

UNIVERSIDAD TECNOLÓGICA DE EL SALVADOR
FACULTAD DE INFORMÁTICA Y CIENCIAS APLICADAS
ESCUELA DE INFORMÁTICA



ESTANDARES DE PROGRAMACIÓN

UNIDAD: III

FACILITADOR: ING. EDWIN ALBERTO CALLEJAS

NOMBRE DE LA TAREA: PLAN DE PRUEBAS

ESTUDIANTES:	CARNET:	PART.:
ARCE AGUIRRE, JASON ALEXANDER	25-0129-2017	100%
CASTILLO ALFARO, GABRIEL ALEJANDRO	25-3339-2005	100%
CORDERO HERNANDEZ, KATHERINE ELIZABETH	25-1461-2017	100%
LOVATO HUEZO, KEVIN ARNOLDO	25-1642-2016	
MEJIA ORELLANA, DONOVAN ERNESTO	25-1318-2014	100%
PORTILLO DELEON MIGUEL EDUARDO	25-1596-2013	100%
SALAZAR MARTINEZ, JONATHAN OSWALDO	25-6019-2016	100%

INDICE

INTRODUCCION	3
OBJETIVOS	4
GENERAL	4
ESPECIFICOS	4
PLAN DE PRUEBAS	5
METODOLOGÍA	5
INSTRUMENTOS PARA ADMINISTRAR EL PLAN DE PRUEBAS	15
ACTIVIDADES	15
CRONOGRAMA DE ACTIVIDADES	31
CASOS DE PRUEBAS	33
BITACORA DE PRUEBAS	39
PRESUPUESTO	40
ORGANIGRAMA	
CONCLUSIONES	

INTRODUCCION

Las organizaciones requieren transformar su negocio al siguiente nivel y avanzar a software más avanzado y especializado para beneficiarse de las tecnologías de vanguardia e renovar los modelos de negocio vigentes, para ello las empresas deben adaptarse y cambiar rápidamente desde adentro la forma en que desarrollan sus productos y servicios, para ello, el software es la clave de su transformación.

Otro factor importante para el diseño de los procesos es la necesidad de mitigar los riesgos del desarrollo de software, una parte considerable del riesgo del software se basa en procesos, por ejemplo, ha habido varios incidentes que podrían haberse evitado con estándares y herramientas de codificación adecuados, aunque estos estándares y herramientas están ampliamente disponibles, no se aplican o no se aplican adecuadamente en muchas situaciones debido a que esto normalmente se debe a la forma en que se organiza el trabajo, la seguridad física y lógica de los sistemas y a las personas con las que se lleva a cabo dicho trabajo. Es un problema del proceso de software y que las empresas deben encontrar, son formas de garantizar que los modelos de proceso se definan correctamente y que además se apliquen de manera adecuada

OBJETIVOS

GENERAL

Proporcionar un diseño al nivel del cambiante entorno técnico y de mercado proporcionando información que ayude a construir de forma óptima y eficaz el software objetivo y enfatizar la necesidad considerada en la evolución del proceso de desarrollo de software como un medio importante para la empresa.

ESPECIFICOS

Organizar el software de una organización y centrar en el diseño, desarrollo, gestión, gobernanza y aplicación del proceso de desarrollo de software en procesos que estén alineados con los objetivos comerciales de la empresa, como la expansión a nuevos dominios o el paso a la producción global.

Generar por medio de un entorno ágil el Sistema Transaccional de Activo Fijo para que pueda su construcción pueda ser gestionada por todos los miembros del equipo.

PLAN DE PRUEBAS

METODOLOGÍA

La metodología o procedimiento que se desea proponer en el presente proyecto de grado contiene los pasos y etapas que buscan garantizar la calidad funcional en un producto de software, siguiendo como lineamiento la idea de detección temprana de problemas del producto a nivel funcional haciéndose necesario que la metodología a proponer vaya alineada con el ciclo de desarrollo del software, desde su etapa más temprana hasta el final de este.

Este procedimiento se encuentra basado en el ciclo de desarrollo del software basado en componentes y en sus mejores prácticas, tiene un futuro muy prometedor para el análisis, implementación y documentación de aplicativos, lo cual queremos fortalecer aún más con esta propuesta metodológica de pruebas funcionales.

A continuación, se pretende proponer la manera como el procedimiento propuesto para realizar pruebas funcionales de software se integra y se adapta a cada uno de los procesos que se deben llevar a cabo por la metodología de desarrollo de software basado en componentes a través de cada una de las fases, diseño de la arquitectura, especificación y búsqueda de componentes, adaptabilidad de componentes e integración de componentes.

Fase de Diseño de la Arquitectura.

La fase de diseño describe la arquitectura del software. Describe cada uno de los componentes que requieren tal estructura y cómo esos componentes se interconectan para interactuar, así como el ambiente y los principios que orientan su diseño. El grupo de desarrollo debe, a partir de esta arquitectura, determinar cuáles componentes se pueden reutilizar y cuáles requieren ser desarrollados en su totalidad. Los componentes reutilizados deben ser adaptados, para satisfacer los requerimientos del sistema, mientras que los componentes nuevos, deben ser diseñados, codificados y probados separadamente durante la fase de implementación. En esta fase de diseño de la arquitectura es posible redefinir el alcance del proyecto con los patrocinadores, identificar los riesgos asociados al proyecto.

Etapas de la Metodología: Planeación de Pruebas.

Esta etapa comienza con la recepción de la documentación y componentes del aplicativo, donde el cliente si es que lo hay, entrega toda la documentación necesaria para poder planificar y diseñar la prueba, se realiza la lectura y entendimiento de la documentación, así como una reunión con el analista para revisar los temas que no se hayan entendido de la documentación de la aplicación. Además, esta etapa de Planeación de Pruebas permite conocer el alcance de las pruebas definiendo aspectos como las entradas de pruebas (requerimientos para pruebas), la valoración de riesgos, las estrategias, los recursos necesarios, el cronograma y el plan de pruebas. Los resultados de la etapa son el plan de pruebas y el cronograma de pruebas, documentos que contienen todos los aspectos antes descritos.

Fase de Especificación y Búsqueda de Componentes.

La especificación y búsqueda de componentes consta de tres etapas. Primero, la identificación de componentes donde se genera un conjunto inicial de interfaces y especificaciones de componentes conectados entre sí en una primera aproximación a la arquitectura. Segundo, la interacción entre componentes donde se decide como los componentes van a trabajar de modo coordinado para conseguir la funcionalidad deseada. Y finalmente, la especificación de componentes donde se definen las especificaciones concretas de cada uno de los componentes utilizados. Al final se deben haber generado las especificaciones de los componentes.

Fase de Adaptabilidad de Componentes.

El propósito de esta fase es adaptar el componente o los componentes de manera tal que cumplan con la funcionalidad especificada, esta adaptación debe realizarse de manera independiente por cada una de las funcionalidades que se requieren, para ello se deben clarificar en primer lugar los requerimientos pendientes, administrar el cambio de los componentes seleccionados, y ejecutar el plan de administración de recursos y mejoras en el proceso de desarrollo para el proyecto.

Etapas de la Metodología: Ejecución de Casos de Pruebas.

En esta etapa se realiza la ejecución de los casos de prueba que se planearon y diseñaron, los cuales son ejecutados a

través de la interfaz de usuario y se conocen como pruebas de caja negra. La ejecución de los casos de prueba se debe realizar siguiendo la descripción de cada caso de prueba, haciendo énfasis tanto en aquellos casos de prueba cuya funcionalidad haya sido desarrollada con componentes reutilizados (estos casos debieron ser marcados con una 'R' desde la anterior etapa), así como en los componentes desarrollados cuyas pruebas funcionales realizadas anteriormente en otro proyecto hayan presentado algún tipo de inconveniente, además se debe registrar el resultado arrojado al ejecutar cada caso de prueba, así como los problemas encontrados para cada componente. En esta etapa además se debe llevar a cabo la ejecución de los casos de prueba realizados para la integración de los componentes que da como resultado el aplicativo final, así mismo como el registro de los problemas encontrados.

Fase de Integración de Componentes.

