**UNIVERSIDAD COOPERATIVA DE COLOMBIA**

2016

**TASHEET TESTING**

**Facultas de Ingeniería de Sistemas**

**Norma Márquez Castro – Diego Gualaco**

**TASHEET TESTING**

**DESCRIPCIÓN**

Desarrollar una aplicación que va dirigida a empresas y/o outsourcing, con el objetivo de ofrecer un rápido y sencillo acceso a resultados de evaluación de cada uno de sus empleados y operarios de una mesa de ayuda. La aplicación funciona sobre cualquier sistema operativo que cuente con acceso a internet.

Se diseñará un sistema de fácil acceso y de forma rápida a visualizar los resultados por personas, por días, por semanas y por meses.

Considerando necesario el desarrollo de un nuevo sistema de gestión de auditorías, así como las bases de datos que recogen datos tanto estadísticos y empresariales del personal, por tanto los resultados de estas auditorías tendrán una gestión más rápida, automática y segura.

**OBJETIVO GENERAL**

Desarrollar una aplicación web, donde se pueda visualizar y evidenciar la productividad, eficiencia, eficacia, cumplimiento y la calificación de los agentes de una mesa de ayuda con el fin de evitar incumplimientos. Teniendo la información detallada del sistema de información de calidad, que satisfagan la necesidad y brinde los resultados a los gerentes, sobre el desempeño de los empleados de la compañía.

**OBJETIVOS ESPECÍFICOS**

* Crear la base de datos que nos permite almacenas la información de la aplicación
* Presentar una interfaz gráfica amable para el usuario.
* Presentar una aplicación sea accesible a través de internet.
* Crear y editar un diagrama de flujo que nos permita tener claros los procesos de nuestro programa.
* Sistema que permitirá la optimización de recursos y el desarrollo de reportes de una forma más rápida, ágil y segura.
* Desarrollo de una herramienta que permita medir a las personas en su desempeño laboral.
* Mayor seguridad de la información.
* Agilizar los procesos de medición en la organización.
* Permite llevar el control y medición para cada una de las actividades de los empleados de la organización
* Es un sistema de automatización que se podrá consultar en tiempo real.

**JUSTIFICACIÓN**

El diseño de esta herramienta donde se pueda ingresar a través de un sitio web llevando a cabo el aseguramiento de la calidad mediante el monitoreo de las llamadas a los agentes de una mesa de ayuda, velando por el cumplimiento de los parámetros establecidos de la prestación del servicio a través de los ítems:

* Productividad de los empleados
* Medición en porcentaje de analistas
* Detectar oportunidades de mejora.
* Identificar necesidades en pro del mejoramiento continuo de la operación.
* Resultados disponibles diarios, semanales y/o mensuales.

**SOFTWARE A UTILIZAR**

Para el desarrollo de nuestra aplicación vamos utilizar el lenguaje de programación PHP integrándolo con HTML para el diseño gráfico

**MODELOS DE DESARROLLO DE METODOLOGÍAS**

Una de las principales acciones en la gestión de proyectos se trata de aplicar la mejor metodología para minimizar los errores, aumentar la eficacia y eficiencia y optimizar resultados a través de mejoras continuas, algunas de las metodologías más utilizadas son:

**Metodología en Cascada:** Se establece en actividades por separado del proceso, como:

* Requisitos: debe comprender el ámbito de la información del software, así como la función, el rendimiento y las interfaces requeridas.
* Diseño: En el diseño del software se enfoca en la estructura de los datos, la arquitectura del software, el detalle procedimental y la caracterización de la interfaz.
* Implementación: Se debe traducir en una forma legible para la máquina.
* Pruebas: La prueba se centra en la lógica interna del software y en las funciones externas, realizando pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren con nuestra aplicación de seguimiento.
* Mantenimiento: Estos ocurrirán cuando se haya encontrado errores, a que el software deba adaptarse a cambios del entorno externo (sistema operativo o dispositivos periféricos) o a que el cliente requiera ampliaciones funcionales o del rendimiento.

