, entre otras.

1. Enunciar la clasificación de métricas para el proceso de implantación de sistemas informáticos propuesta por los autores del artículo.

Para la propuesta de métricas del proceso de implantación, se definió una clasificación basada en la

propuesta de ISO/IEC 9126 y la propuesta definida por Basso y Kan. De la última

clasificación se consideran solo las categorías: métricas del proceso y métricas del producto. Se consideran las métricas del proceso debido a que resulta de interés las tareas, actividades y roles que comprenden al proceso de implantación de sistemas informáticos. La métrica del producto se la considera por la necesidad de contemplar características como la complejidad del producto software a instalar, los requisitos de instalación para el producto software, la integración con la infraestructura del cliente, entre otras; estas tendrán impacto en el proceso de implantación estudiado.

1. En el artículo se mencionan diferentes métodos de estimación, enumerarlos.

Karner: Cálculo del esfuerzo requerido para la implantación del software.

Dentro de este conjunto se encuentran métodos como COCOMO y COCOMO II de Boehm, SLIM de Putnam, y Puntos de Casos de Uso propuesto por Karner.

# ***Caso Práctico N°21: Aplicando Técnicas de estimación: Planning Pocker.***

Caso Inmobiliaria Norte

|  |  |
| --- | --- |
| **Puntos de historias estimados** | |
| **Funcionalidad** | **Puntos de historias estimados** |
| Liquidación de expensas | 20 |
| Generar pedido de reparación | 8 |
| Ingresar datos bancarios | 3 |
| Gestión pago de reparación | 13 |
| **Puntos de historias de usuarios totales** | **44** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cantidad**  **Desarrolladores** | **Días por sprint** | **Puntos de historias por sprint** | **Cantidad de sprints** | **Días totales de desarrollos** |
| 4 | 15 | 32 | 1,375 | 20,625 |

# ***Caso Práctico N°22: Aplicando Técnicas de estimación: Planning Pocker.***

Caso Escuela Siglo XXI

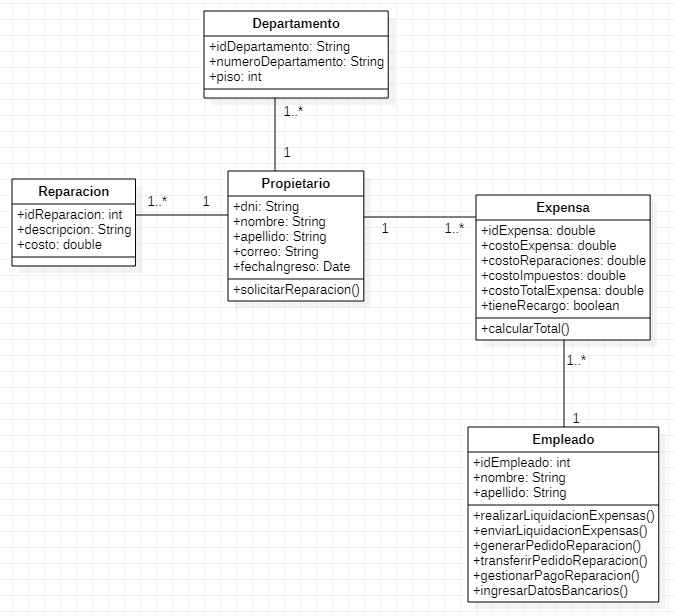
|  |  |
| --- | --- |
| **Puntos de historias estimados** | |
| **Funcionalidad** | **Puntos de historias estimados** |
| Gestionar Inscripciones | 20 |
| Cobro de cuotas | 13 |
| **Puntos de historias de usuarios totales** | **33** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cantidad**  **Desarrolladores** | **Días por sprint** | **Puntos de historias por sprint** | **Cantidad de sprints** | **Días totales de desarrollos** |
| 4 | 15 | 32 | 1,031 | 15,465 |

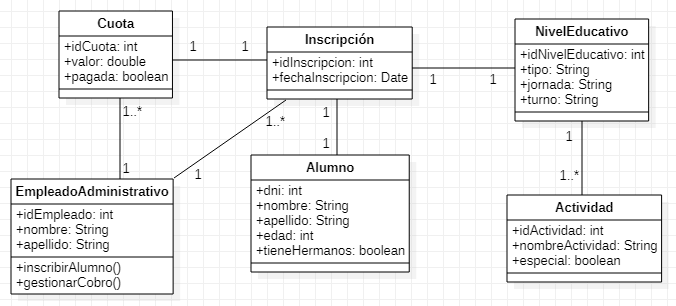
# ***Caso Práctico N°23. Comparando Técnicas de Estimación: Puntos de casos de uso & Puntos de historias de usuarios.***

|  |  |
| --- | --- |
| Puntos de casos de uso | Puntos de historias de usuario |
| Se asigna peso a cada caso de uso y a cada actor | Los desarrolladores van dándole el punto por nivel de dificultad a cada historia de usuario |
| Se evalúa la complejidad técnica del sistema | Esta relacionada directamente con las funcionalidades a ser desarrolladas y no con el tiempo de desarrollo requerido. |
| Se evalúa el entorno donde será desarrollada la aplicación |  |

# **Caso Práctico N°24: Diagrama estructural del sistema “INMOBILIARIA NORTE”: Diagrama de clases.**

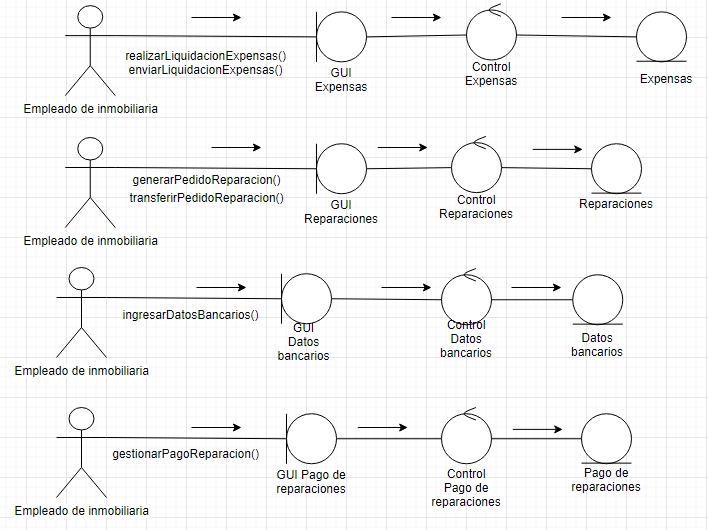


# ***Caso Práctico N°25: Diagrama estructural del sistema “ESCUELA SIGLO XXI”: Diagrama de clases.***



# ***Caso Práctico N°26: Construyendo el modelo de análisis - Diagrama de comunicación.***

Caso Inmobiliaria Norte



# ***Caso Práctico N°27: Construyendo el modelo de análisis - Diagrama de comunicación.***

Caso Escuela Siglo XXI