La integración de componentes o elementos de software no es un concepto nuevo. Desde hace muchos años el concepto de integración de código de dos o más componentes ha sido crítico en el éxito de las organizaciones de desarrollo lo que ha llevado a la integración de software a límites mucho más extensos. La integración de este tipo de componentes ya sean reutilizables o desarrollados, por su misma naturaleza ofrece casi todas las ventajas para construir un aplicativo de calidad, aplicativos que estén a la medida del cliente mediante la integración de elementos que más se adaptan a sus necesidades evitando en la medida de lo posible la modificación de las aplicaciones a integrar haciendo que los componentes que fueron identificados para el desarrollo interactúen entre sí de tal modo que conjuntamente logren cumplir con la funcionalidad propuesta del aplicativo especificando exactamente la manera como estos se complementan entre sí.

Sin embargo, no son solo ventajas las que ofrece la integración de componentes, se debe mencionar además que este modo de desarrollo tiene también sus desventajas y propios riesgos. El punto crítico en la integración de componentes es precisamente esa, la integración, lograr integrar los componentes usados de manera tal que logren cumplir con la funcionalidad propuesta en ocasiones no es tarea sencilla, esto precisamente no se garantiza validando que cada uno de

los componentes funcione individualmente como debiera, el resultado del trabajo conjunto de todos los componentes a pesar de que todos hayan sido probados por separado y funcionen como debieran, es inesperado y pueden presentarse incompatibilidades.

Etapas de la Metodología: Evaluación de Pruebas.

Esta etapa entrega un documento de certificación y el resumen de pruebas a partir de los registros de ejecución de las pruebas y el reporte de problemas. La evaluación de la ejecución de pruebas funcionales permite determinar si los criterios de completitud de los casos de prueba han terminado a satisfacción o si se han encontrado problemas o se han detenido casos de prueba que se estaban ejecutando, determinando así si hubo una terminación normal o anormal. Una terminación normal, es aquella donde todos los casos de prueba fueron ejecutados y todos los datos son válidos. Una terminación anormal, es donde uno o varios casos de prueba, no se pudieron completar o cuando hubo una falla del sistema y se debió detener la ejecución de las pruebas. En este caso toda la ejecución de las pruebas se debe volver a realizar. De lo contrario si hubo una terminación normal, se debe evaluar que el aplicativo en su conjunto funcione como se especificó inicialmente, es decir se evalúa que el hecho de haber probado las funcionalidades con sus componentes por separado se vea reflejado en la integración final de los componentes que cumplan con la funcionalidad principal para la cual fue pensado el aplicativo en un primer lugar.

Para que se genere mayor facilidad en el proceso de pruebas, los problemas son reportados en una aplicación que permita el seguimiento de estos problemas y permita obtener un registro e historial de los mismos.

Dependiendo de la gravedad del problema y los criterios de salida de la prueba, se debe dar prioridad a estos para una acción correctiva. Los de menor gravedad deberán ser documentados y manejarse según los procedimientos de informes y seguimiento del problema para tener la seguridad de una corrección posterior.

La idea es conformar un comité de cambios el cual se debe reunir periódicamente (máximo cada semana) con el fin de asignar los problemas al desarrollador o grupo de desarrolladores, definir prioridad, severidad, nivel de impacto y

establecer compromisos concretos de solución.

Se evalúan los resultados de las pruebas funcionales, analizando las incidencias recibidas y comprobando que se han llevado a cabo todos los casos de prueba establecidos en el Plan de Pruebas.

La evaluación consiste en:

- Comparar los resultados obtenidos con los esperados.
- Identificar el origen de cada problema para poder remitirlo a quien corresponda y determinar qué acciones o medidas correctivas se deben llevar a cabo para resolverlo de forma satisfactoria.
- Indicar qué pruebas se deben volver a realizar, o si será necesario contemplar nuevos casos de prueba.

Durante la etapa de evaluación de pruebas se obtienen las métricas de todo el proceso de pruebas del proyecto.

Prácticas para que las pruebas sean exitosas

En general, es una buena práctica contar con el manual de uso del aplicativo a probar, y en este caso, para pruebas de software basado en componentes sería aún mejor si se puede contar con el manual de uso propio de cada componente usado durante el desarrollo del aplicativo, de este modo se puede obtener una idea más concreta de las funcionalidades clave del aplicativo.

En la medida de lo posible se debe contar con un deck de pruebas elaborado por el desarrollador del aplicativo. El deck básicamente son pruebas unitarias realizadas por el desarrollador donde se reflejan las evidencias del test sobre la prueba, es decir la evidencia sobre la funcionalidad principal del aplicativo, esto con el fin de que el ejecutor de pruebas quién es un poco más ajeno al aplicativo le sea más fácil entender la funcionalidad principal y a partir de allí elaborar los demás casos de prueba para el resto de las funcionalidades a evaluar.

Se deben evitar los casos desechables, es decir, los no documentados ni diseñados con cuidado, ya que suele ser necesario probar muchas veces el software y por tanto hay que tener claro qué funciona y qué no.

Indagar siempre en caso de duda. Es muy normal que al principio de las pruebas a realizar se presenten inquietudes

respecto al aplicativo. En este caso no dudar de preguntar para resolver dichas inquietudes, por lo general la persona más indicada para aclarar este tipo de preguntas es el usuario final quién conoce y sabe cómo debe funcionar correctamente el aplicativo.

Nunca asumir o suponer durante el proceso de pruebas. Es mucho más recomendable preguntar en caso de dudar en lugar de hacer suposiciones.

La experiencia parece indicar que donde hay un fallo hay otros, es decir, la probabilidad de descubrir nuevos fallos en una parte del software es proporcional al número de fallo ya descubierto.

Nunca pensar que una funcionalidad es muy obvia para ser probada. Las mejores pruebas realizadas sobre un aplicativo son aquellas que evalúan incluso las funcionalidades más obvias y no suponen ni asumen ningún resultado antes de ser realizadas.

No se puede ser confiado del aplicativo, por qué por muy remoto que parezca, esa funcionalidad que pensamos que no debe fallar y por tanto no le prestamos la suficiente atención a las pruebas que le realizaremos puede presentar resultados inesperados, ninguna funcionalidad está exenta de este tipo de sucesos.

Para realizar testing efectivo el equipo desarrollador de software debe realizar revisiones técnicas formales (RTF) con el fin de eliminar errores antes del inicio de la etapa de pruebas.

El testing empieza en un nivel bajo (de componentes) y trabaja “hacia afuera” con el fin de integrar todo el sistema. De lo menos hacia lo más. Toda estrategia de pruebas de software debe incluir pruebas de bajo nivel, necesarias para verificar que pequeños segmentos de código estén correctamente implementados (unidades lógicas), como también debe incluir pruebas de alto nivel que validen la funcionalidad general del sistema.

Aseguramiento de la calidad del software

El aseguramiento de la calidad es un esfuerzo planeado para asegurar que el producto (software) cumpla con los criterios y los factores de calidad mencionados en el numeral anterior. Estas actividades y funciones no solo son

responsabilidad del equipo asegurador de calidad, sino que incluyen también al gerente/líder del proyecto, el personal del proyecto y los usuarios.

Los objetivos de la calidad del software se logran siguiendo el plan de aseguramiento de calidad de software, el cual establece los métodos que el proyecto debe emplear para asegurar que los documentos y/o productos generados y revisados en cada etapa, sean de alta de calidad.

El aseguramiento de la calidad del software es una estrategia adoptada por la gestión del riesgo. Considerar la calidad de software dentro de la gestión del riesgo es importante porque en muchas ocasiones la calidad tiene un alto costo en los proyectos de software. Se necesita evitar:

- Fallas frecuentes en la funcionalidad del software.
- Consecuencias secundarias de fallas en el software, como problemas financieros.
- Sistemas no disponibles cuando se requiere.
- Costosas mejoras en el software
- Altos costos en la detección y corrección de errores.

Estrategias de software

Si se considera el proceso de prueba de software desde un punto de vista procedimental, las pruebas consisten en una serie de pasos que se implementan de manera secuencial. Estos pasos empiezan desde la prueba de unidad, asegurando que funciona de manera apropiada como unidad; continúan con prueba de integración, que se realizan cuando se integran los paquetes y que considera problemas como doble verificación y construcción del programa; prueba de validación, primera prueba de alto nivel y se realiza cuando se ha integrado y probado el programa, en ella se debe evaluar los criterios de validación establecidos en la etapa de análisis de requisitos incluyendo requisitos funcionales, de

comportamiento y de desempeño; y por último se realiza la prueba de sistema, la cual queda por fuera de todos los límites del ingeniero de software, ya que cuando el producto está listo debe combinarse con otros elementos del sistema como hardware, personas, bases de datos, etc. En esta prueba se verifica que todos los elementos encajen apropiadamente y que se logre la función y el desempeño generales del sistema.