Sin embargo, uno de los problemas de esta metodología es que no existe una comunicación constante con el cliente, siempre se sigue de un paso al otro y si existe un error se notará en la etapa de pruebas, en este momento será necesario devolverse a la etapa inicial, lo cual tomará más tiempo

**Metodología en Espiral:** También llamado el modelo de desarrollo evolutivo, surge de un sistema inicial que se desarrolla rápidamente a partir de especificaciones abstractas, basándose en las peticiones del cliente para producir un sistema que satisfaga sus necesidades, en este modelo se ve una nueva etapa que es el análisis de los riesgos y siempre se están haciendo ajustes al producto final.

Sin embargo, uno de los problemas de esta modelo es que toma demasiado tiempo y tiende a entregar un producto en un tiempo mayor al que probablemente el cliente desee.

**Metodología XP: l**lamado Extreme Programming (XP), donde se plantea una metodología ágil, enfocada a la satisfacción del cliente y a reducir costos cuando existe un cambio ya que define 4 variables: costo, tiempo, calidad y alcance, este modelo se orienta a una estrecha relación con el cliente en todo el proceso del desarrollo, lo cual permite garantizar que el producto final cumpla con todos los requerimientos del cliente.

Sin embargo, puede generar altos costos en horas de desarrollo y algunos problemas para el trabajo en equipo ya que el cliente puede generar retrasos en el proceso.

**Crystal Methodologies:** Se centra en el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo (de ellas depende el éxito del proyecto) y la reducción al máximo del número de artefactos producidos. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas.

**Dynamic Systems Development Method6 (DSDM):** Desarrollo de un proceso de producción de software, con el objetivo de crear una metodología RAD unificada. Sus principales características son: proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos. Propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación.

**Adaptive Software Development (ASD):** Sus principales características son: iterativo, orientado a los componentes software más que a las tareas y tolerante a los cambios. El ciclo de vida que propone tiene tres fases esenciales:

* Especulación: se inicia el proyecto y se planifican las características del software
* Colaboración: en la segunda desarrollan las características.
* Aprendizaje: Se revisa su calidad, y se entrega al cliente.

La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo.

**SCRUM**: es una metodología ágil de desarrollo, un conjunto de prácticas y reglas que abarca la transparencia y la adaptación de los requerimientos nuevos que nacen en medio de la construcción de otros, como suele suceder en muchas ocasiones donde el cliente tiene ideas nuevas con respecto a lo que solicitó.

**METODOLOGÍAS A APLICAR EN EL PROYECYO**

El modelo a aplicar es el XP donde se definen cuatro variables:Costo, tiempo, calidad y alcance.Este ciclo de vida incluye, entender lo que el cliente necesita, crear la solución y entregar el producto final al cliente. Se trata de realizar ciclos de desarrollo cortos (llamados iteraciones), con entregables funcionales al finalizar cada ciclo. En cada iteración se realiza un ciclo completo de análisis, diseño, desarrollo y pruebas.

**Fase de exploración:** Es la fase en la que se define el alcance general del proyecto. En esta fase, el cliente define lo que necesita mediante la redacción de sencillas “historias de usuarios”. Los programadores estiman los tiempos de desarrollo en base a esta información. Debe quedar claro que las estimaciones realizadas en esta fase son primarias (ya que estarán basadas en datos de muy alto nivel), y podrían variar cuando se analicen más en detalle en cada iteración. Esta fase dura típicamente un par de semanas, y el resultado es una visión general del sistema, y un plazo total estimado.

**Fase de planificación:** La planificación es una fase corta, en la que el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario, y, asociadas a éstas, las entregas. Típicamente esta fase consiste en una o varias reuniones grupales de planificación. El resultado de esta fase es un Plan de Entregas, o “Release Plan”, como se detallará en la sección “Reglas y Practicas”.

**Fase de Iteracciones:** Esta es la fase principal en el ciclo de desarrollo de XP. Las funcionalidades son desarrolladas en esta fase, generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración. Como las historias de usuario no tienen suficiente detalle como para permitir su análisis y desarrollo, al principio de cada iteración se realizan las tareas necesarias de análisis, recabando con el cliente todos los datos que sean necesarios. Las iteraciones son también utilizadas para medir el progreso del proyecto. Una iteración terminada sin errores es una medida clara de avance.

**Fase de puesta en producción:** Si bien al final de cada iteración se entregan módulos funcionales y sin errores, puede ser deseable por parte del cliente no poner el sistema en producción hasta tanto no se tenga la funcionalidad completa. En esta fase no se realizan más desarrollos funcionales, pero pueden ser necesarias tareas de ajuste (“fine tuning”).

**ENTREGABLES DEL PROYECTO**

|  |  |  |
| --- | --- | --- |
| **METODOLOGÍA XP** | **ACTIVIDADES** | **FASE DE CUMPLIMIENTO** |
| **PLANIFICACIÓN** | Descripción | 07 Septiembre de 2016 |
| Objetivo General | 07 Septiembre de 2016 |
| Objetivo Específico | 07 Septiembre de 2016 |
| Justificación del proyecto | 09 Septiembre de 2016 |
| Software a utilizar | 09 Septiembre de 2016 |
| Metodología Xp a aplicar | 14 Septiembre de 2016 |
| Requerimientos funcionales | 28 Septiembre de 2016 |
| Requerimientos no funcionales | 28 Septiembre de 2016 |
| SQA Aplicado al proyecto | 01 Octubre de 2016 |
| Historias de Usuarios | 07 Octubre de 2016 |
| Versiones de Software | 01 Noviembre de 2016 |
|  |  |
| **DISEÑO** | Página de autenticación | 28 Septiembre de 2016 |
| Banner y menús desplegables | 30 Septiembre de 2016 |
| Diseño página principal | 05 Octubre de 2016 |
| Diseño submodulos | 05 Octubre de 2016 |
| Modelo Relacional | 07 Octubre de 2016 |
| Modelo Entidad Relación | 07 Octubre de 2016 |
| Casos de Uso | 07 Octubre de 2016 |
|  |  |
| **DESARROLLO** | Página de autenticación en funcionamiento | 12 Octubre de 2016 |
| Banner y menús desplegables | 14 Octubre de 2016 |
| Diseño página principal | 05 Octubre de 2016 |
| Diseño de banner y paneles | 18 Octubre de 2016 |
|  |  |
|  |  |
| **PRUEBAS** | Prueba de autenticación | 14 Octubre de 2016 |
| Pruebas de visualización de Banner y menús desplegables | 14 Octubre de 2016 |
|  |  |
|  |  |
|  |  |

**REQUERIMIENTOS FUNCIONALES**

* Acceso a la aplicación vía web.
* Se puede ingresar desde un PC o desde un dispositivo móvil.
* Autenticación en la aplicación.
* Menús desplegables
* Módulo de plantillas donde se registran las preguntas a evaluar
* Almacenamiento de datos en la BD de una forma segura y protegida.
* Visualización de estadísticas por grupo.
* Impresión en pdf
* Propuestas de Acciones de mejora

**REQUERIMIENTOS NO FUNCIONALES**

* Se debe contar con conexión a internet
* Interfaz de fácil uso y amigable para los auditores
* Restricciones de tiempo “Timeout”, si la aplicación detecta que no hay movimientos la sesión se cerrará.
* Utilización de software libre.

**SQA APLICADO AL PROYECTO**

1. Estándares

En estos estandares es donde se tiene que tratar de implementar el proyecto, con el fin de que se logre tener una propiedad colectiva con el codigo. Todos y cada uno de los programadores deben tener los conocimientos suficientes para que en cualquier comento cuando puedan continuar con la codificacion que alguien mas empezo sin que se presente un traumatizmo para nadie.

Uno de los principales motivos por los que se prueve esta practica dentro de la programacion extrema es la posibilidad que brinda de evitar los cuellos de botella, un ejemplo es si algun programador se retrasa debido a incovenientes no estimados puede ser ayudado o reemplazado por otro programador, es recomendable prestar atencion a este mecanismo de integracion, para evitar problemas en el proyecto que reduzcan bien sea la calidad del proyecto o el rendimiento de desarrollo, tambien es de gran importancia contar con estandares confiables y apropiados, pues las tareas de monitoreo, evalucion y auditoria toman como entrada estas definiciones.