Identificar las funcionalidades de sistemas existentes que deben probarse

Se identificarán las funcionalidades existentes que estén siendo impactadas por el desarrollo de alguna forma, considerando todos los componentes afectados en todas las capas de la arquitectura de software.

Estas son las dos situaciones que se puede encontrar al identificar estas funcionalidades:

- Funcionalidades modificadas de cara al usuario: Por ejemplo, si una funcionalidad está siendo modificada agregando más pantallas o cambios a el flujo de proceso, debe ser incluida en el plan de pruebas de software.
- Funcionalidades modificadas en sus componentes internos: Son funcionalidades no modificadas de cara al usuario, manteniendo la misma interfaz gráfica y flujo de procesos, sin embargo, si se modifican componentes internos que comparten con otras funcionalidades del sistema, en las capas de lógica de negocio o acceso a datos.

Quienes pueden suministrar la información serán los Analistas de negocio o Arquitectos de software, familiarizados con el sistema informático implementado en entorno de producción.

Definir la estrategia de pruebas

Consiste en seleccionar cuáles son los tipos de pruebas de software que se deben realizar.

Se seguirá un marco de referencia para determinar los tipos de prueba, como por ejemplo los tipos de pruebas de software definidos por el ISTQB.

Pruebas funcionales:

Se determinarán los conjuntos de pruebas a realizar, correspondiente con cada funcionalidad nueva o existente que se

está modificando.

Se tienen distintos tipos de pruebas funcionales, por ejemplo, las pruebas de sistema (o pruebas integradas de sistemas), que se realizarán después que el equipo de desarrollo ha integrado los componentes de distintas capas.

Las pruebas funcionales son definidas por el ISTQB como pruebas basadas en especificación. Serán diseñadas usando técnicas de diseño de pruebas de caja negra.

Pruebas no funcionales:

Se define un conjunto de pruebas no funcionales para cada requisito de este tipo. Aquí se incluyen pruebas sobre el desempeño, tiempo de respuesta, mantenibilidad, pruebas de seguridad de software, entre otros aspectos, según la clasificación de requisitos no funcionales que se tienen en el proyecto.

Pruebas de caja blanca:

Según la estructura y arquitectura del software con la que se desarrolló.

Pruebas de regresión:

Se definirán sobre las funcionalidades modificadas en sus componentes internos.

Tipos de pruebas de software en metodologías ágiles.

Definir los criterios de inicio, aceptación y suspensión de pruebas

Criterios de aceptación o rechazo:

Para definir los criterios de aceptación o rechazo, se definirá el nivel de tolerancia a fallos de calidad. Si la tolerancia a fallos es muy baja puede definirse como criterio de aceptación que el 100% de los casos de prueba estén sin incidencias. Lograr este margen en todos los casos de prueba principales y casos borde será muy difícil, y podría comprometer los plazos del proyecto (incrementa los riesgos), pero asegura la calidad del producto.

En el caso de realizar un Soft Launch, o un mínimo producto viable, se podría definir como criterio de aceptación el 100% de los casos de prueba principales (considerados clave) y 20% de casos de prueba no principales (casos bordes).

Una vez logradas las condiciones, se darán por aceptadas las pruebas y el desarrollo de software.

Criterios de inicio o reanudación:

Se definirán las condiciones que deben cumplirse para dar inicio o reanudar las pruebas.

Para el caso de la reanudación las condiciones están relacionadas, se determina a partir de cuales criterios de suspensión se presentaron para detener las pruebas. Una vez que estás condiciones ya no existan (sean solventadas) se procede con la reanudación.

Criterios de suspensión:

Las condiciones van a depender de los acuerdos de nivel de servicio (SLAs) internos de la organización y también de los acuerdos establecidos en cada proyecto individual.

Se podrá observar un determinado porcentaje de casos fallidos que resulten en incidencias. Si la condición se cumple, se detienen las pruebas y se dedicará el personal a otras actividades.

Identificar los entornos (ambientes) requeridos

Posteriormente se definirán y documentarán las características de los entornos de Hardware y Software necesarios para realizar la ejecución de las pruebas de software.

Esta información se obtendrá a partir del equipo de desarrollo y de los arquitectos de software, quienes pueden suministrar los requisitos mínimos y óptimos para la operación del sistema.

Como mejor práctica, el ambiente de pruebas de software será lo más similar posible al ambiente de producción, sin embargo, no se asegura la exactitud en este ambiente debido a limitaciones de recursos (financieros). Por eso se estudiarán cuáles son los requisitos que aseguran un mínimo de confiabilidad de estas pruebas respecto al entorno de producción.

Además, en esta sección del plan de pruebas, también se definirán los requisitos de sistemas operativos, software y herramientas de las estaciones de trabajo de los Testers.

También se definirán los requisitos de hardware y software para los siguientes componentes:

- Herramienta de gestión de calidad de software.
- Herramientas para automatización de pruebas.
- Herramientas de BDD, TDD y Testing de Web Services).

INSTRUMENTOS PARA ADMINISTRAR EL PLAN DE PRUEBAS

Actividades

Actividad 1: Registro petición de servicio Estrategia de Pruebas

El Gestor de Proyecto o el Director Técnico de Proyecto deberá dar de alta la petición de servicio en la herramienta de gestión de servicio. En la petición se deberá indicar el proyecto y las características de la entrega. Además, deberá asegurar que la documentación del proyecto es accesible por el Equipo de Testing y está actualizada, con el fin de poder determinar los servicios que aplican a la entrega, así como el dimensionamiento de estos.

Tareas

- Definir las características de la entrega.
- Asegurar la actualización de la documentación necesaria.
- Dar de alta la petición de servicio en la herramienta de gestión de servicio.
- Responsable.
- Gestor de Proyecto / Director Técnico de Proyecto.

Productos

- Petición de servicio registrada en la herramienta de gestión de servicio.

Actividad 2: Definición del Alcance de las Pruebas

En función de la solución adoptada en el desarrollo del sistema de información, es posible que determinados niveles de pruebas sean especialmente críticos y otros no sean necesarios.

En esta tarea se especifican y justifican los niveles de pruebas a realizar, así como el marco general de planificación de cada nivel de prueba, según el siguiente esquema:

- Definición de los perfiles implicados en los distintos niveles de prueba.
- Planificación temporal.
- Criterios de verificación y aceptación de cada nivel de prueba.
- Definición, generación y mantenimiento de verificaciones y casos de prueba.
- Análisis y evaluación de los resultados de cada nivel de prueba.
- Productos para entregar como resultado de la ejecución de las pruebas.

Productos

De entrada

- Catálogo de Requisitos.
- Catálogo de Normas.
- Descripción General del Entorno Tecnológico.
- Especificación de Interfaz de Usuario.

En Análisis Estructurado:

- Contexto del Sistema.
- Modelo de Procesos.
- Modelo Lógico de Datos Normalizado.

En Análisis Orientado a Objeto:

- Modelo de Casos de Uso.
- Especificación de Casos de Uso.
- Descripción de Subsistemas de Análisis.
- Descripción de Interfaces entre Subsistemas.
- Modelo de Clases.
- Comportamiento de Clases.
- Análisis de la Realización de los Casos de Uso.

De salida

- Plan de Pruebas:
- Especificación de los Niveles de Pruebas.

Prácticas

- Sesiones de Trabajo

Participantes

- Jefe de proyecto
- Analistas
- Equipo de Soporte Técnico
- Usuarios Expertos

Actividad 3: Elaboración de propuesta de servicios de testing

El Jefe de Equipo de Testing deberá elaborar una propuesta que contenga los servicios más adecuados para la entrega. Para la elaboración de dicha propuesta se deberá tener en cuenta, además de la documentación aportada, la tecnología

del proyecto, y el estado, criticidad y urgencia de la entrega.