1. Revisiones

Las revisiones se van a realizar periodicamente donde se observara y se verificara que las actividades se esten llevando adecuadamente según el cronograma de actidades, en estas revisiones se identificara de defectos en el producto de trabajo durante el ciclo de vida del software las cuales se llevaran 6 etapas: planificacion, orientacion, preparacion, inspeccion, rework y seguimiento las cuales seran llevadas por un equipo la cual tendran actividades y responsabilidades definidas.

1. Prueba

En estas pruebas se enfatizan mucho los aspectos relacionados con la metodologia XP, clasificandolas en diferentes tipos y funcionalidades especificas, indicando quien, cuando y como deben ser implementadas y ejecutadas. Cuando se tiene bien implementadas las pruebas no habra temor de modificar el codigo de otro programador en sentido que si se daña alguna seccion, las pruebas mostraran el error y permitiran encontrarlo. Si se tiene un grupo de pruebas que garantice su buen funcionamiento, este temor se mitiga en gran medida.

Con la metodologia XP se debe ser muy estricto con las pruebas, solo se debera liberar una nueva version al producto cuando la pasada este al 100% confiable y sin errores.

1. Analisis de defectos

Los defectos aparecen y ocurren a lo largo de todo el ciclo de vida del software sin excepcion, por eso es importante hacer las revisiones para detectar si hay algun problema con el desarrollo del software y cuando se encuentren ese tipo de errores tratar de solucionarlos lo mas pronto posible que el las personas que sigan avanzando en el proyecto no vayan a presentar mas fallas por ese mismo error, esto se llevara con auditorias y evaluaciones que seran las encargadas de validar que si se este cumpliendo con el proceso adecuado a la hora del desarrollo del software. SQA debe responsabilizarse de crear el metodo para que se identifiquen los errores, analizarlos y realizar los cambios necesarios para mejorar su eficiencia y eficacia.

1. Gestión de configuración

En esta gestión se debe garantizar la integridad de los productos a través del ciclo de vida del software, en la cual SQA debe garantizar que las actividades se adjunten al plan de SCM (Software Configuration Management) para que se logre todo el proceso en el cual van incluidos todos los procesos del proyecto y serán calificados y auditados para la verificación que si se está cumpliendo lo mencionado anteriormente.

**HISTORIAS DE 5 USUARIOS**

El proyecto consiste en el desarrollo de un sistema de gestión para una empresa de confecciones. En dicha gestión de la empresa se incluyen gestión de pedidos, gestión de clientes (tanto principal como los de temporada), facturación, gestión de productos, gestión de materias primas, etc...

|  |  |
| --- | --- |
| **Número: 1** | **Usuario:** Secretaria |
| **Nombre historia:** Introducción de pedido (cliente preferente) | |
| **Puntos estimados:** 4 | **Programador responsable:** Paco Valero – David Ferrer |
| **Descripción:**  El pedido llega en un fichero de texto ASCII y es procesado automáticamente siguiendo el formato de la plantilla de pedido e introduciendo éste en la base de datos. | |
| **Observaciones:** | |