Tareas

- Analizar la documentación del proyecto.
- Determinar los servicios que aplican en función de la tecnología.
- Entre los servicios que se pueden aplicar, elegir aquellos servicios que aporten mayor valor teniendo en cuenta el estado y la criticidad de la entrega, pero sin olvidar su urgencia.
- Registrar la propuesta en la herramienta de gestión de servicio.
- Responsable.
- Jefe Equipo de Testing / Equipo de Testing.

Productos

- Registro de la propuesta de servicios en la herramienta de gestión del servicio.

Actividad 4: Definición de Requisitos del Entorno de Pruebas

El objetivo de esta tarea es la definición o recopilación de los requisitos relativos al entorno de pruebas, completando el plan de pruebas.

La realización de las pruebas aconseja disponer de un entorno de pruebas separado del entorno de desarrollo y del entorno de operación, garantizando cierta independencia y estabilidad en los datos y elementos a probar, de modo que los resultados obtenidos sean objetivamente representativos, punto especialmente crítico en pruebas de rendimiento.

No es objeto de métrica versión 3 en general, ni de esta tarea en particular, la especificación formal de entornos y procedimientos de pruebas en el ámbito de una instalación.

Independientemente de la existencia o no de dichos entornos, en esta tarea se inicia la definición de las especificaciones necesarias para la correcta ejecución de las distintas pruebas del sistema de información. Entre ellas tenemos las

siguientes:

- Requisitos básicos de hardware y software base: sistemas operativos, gestores de bases de datos, monitores de teleproceso, etc.
- Requisitos de configuración de entorno: librerías, bases de datos, ficheros, procesos, comunicaciones, necesidades de almacenamiento, configuración de accesos, etc.
- Herramientas auxiliares. Por ejemplo, de extracción de juegos de ensayo, análisis de rendimiento y calidad, etc.
- Procedimientos para la realización de pruebas y migración de elementos entre entornos.

Productos

De entrada

- Catálogo de Requisitos.
- Descripción General del Entorno Tecnológico.
- Plan de pruebas.

De salida

- Plan de pruebas:
- Definición de Requisitos del Entorno de Pruebas

Prácticas

- Sesiones de Trabajo
- Participantes
- Jefe de proyecto
- Analistas
- Equipo de Soporte Técnico

- Usuarios Expertos

Actividad 5: Especificación de pruebas individuales y grupales

En esta actividad se especifican los estándares en las pruebas individuales y grupales que se realizaron a los componentes del Sistema, una vez codificados con el objetivo de comprobar que su estructura era la correcta y que ajustara a la funcionalidad establecida, para ello se efectúa el correspondiente análisis y evaluación de los resultados.

Las pruebas al Sistema permiten encontrar posibles errores en la programación de los módulos, en las Base de datos, errores de verificación y otros que pudieron haberse generado en cada uno de los componentes a evaluar.

El estándar diseñado para las pruebas del Sistema informático involucra las siguientes observaciones:

Efectuar pruebas a los módulos en forma individual e integrada, con el propósito que éste resulte confiable funcional y en conformidad con las especificaciones establecidas.

Actividad 6: Definición de las Pruebas de Aceptación del Sistema

En esta tarea se realizará la especificación de las pruebas de aceptación del sistema, labor fundamental para que el usuario valide el sistema, como último paso, previo a la puesta en explotación.

Se insistirá, principalmente, en los criterios de aceptación del sistema que sirven de base para asegurar que satisface los requisitos exigidos.

Los criterios de aceptación serán definidos de forma clara, prestando especial atención a aspectos como:

- Procesos críticos del sistema.
- Rendimiento del sistema.
- Seguridad.
- Disponibilidad.

Productos

De entrada

- Catálogo de requisitos.
- Especificación de Interfaz de Usuario.
- Plan de Pruebas.

En Análisis Estructurado:

- Contexto del Sistema.
- Descripción de Interfaz con otros Sistemas.
- Modelo de Procesos.
- Modelo Lógico de Datos Normalizado.

En Análisis Orientado a Objetos:

- Modelo de Casos de Uso.
- Especificación de Casos de Uso.
- Descripción de Subsistemas de Análisis.
- Descripción de Interfaces entre Subsistemas.
- Modelo de Clases.
- Comportamiento de Clases.
- Análisis de la Realización de los Casos de Uso.

De salida

- Plan de Pruebas

Prácticas

- Sesiones de Trabajo

Participantes

- Jefe de Proyecto
- Analistas
- Equipo de Soporte Técnico
- Usuarios Expertos

Actividad 7: Validación de propuesta y acuerdo modelo propuesta aceptación / reentrega

El Gestor de Proyecto, el Jefe de Equipo de Explotación de Sistemas y el Jefe de Equipo de Testing valorarán si la propuesta de servicios es la adecuada. Además, decidirán el nivel de certificación y la exhaustividad de las pruebas para cerrar el modelo de propuesta de aceptación /reentrega que se pretende aplicar a esta entrega.

Tareas

- Decidir los servicios que finalmente serán aplicados a la entrega.
- Determinar el nivel de certificación con el que se va a valorar la entrega y la exhaustividad de cada prueba.
- Registrar esta información en la herramienta de gestión del servicio.

Responsable

Gestor de Proyecto / Director Técnico de Proyecto / Jefe Equipo de Explotación de Sistemas de Información / Equipo de Sistemas de Información / Jefe Equipo de Testing / Equipo de Testing

Productos

- Servicios de testing que se van a ejecutar para esta entrega registrados en la herramienta de gestión de servicio.
- Nivel de certificación y exhaustividad de cada servicio registrado en la herramienta de gestión de servicio.

Actividad 8: Determinar necesidades de personal y entrenamiento

Debe completarse previamente la estimación del esfuerzo de pruebas a partir del diseño de casos de prueba.

Para ello se buscará la respuesta a las siguientes preguntas:

¿Qué conocimientos de procesos de negocio se necesitan?

¿Qué sistemas se están probando y quienes tienen experiencia en su funcionamiento?

¿Se necesitan conocimientos específicos en pruebas de requisitos no funcionales? Por ejemplo, para pruebas de desempeño o estrés.

¿Cuáles herramientas de gestión de calidad de software se va a utilizar?

¿Se necesitan conocimientos en herramientas técnicas como Lenguajes de programación o herramientas de pruebas de webservices?

¿Se necesitan conocimientos en herramientas de pruebas automatizadas?

Requisitos de entrenamiento:

Transferencia de conocimientos en el sistema y su operación con el área de negocio.

Formación en metodologías de pruebas de software.

Entrenamiento en tipos de pruebas especializados (desempeño, estrés, arquitectura, caja blanca).

Formación en automatización de pruebas de software.

Actividad 9: Metodología de pruebas

Para poder obtener los resultados esperados de un sistema informático será necesario determinar métodos y técnicas para el desarrollo de las pruebas que nos ayudarán a verificar el buen o mal funcionamiento del sistema.

La prueba del software es un elemento crítico para la garantía de la calidad del software. El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado.

Los objetivos principales de realizar una prueba son:

- Detectar un error.

- Tener un buen caso de Prueba.
- Descubrir un error no descubierto antes.

A continuación, se presentan los mecanismos de prueba de software más conocidos:

- Prueba de Caja Negra

Los usuarios de un Sistema Informático tratarán el programa como una caja negra a la cual le introducen datos de entrada y obtienen de ella los datos de salida que se esperan.

El Método de la Caja Negra se centra en los requisitos fundamentales del software y permite obtener entradas que prueben todos los requisitos funcionales del programa. Con este tipo de pruebas se intentará encontrar:

- Funciones incorrecta o ausente.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación

Prueba de Caja de Cristal (o Caja Blanca)

El segundo enfoque en la selección de los datos de prueba inicia con la observación de que un programa difícilmente puede considerarse como probado por completo si su código contiene partes que nunca han sido ejecutadas

Actividad 10: Identificar los riesgos y definir planes de respuesta

Para el Software Testing, los riesgos por lo general están vinculados con factores como:

Posibles dificultades en la disponibilidad de entornos.

Pruebas que dependen de factores externos al proyecto y la organización.

Disponibilidad de personal con conocimientos especializados en alguna herramienta, o en la funcionalidad específica que

se está desarrollando.

Dependencias con otros proyectos.