|  |  |  |
| --- | --- | --- |
| **Número: 2** | | **Usuario:** Secretaria |
| **Nombre historia:** Gestión de clientes normales (de *temporada*) | | |
| **Puntos estimados:** 3.5 | | **Programador responsable:** Paco Valero |
| **Descripción:**  Llevaremos el alta, baja y modificación de los datos relacionados con los clientes. | | |
| **Observaciones:**  Los clientes de temporada sólo hacen un pedido al principio de la temporada. | | |
| **Número: 3** | **Usuario:** Almacenista | |
| **Nombre historia:** Generación de albaranes | | |
| **Puntos estimados:** 3 | **Programador responsable:** David Ferrer | |
| **Descripción:**  El almacenista seleccionará la opción del menú “Albaranes”, verá el listado de artículos contenidos en el albarán, tras seleccionar uno. En cada artículo se muestra la cantidad de artículos requerida y el almacenista verificará para cada artículo que se puede satisfacer la demanda con el stock disponible en el almacén, escribe la cantidad disponible (menor o igual a la solicitada) y pulsa el botón imprimir el cual:  1) Imprime el albaran con los productos los cuales se puede servir alguna cantidad.  2) Guarda el albaran impreso en la base de datos.  3) Comprueba si alguno de los pedidos de los cuales sirve material se ha completado y de ser así se marca este como completado y se actualiza la fecha de envío a la actual. | | |
| **Observaciones:**   * Albarán = pedidos desglosados por tiendas de un cliente (1 albarán puede corresponderse con múltiples pedidos retrasados y/o actuales). * El almacenista sólo ve el albaran del día (lista de productos). * *El marcar como completado un pedido no está claro si es aquí o en la historia 2.* | | |

|  |  |
| --- | --- |
| **Número: 18** | **Usuario:** Almacenista |
| **Nombre historia:** Recepción de productos semielaborados y artículos | |
| **Puntos estimados:** 1.1 |  |
| **Descripción:**  Cuando el pedido de producto semielaborado o de artículos llega se marca como procesado en la BD y se actualiza el stock existente. | |
| **Observaciones:** | |

|  |  |
| --- | --- |
| **Número: 16** | **Usuario:** Jefe de producción |
| **Nombre historia:** Mantenimiento de productos de proveedores de materias primas | |
| **Puntos estimados:** 1 | **Iteración asignada:** ninguna |
| **Programador responsable:** | |
| **Descripción:**  Alta, baja y modificación de las materias primas que nos suministran los proveedores. | |
| **Observaciones:**  Sólo productos del proveedor | |

**MANEJO DE VERSIONES DE SOFTWARE**

El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante. A pesar de que los ejemplos de este libro muestran código fuente como archivos bajo control de versiones, en realidad cualquier tipo de archivo que encuentres en un ordenador puede ponerse bajo control de versiones.

**SISTEMAS DE CONTROL DE VERSIONES DE SOFTWARE**

**Subversión**

Fue lanzado en el año 2000 bajo una licencia Apache/BSD y su última versión estable fue liberada en febrero de este mismo año. Subversión es uno de los sistemas más legendarios, sin embargo su uso ha ido decreciendo con el paso de los años. Hay quienes afirman que está cerca del final de su ciclo de vida pero todavía miles de empresas lo usan en el día a día.

**Git**

Fue desarrollado nada más y nada menos que por Linus Torvals, el mismo padre del kernel Linux, en el año 2005. [Git](http://git-scm.com/) surge como alternativa a BitKeeper, un control de versiones privativo que usaba en ese entonces para el kernel. Es liberado bajo una licencia GNU GPLv2 y su última versión estable fue publicada a inicios de Abril de este año. Se ha convertido en uno de los más usados alrededor del mundo.