Posibilidad que alguna premisa no se cumpla.

Para identificar los riesgos se enumerarán cada una de estas dependencias y por medio de mesas de trabajo y tormentas de ideas pensar en las posibilidades de que algo salga mal (u oportunidades para que salga bien).

Luego de la identificación, también se definirán planes de respuesta, los cuales serán específicos para cada situación particular y riesgo.

Actividad 11: Diseño de pruebas

- Pruebas de Unidad

La prueba de unidad se centrará en el módulo usando la descripción del diseño detallado como guía, se probarán los caminos de control importantes con el fin de descubrir errores dentro del ámbito del módulo. La prueba de unidad hace uso intensivo de las técnicas de prueba de caja blanca.

- Pruebas de Integración

Son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias. Consiste en realizar pruebas para verificar que un gran conjunto de partes de software funciona juntas.

- Pruebas de Aceptación

Estas pruebas las realizará el cliente. Son básicamente pruebas funcionales, sobre el Sistema completo, y buscan una cobertura de la especificación de requisitos y del manual del usuario

- Prueba general del Sistema

Se verifica que cada elemento encaje de forma adecuada y que se alcance la funcionalidad y el rendimiento del Sistema total. La prueba del Sistema está constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar

profundamente el sistema basado en computadora.

Actividad 12: Ejecución de pruebas

La ejecución de pruebas se basa en los ciclos o iteraciones de prueba, en los cuales los conjuntos de pruebas determinados y en busca de una estabilidad incremental del producto de software.

A continuación, se presentan las pruebas a realizar:

- Pruebas de Unidad

Las pruebas unitarias se hacen en cada uno de los módulos: Mercado, Contabilidad y Tesorería.

Resultados de la Prueba de Unidad:

Según las pruebas realizadas, se logra comprobar que el registro de activo fijo se realiza de una forma segura y correcta, ya que mediante validaciones y mensajes de advertencia no es posible la infiltración de información incorrecta. Así, la ejecución de las pruebas unitarias concluye de la forma información segura al Sistema.

- Pruebas de integración.

Actividad 13: Las pruebas de seguridad y control de acceso

Las pruebas de seguridad y control de acceso se centran en dos áreas claves de seguridad:

- Seguridad del sistema, incluyendo acceso a datos o Funciones de negocios.
- Seguridad del sistema, incluyendo ingresos y accesos remotos al sistema.

Nivel de seguridad de la aplicación

Verifica que un actor solo pueda acceder a las funciones y datos que su usuario tiene permitido.

Las pruebas de seguridad del sistema garantizan que solamente aquellos usuarios autorizados a acceder al sistema son capaces de ejecutar las funciones del sistema a través de los mecanismos apropiados.

Pruebas de seguridad y control de acceso

El objetivo es evaluar el funcionamiento correcto de los controles de seguridad del sistema para asegurar la integridad y confidencialidad de los datos.

Consideraciones de prueba

- Controles automáticos, incluyendo aquellos para edición de datos, chequeo de máquinas, errores del operador, acceso a datos elementales y archivos, acceso a funciones, auditoría, entre otros.
- Existencia de datos confidenciales en reportes y pantallas.
- Controles manuales, incluyendo aquellos para autorización y aprobación, formularios, documentación numerada, transmisión de datos, balances y conversión de datos.
- Controles de acceso físico.
- Acceso a estructuras de datos específicas a través de los programas de aplicación.
- Seguridad en sitios remotos.

Modificar

Crear pruebas:

Se modificarán tipos de usuarios y se volverán a ejecutar las pruebas. En cada caso, se verificará si los datos o funciones adicionales quedan correctamente permitidos o denegados.

Se crearán pruebas para cada tipo de usuario y verificar cada permiso, creando transacciones específicas para cada tipo de usuario.

Criterio de Completitud:

Consideraciones Especiales:

Para cada tipo de usuario conocido, las funciones y datos apropiados y todas las transacciones deben funcionar como se esperaba.

El acceso al sistema debe ser revisado y discutido con los administradores de la red y/o del sistema.

Técnicas:

Funciones / Seguridad de Datos: Identificar cada tipo de usuario y las funciones y datos a los que se debe autorizar.

Actividad 14: Pruebas de la interfaz de usuario

Las pruebas de la interfaz de usuario (IU) permitirán asegurar de que el sistema cumpla con sus requisitos funcionales y logre un estándar de calidad alto, al punto que aumente las probabilidades de que los usuarios la adopten con éxito.

La verificación y validación de la interfaz del usuario se hará en tres puntos del proceso, en el análisis (formulación y análisis de requisitos), en el diseño (al diseñar la interfaz que garantice la calidad) y durante las pruebas donde se prueba ejecutando la aplicación.

Los objetivos que se obtendrán con las pruebas de interfaz son:

- Las características de la interfaz se prueban para asegurar que las reglas del diseño, la estética y el contenido visual relacionado están a disposición del usuario sin error alguno.
- Los mecanismos individuales de la interfaz se prueban en forma unitaria (por ejemplo, se prueba HTML dinámico, cgi, agregar activo fijo, login.)
- La interfaz se probará frente cada caso de uso (USN) para descubrir errores de semántica, y facilidad de uso.
- La interfaz se probará dentro de una diversidad de ambientes para asegurar su compatibilidad.

Actividad 15: Pruebas de usabilidad

En las pruebas de usabilidad, los testers de software se enfocarán en validar que tan fácil de usar es la aplicación.

Las características evaluadas en la usabilidad incluyen:

- Facilidad de aprendizaje: Que tan fácil es para los usuarios realizar funciones básicas la primera vez que utilizan la aplicación.
- Eficiencia: Que tan rápido los usuarios experimentados pueden realizar sus tareas.

- Memorización: Que tan fácil de memorizar es el uso de la aplicación, esto es, cuando un usuario pasa mucho tiempo sin usar la aplicación, puede recordar lo suficiente para usarla con efectividad la próxima vez, o tiene que empezar a aprender de nuevo.
- Errores: Cuantos errores atribuibles al diseño comete el usuario, que tan severos son y que tan fácil es recuperarse de los mismos.
- Satisfacción: Que tanto le gusta (o desagrada) al usuario utilizar el sistema.

Actividad 16: Pruebas de aceptación

Elementos que serán sujetos a las pruebas:

- Los procesos de negocio de sistemas que ya han sido integrados.
- Procesos operacionales y de mantenimiento.
- Procedimientos de usuario.
- Formularios.
- Reportes.
- Datos de configuración.

Las pruebas de aceptación servirán para evaluar el grado en que el sistema está listo para ser implementado y usado. Se enfocarán en verificar si el sistema es “apto para el uso”. Se diseñarán principalmente a partir de las especificaciones de requerimientos, casos de uso y de los procesos de negocio definidos.

Puede ocurrir que, en la sesión de aceptación de software, el usuario pueda definir casos de prueba adicionales, por lo que debe hacerse todo lo posible que estos casos sean identificados de manera temprana, preferiblemente mucho antes de llegar a la sesión de aceptación.

- Pruebas de aceptación operacional

Comprende la aceptación del nuevo sistema o funcionalidad por parte de los administradores, es decir el área de operaciones de informática de la organización.

Entre los aspectos a validar se incluyen:

- Pruebas de respaldo y recuperación.
- Recuperación ante desastres.
- Gestión de cuentas de usuario y control de acceso al sistema.
- Tareas de mantenimiento.
- Tareas de carga y migración de datos.
- Revisiones periódicas de vulnerabilidades de seguridad.

Pruebas de aceptación de contratos y regulaciones

Las pruebas de aceptación de regulaciones se realizan verificando que la funcionalidad del sistema cumple con dichas regulaciones, tales como las definidas por el gobierno, las leyes o estándares de seguridad.

Pruebas alfa o beta (Pruebas de campo)


- Los desarrolladores de software comercial (de paquete), a menudo necesitan obtener feedback de clientes existentes o potenciales de su mercado antes de sacar a la venta el producto.
- Las pruebas Alfa son realizadas por la misma organización desarrolladora del software, pero no por el equipo de desarrollo.
- Las pruebas Beta son realizadas por clientes existen o potenciales en sus propias instalaciones. A estas pruebas también se les conoce como pruebas de campo.

CRONOGRAMA DE ACTIVIDADES

	Semanas
--	---------

Actividades	1	2	3	4	5	6	7	8
Registro petición de servicio Estrategia de Pruebas								
Definición del Alcance de las Pruebas								
Elaboración de propuesta de servicios de testing								
Definición de Requisitos del Entorno de Pruebas								
Especificación de pruebas individuales y grupales								
Definición de las Pruebas de Aceptación del Sistema								
Validación de propuesta y acuerdo modelo propuesta aceptación / reentrega								
Determinar necesidades de personal y entrenamiento								
Metodología de pruebas								
Identificar los riesgos y definir planes de respuesta								

Diseño de pruebas								
Ejecución de pruebas								
Las pruebas de seguridad y control de acceso								
Pruebas de la interfaz de usuario								
Pruebas de usabilidad								
Pruebas de aceptación								

	PROCESO		CÓDIGO:
	DISEÑO DE SISTEMAS		VERSIÓN: 1
	FORMATO DE CASOS DE PRUEBAS		FECHA DE REALIZACION: 10/09/2020

DATOS DEL PROYECTO					
Fecha de Iniciación	28 de julio 2020	Nombre del Proyecto	Sistema de control de activo fijo	Versión	1.0
		Caso de Desarrollo	Pruebas de Interfaces	Desarrolladores	Grupo 06

ESPECIFICACIÓN DE PRUEBAS							
ID	Fecha	*Estado	Nombre de Identificador	Descripción del Interfaz	Pasos	Pre-Condición	Resultados Esperados
1	2 octubre 2020	abierto	CasoPrueba1.0	Pantalla de logeo dejando los campos requeridos vacíos	Al momento de cargar el login se dará click en ingresar dejando todos los campos en blanco	Debe estar creada la unidad de base de datos	Mostrara un mensaje resaltando el campo " correo" " password" como requerido y no ingresara
2	2 octubre 2020	Abierto	CasoPrueba1.2	Pantalla de logeo dejando solo el campo password vacío	Al momento de cargar el login se llena el campo correo y se deja en vacío el password y click en ingresar	Debe estar creada la unidad de base de datos	Mostrara un mensaje resaltando el campo password como requerido y no ingresara
3	2 octubre 2020	Abierto	CasoPrueba1.3	Pantalla logeo se llenarán todos los campos requeridos	Al momento de cargar el login se llenarán todos los campos requeridos y click en ingresar	Debe estar creada la unidad de base de datos	Cargara una pantalla con estado de registro en donde muestra los usuarios
4	2 octubre 2020	Abierto	CasoPrueba1.4	Pantalla de logeo se llenarán todos los campos requeridos	Al momento de cargar el login se llenarán todos los campos requeridos y se dará click en	Debe estar creada la unidad de base	Al momento de dar click en el botón mostrar contraseña nos mostrara en el campo password

					<i>mostrar contraseña y luego click en ingresar</i>	<i>de datos</i>	<i>la contraseña que se escribo y luego dar click en ingresar</i>
5	2 octubre 2020	abierto	CasoPrueba1.5	<i>Pantalla de logeo se llenarán todos los campos requeridos</i>	<i>Al momento de cargar el login se llenarán todos los campos requeridos y se escribirá una contraseña incorrecta y click en ingresar</i>	<i>Debe estar creada la unidad de base de datos</i>	<i>Al momento de dar click en el botón ingresar nos mostrará un mensaje con contraseña incorrecta por lo cual se tendrá que digitar la correcta y dar en ingresar</i>
6	2 octubre 2020	abierto	CasoPrueba1.6	<i>Pantalla de logeo se llenarán todos los campos requeridos</i>	<i>Al momento de cargar el login se llenarán todos los campos requeridos y se escribirá un correo incorrecta y click en ingresar</i>	<i>Debe estar creada la unidad de base de datos</i>	<i>Al momento de dar click en el botón ingresar nos mostrará un mensaje con correo incorrecta por lo cual se tendrá que digitar el correcto y dar en ingresar</i>

**Estado: Registrado (inicio) o Resultados (solucionado)*

ESPECIFICACIÓN DE PRUEBAS							
ID	Fecha	*Estado	Nombre de Identificador	Descripción del Interfaz	Pasos	Pre-Condición	Resultados Esperados
1	2 octubre 2020	abierto	CasoPrueba2.0	<i>Pantalla genérica para conocer el estado de una transacción y/o los datos resultantes al envío de una transacción.</i>	1- <i>Seleccionar la opción de "Estado de Registro".</i> 2- <i>Verificar la información registrada.</i> 3- <i>Seleccionar la opción de salir.</i>	<i>Debe haberse hecho el registro, para seleccionar ver esa información.</i>	<i>-Visualizar la información del registro consultado.</i> <i>-Salir de la interfaz y volver a la anterior.</i>


ESPECIFICACIÓN DE PRUEBAS							
ID	Fecha	*Estado	Nombre de Identificador	Descripción del Interfaz	Pasos	Pre-Condición	Resultados Esperados
1	3 octubre 2020	abierto	CasoPrueba3.0	<i>Pantalla de Creación</i>	<i>al momento de cargar la</i>	<i>Debe estar</i>	<i>Mostrara un mensaje en el cual</i>

				<i>de Activo dejando los campos requeridos vacíos</i>	<i>pantalla de creación de activo se dará click en Guardar dejando todos los campos en blanco</i>	<i>creada la unidad de base de datos para activo fijo</i>	<i>se tiene que llenar los campos requeridos para poder guardar el formulario.</i>
2	3 octubre 2020	abierto	CasoPrueba3.1	<i>Pantalla de creación de activo llenado las cajas de texto</i>	<i>al momento de cargar la pantalla de creación de activo se dará click en Guardar llenado las cajas de textos</i>	<i>Debe estar creada la unidad de base de datos para activo fijo</i>	<i>Mostrara un mensaje en el cual se guardo exitosamente</i>
3	3 octubre 2020	abierto	CasoPrueba3.2	<i>Pantalla de creación de activo se dara click en el botón verificar</i>	<i>al momento de cargar la pantalla de creación de activo se dará click en verificar</i>	<i>Debe estar creada la unidad de base de datos para activo fijo</i>	<i>Mostrara un listado con los registros que emos guardado exitosamente en el cual podemos verificar si están exitosamente</i>
4	3 octubre 2020	abierto	CasoPrueba3.3	<i>Pantalla de creación de activo dar click en el botón modificar</i>	<i>al momento de cargar la pantalla de creación de activo se dará click en modificar</i>	<i>Debe estar creada la unidad de base de datos para activo fijo</i>	<i>Mostrar un listado en el cual se elegirá cual registro modificar y al modificar se guardará exitosamente</i>
5	3 octubre 2020	abierto	CasoPrueba3.4	<i>Pantalla de creación de activo dar click en el botón Cerrar</i>	<i>Al momento de cargar la pantalla de creación de activo se dara click en cerrar</i>	<i>Debe estar creada la unidad de base de datos para activo fijo</i>	<i>Al momento de dar click en cerrar este nos sacara de la pantalla de creación de activo</i>

ESPECIFICACIÓN DE PRUEBAS							
ID	Fecha	*Estado	Nombre de Identificador	Descripción del Interfaz	Pasos	Pre-Condición	Resultados Esperados
1	4 octubre 2020	Abierto	Caso de prueba	Pantalla encargada del	1)Ingresar usuario y password	Debe de	Mostrara un mensaje resaltando

			4.0	registro de toda la información referente a un activo fijo.	2)Seleccionar la opción "Registro de Activo" 3)Sin seleccionar ninguna opción, ni llenar ningún campo se dará click en la opción de Guardar.	haberse iniciado de Sesión.	que los campos vacíos son requeridos para hacer el registro.
2	4 octubre 2020	Abierto	Caso de prueba 4.1	Pantalla encargada del registro de toda la información referente a un activo fijo.	1)Ingresar usuario y password 2)Seleccionar la opción "Registro de Activo" 3)Se llenarán todos los campos.	Debe de haberse iniciado de Sesión.	Se mostrará un mensaje confirmando el registro del activo con éxito.
3	4 octubre 2020	Abierto	Caso de prueba 4.2	Pantalla encargada del registro de toda la información referente a un activo fijo. Parte de la depreciación	1)Ingresar usuario y password 2)Seleccionar la opción "Registro de Activo" 3)Se llenarán todos los campos, dejando vacía solo la parte de la depreciación.	Debe de haberse iniciado de Sesión.	Se mostrará un mensaje resaltando que la parte de la depreciación debe de ser llenada.
4	4 octubre 2020	Abierto	Caso de prueba 4.3	Pantalla encargada del registro de toda la información referente a un activo fijo. Parte del mantenimiento	1)Ingresar usuario y password 2)Seleccionar la opción "Registro de Activo" 3)Se llenarán todos los campos, dejando vacía solo la parte del mantenimiento.	Debe de haberse iniciado de Sesión.	Se mostrará un mensaje resaltando la parte del mantenimiento, para seleccionar las opciones requeridas
5	4 octubre 2020	Abierto	Caso de prueba 4.4	Pantalla encargada del registro de toda la información referente a un activo fijo. Parte de "Datos de la compra"	1)Ingresar usuario y password 2)Seleccionar la opción "Registro de Activo" 3)Se llenarán todos los campos, dejando vacía solo la parte de datos de la compra.	Debe de haberse iniciado de Sesión.	Se mostrará un mensaje resaltando la parte de los datos de la compra, para ser llenados, para poder realizar el registro con éxito.
6	4 octubre 2020	Abierto	Caso de prueba 4.5	Pantalla encargada del registro de toda la información referente a un activo fijo.	1)Ingresar usuario y password 2)Seleccionar la opción "Registro de Activo" 3)Se llenarán todos los campos, dejando vacía en la	Debe de haberse iniciado de Sesión.	Se mostrará un mensaje diciendo que el registro se hizo con éxito, ya que la parte de las observaciones es la única información que no es

					parte de la depreciación, las observaciones		obligatoria, para realizar el registro.
--	--	--	--	--	---	--	---

	PROCESO	CÓDIGO:
	DISEÑO DE SISTEMAS	VERSIÓN: 1
	FORMATO DE CASOS DE PRUEBAS	FECHA DE REALIZACIÓN: 10/09/2020

DATOS DEL PROYECTO					
Fecha de Iniciación	28 de julio 2020	Nombre del Proyecto	Sistema de control de activo fijo	Versión	1.0

ESPECIFICACIÓN DE BITACORA		
Fecha	Actividad	Comentarios
2 septiembre 2020	Plantear una propuesta para la creación del sistema de control de activo fijo	Se realizará en conjunto con todo el grupo de trabajo
3 septiembre 2020	Creación de una base de datos	para poder obtener credenciales y poder entrar al sistema y poder guardar todo tipo de registro se necesitará una base de datos al cual podemos tener acceso mediante el sistema para poder manipular todo tipo de registro
9 septiembre 2020	pantalla Creación de diseño de entrada, salida y de reportes	En grupo de trabajo se debate sobre el diseño de interfaces que llevara dicho sistema
11 septiembre 2020	pantalla Creación de logeo del sistema	Se necesitará un correo y una contraseña para poder ingresar
12 septiembre 2020	Pantalla Creación de Estado de registro	Este registro nos permitirá conocer el estado de una transacción y/o los datos resultantes al envío de una transacción.
14 septiembre 2020	Pantalla de Creación de activo	Creación de un activo en el sistema con todos los datos iniciales que requiere.
14 de septiembre 2020	Pantalla de Registro de activo	Pantalla encargada del registro de toda la información referente a un activo fijo, a su vez será enviada dentro de una transacción

Presupuesto de un proyecto de software

Análisis de requerimientos

Generar cuadro de mando de activos fijos
Descripción: Se presentará al usuario una serie de indicadores de acuerdo a sus responsabilidades, esta será de forma gráfica o tabular, los indicadores a presentar serán: últimos recursos asignados, Recursos más reparados, recursos más consumidos, recursos que generan más costos, próximos mantenimientos, Ultimas operaciones (traslados y descargos) y el listado de los recursos más descargados por cantidad y costo total
Generar reporte a contabilidad de ingreso, retiro, y depreciación de los activos fijos
Descripción: Documentos necesarios para respaldar las operaciones de contabilidad con respecto a los bienes de AF de la facultad, este reporte tendrá su pre visualización en pantalla y podrá ser impreso en papel.
Reporte de depreciación a contabilidad.
Descripción: Documento que se presentara a las entidades que así lo requieran como lo es AF, Corte de cuentas de la república entre otros, este reporte tendrá su pre visualización en pantalla y podrá ser impreso en papel.
Reporte de proveedores
Descripción: Documento que se presentara a las entidades que así lo requieran como lo es AF, Corte de cuentas de la república entre otros, este reporte tendrá su pre visualización en pantalla y podrá ser impreso en papel.
Reporte de movimientos
Descripción: Documento que se presentara a las entidades que así lo requieran como lo es AF, Corte de cuentas de la república entre otros, este reporte tendrá su pre visualización en pantalla y podrá ser impreso en papel

Requerimientos funcionales de un sistema transaccional de activo fijo

Con los requerimientos funcionales definidos podemos seguir con el siguiente paso, que será realizar una medición de la magnitud o tamaño del software a desarrollar.

Medición del software

Desarrollaremos la medición en dos pasos, primero determinaremos los componentes funcionales del presupuesto de desarrollo de software, a partir del análisis de requerimientos realizado anteriormente. Seguidamente,

realizaremos el cálculo de los puntos de función, con lo cual obtendremos una medida del tamaño del proyecto.

Determinar los componentes funcionales

Para determinar los componentes funcionales, debemos determinar tanto las transacciones de negocio como los componentes de datos, siguiendo el **método de análisis de puntos de función**.

Transacciones de negocio

Las transacciones de negocio que podemos desglosar a partir de los requerimientos de software son las siguientes:

- Consultar lista de órdenes de compra.
- Ingresar orden de compra.
- Ingresar línea de orden de compra.
- Modificar orden de activo.
- Modificar línea de orden de compra.
- Consultar órdenes de compra por aprobar.
- Aprobar orden de compra.
- Imprimir orden de compra.
- Enviar por email orden de compra.

Seguidamente, clasificamos las transacciones de negocio, que pueden ser de 3 tipos: Entradas, salidas y consultas. Adicionalmente, debemos asignar un nivel de complejidad alto, medio o bajo a cada uno.

Los niveles de complejidad dependen de factores como por ejemplo número de campos no repetidos, número de archivos a ser leídos, creados o actualizados, número de sub grupos de datos o formatos de registros, entre otros.

Al clasificar las transacciones de negocio y asignar los niveles de complejidad se tiene lo siguiente:

- Consulta externa (Alta): Consultar lista de órdenes de compra.
- Entrada externa (Bajo): Ingresar orden de compra.
- Entrada externa (Bajo): Ingresar línea de orden de compra.
- Entrada externa (Bajo): Modificar orden de compra.
- Entrada externa (Bajo): Modificar línea de orden de compra.
- Consulta externa (Alta): Consultar órdenes de compra por aprobar.
- Entrada externa (Bajo): Aprobar orden de compra.
- Salida externa (Bajo): Imprimir orden de compra.
- Salida externa (Medio): Enviar por email orden de compra.

Componentes de datos

Para determinar los componentes de datos para nuestro ejemplo de presupuesto de un proyecto de software,

necesitamos tener una idea preliminar de cuál será el modelo de datos, o bien las entidades de nuestro modelo entidad relación.

Como estamos en una fase de presupuesto, no podemos realizar el análisis de sistema e ingeniería de software completa, pero debemos tratar de llegar a un nivel de detalle suficiente para poder elaborar un presupuesto realista.

Los componentes de datos se pueden clasificar en archivos internos y archivos externos. Igualmente, debemos asignar un nivel de complejidad. En el caso de componentes de datos dependerá del número de campos que tenga (más campos más complejidad) y las agrupaciones de estos datos, entre otros aspectos.

Siguiendo el ejemplo hemos determinado que los componentes de datos son los siguientes:

- Archivo lógico interno (Bajo): Tabla de Orden de compra
- Archivo lógico interno (Bajo): Tabla de línea de orden de compra
- Archivo lógico interno (Bajo): Tabla de Proveedor
- Archivo lógico interno (Bajo): Tabla de Artículo

Cálculo de los puntos de función

El método de puntos de función establece una cierta cantidad de puntos a asignar según el nivel de complejidad de los componentes funcionales, esta medida es relativa a unos a otros y a mayor complejidad mayor cantidad de puntos de función asignados.

Los puntos de función se asignan según la siguiente tabla:

Tipo de función	Puntos de función (Dificultad Baja)	Puntos de función (Dificultad Media)	Puntos de función (Dificultad Alta)
Entrada externa (EI)	x3	x4	x6
Salida externa (EO)	x4	x5	x7
Consulta externa (EQ)	x3	x4	x6
Archivo lógico interno (ILF)	x7	x10	x15
Archivo lógico externo (ELF)	x5	x7	x10

Utilizando la tabla, podemos realizar el conteo de puntos de función para los componentes funcionales.

- Consulta externa (Alta): Consultar lista de órdenes de compra. **6 puntos.**
- Entrada externa (Bajo): Ingresar orden de compra. **3 puntos.**
- Entrada externa (Bajo): Ingresar línea de orden de compra. **3 puntos.**
- Entrada externa (Bajo): Modificar orden de compra. **3 puntos.**
- Entrada externa (Bajo): Modificar línea de orden de compra. **3 puntos.**
- Consulta externa (Alta): Consultar órdenes de compra por aprobar. **6 puntos.**
- Entrada externa (Bajo): Aprobar orden de compra. **3 puntos.**
- Salida externa (Bajo): Imprimir orden de compra. **4 puntos.**
- Salida externa (Medio): Enviar por email orden de compra. **5 puntos.**
- Archivo lógico interno (Bajo): Tabla de Orden de compra. **7 puntos.**

- Archivo lógico interno (Bajo): Tabla de línea de orden de compra. **7 puntos.**
- Archivo lógico interno (Bajo): Tabla de Proveedor. **7 puntos.**
- Archivo lógico interno (Bajo): Tabla de Artículo. **7 puntos.**

Total de puntos de función: 64

Al determinar los puntos de función tenemos una medida de la magnitud del tamaño del software y del esfuerzo que se requiere para desarrollarlo.

Productividad del equipo de trabajo

Para continuar con el ejemplo de presupuesto de un proyecto de software, necesitamos conocer cuántos puntos de función por unidad de tiempo puede desarrollar nuestro equipo de trabajo en un tiempo dado, esto para determinar la duración del proyecto.

Necesitamos valernos de la información de proyectos pasados que tenga la organización, también podemos usar información de otras fuentes, otras organizaciones y bases de datos de Benchmarking.

Para que la medición sea exacta, debemos considerar puntos de función completamente desarrollados, es decir donde se ejecutó el análisis, diseño, desarrollo, pruebas. Es decir, todo lo necesario para que sea puesto en producción.

Determinar la productividad solamente con tiempos de desarrollo no nos ayuda en este caso.

Considerando que un mes tiene 21 jornadas en promedio, podemos determinar el número de jornadas que necesitamos para producir 32 puntos de función, a saber:

Para producir 32 puntos de función necesitamos:

- 84 jornadas de desarrolladores de software.
- 42 jornadas de testing.
- 21 jornadas de Project Management.

Por lo tanto la productividad del equipo es:

- Gerente de proyecto: 1,52 puntos de función por jornada.
- Desarrollo de software: 0,38 puntos de función por jornada.
- Testing de Software: 0,76 puntos de función por jornada.

Estimación de esfuerzo y personal necesarios en base a la productividad

Conocida la productividad, podremos realizar el cálculo del esfuerzo necesario (medido en jornadas) y estimación de personal.

En el ejemplo, sabiendo que el desarrollo de software tiene una medición de 64 puntos de función, podemos determinar que necesitamos:

- Gerente de proyectos: 44 jornadas.
- Desarrolladores de software: 168 jornadas.
- Testers: 84 jornadas.

¿Cuántas personas necesitamos para ejecutar el proyecto? Dependerá del tiempo en que necesitemos ejecutarlo. En nuestro **ejemplo de presupuesto de un proyecto de software**, si mantenemos la configuración de nuestro equipo de trabajo, el proyecto tomará 2 meses.

Podríamos aumentar el número de personas para ejecutarlo en menos tiempo, sin embargo, no podemos aplicar la misma productividad que calculamos en base al equipo de trabajo previo, pues está demostrado que la relación cantidad de personas con la productividad no es linealmente proporcional.

Para determinar esta nueva productividad necesitamos un factor de ajuste o encontrar información de proyectos ejecutados con más personas (para conocer cuántos puntos de función se desarrollaron en un tiempo dado).

Costos del personal del proyecto de desarrollo de software

Teniendo definidas el número de jornadas que requiere nuestro proyecto de desarrollo de software, pasamos a determinar los costos, para lo cual lo primero que necesitamos conocer es el costo por jornada del personal.

El costo presupuestado para el desarrollo del sistema en doce meses de trabajo es el siguiente:

7 desarrolladores con un costo de \$2,000.00 mensuales por 12 meses de trabajo c/u se totaliza en \$168,000.00 menos el

descuento de la Renta por Servicios Profesionales de \$16,800.00 totaliza \$151,200.00

Teniendo un monto para imprevistos del 25% sobre el precio total (sin descuentos) de \$42,000.00

El precio Total del Sistema de Activo Fijo se totaliza en \$210,000.00

Costo fijo: Se tendrá en cuenta para el costo fijo el pago de un local arrendado y amueblado para oficina por \$400.00 mensuales para uso de oficinas de desarrollo que incluye agua potable, luz eléctrica y servicios públicos, por 12 meses lo cual totaliza \$4,800.00

La depreciación de equipo informático de 7 programadores destinados al desarrollo del sistema de Activo Fijo se prevee en \$10.00 mensuales c/u y 3 servidores con el detalle:

- Servidor de aplicaciones \$50.00
- Servidor de Base de Datos \$50.00
- Servidor de Servicios de Red de uso general (FTP, HTTP, Git, etc.) \$50.00
- Generando un total de depreciación de equipos por 12 meses de \$3,800.00

Costo Variable: Dado que el local arrendado posee los servicios de uso básicos y amueblado, se incluye el costo de alimentación para almuerzos del equipo de desarrollo en concepto de viáticos o “per diem” por un total de \$4.50 c/u estimando los días previstos de desarrollo por 53 semanas laborales de 5 días sin extra tiempo, totalizando un valor de

costo variable por \$8,347.50 hasta \$10,000.00

Y una reserva de fondos por extra tiempo de un día a la semana que se deduce del cobro individual por \$1,669.50

Para nuestro ejemplo de presupuesto, los costos mensuales de nuestro personal serán los siguientes:

7 desarrolladores con un costo de \$2,000.00 mensuales por 12 meses de trabajo c/u se totaliza en \$168,000.00 menos el descuento de la Renta por Servicios Profesionales de \$16,800.00 totaliza \$151,200.00

Con estos datos podríamos determinar el costo por jornada, y como conocemos las jornadas del proyecto podríamos calcular el presupuesto. Sin embargo, para obtener una medida de costo que podamos aplicar universalmente a todos los proyectos, es más útil definir cuál es el costo.

CONCLUSIONES

El campo de la tecnología de la información ya no es nuevo, y ha llegado el momento de que la educación se centre en producir productos de calidad de forma más rápida y económica y con muchas alternativas nuevas sobre cómo gestionar y modelar un sistema utilizando herramientas de análisis sofisticadas y prácticas de gestión avanzadas, así como enfatizar el cómo y cuándo pueden aplicarse mejor y qué beneficios pueden derivarse de su aplicación.

Hay una serie de temas nuevos que son parte integral del proceso de desarrollo de software, incluidas áreas como ciber seguridad, big data y la transformación digital, así como interactuar directamente con los usuarios, pero el verdadero desafío siempre se aborda en el cómo los analistas necesitan pronosticar o predecir lo que necesitarán los usuarios en el futuro. Por ejemplo, los analistas deberán asumir riesgos y comprender que los requisitos predictivos que puedan desarrollar, tendrían una vida útil relativamente corta, antes de que nuevas necesidades de los usuarios o las leyes mismas los puedan volver obsoletos y desafortunadamente no existen soluciones instantáneas para predecir exitosamente si se producirán fallas o no.