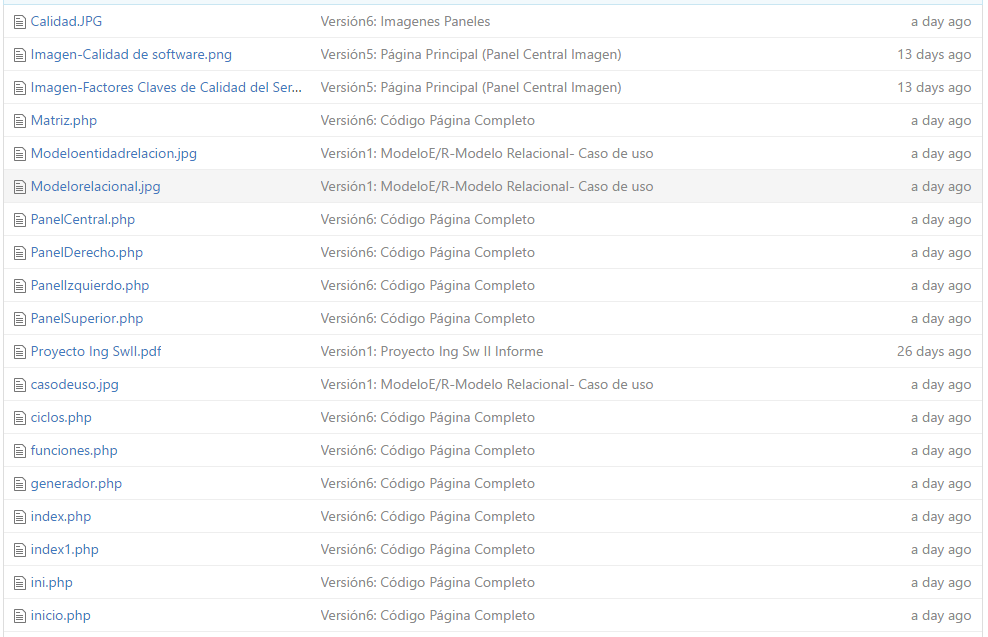
**Mercurial**

Este proyecto nació en 2005 y su desarrollo comenzó a pocos días del de Git, por esto y por algunas similitudes en sus características es que muchos los consideran sistemas hermanos. [Mercurial](http://mercurial.selenic.com/) fue desarrollado por Matt Mackall, y al igual que Linus, Matt buscaba una alternativa a BitKeeper para el control del versiones del kernel Linux. También es liberado bajo una licencia GNU GPL v2 y su última versión estable fue publicada en Abril de este mismo año.

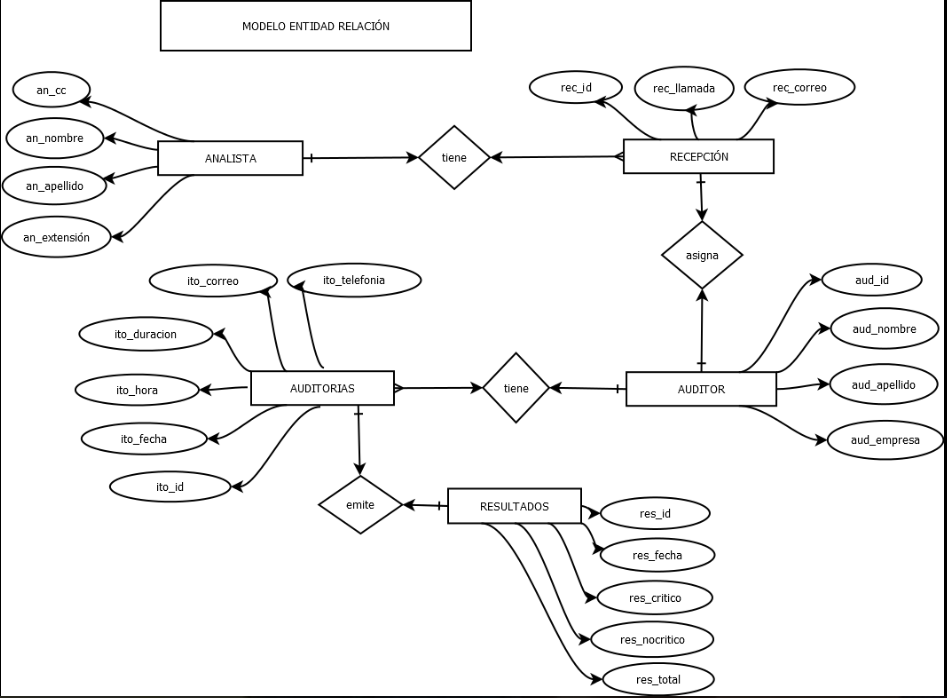
**VERSIÓN DE SOFTWARE APLICADA AL PROYECTO**

El repositorio utilizado en el proyecto para controlar las versiones del programa, perfecciones, modificaciones y control fue GitHub, siendo un servicio de almacenamiento de repositorios de software que utilizan Git el cual se utilizó como base de datos central que contiene todos los archivos cuyas versiones se controlan y sus respectivas historias. Cada desarrollador tiene su propia copia de trabajo, cada uno de nosotros puede obtener la última versión del repositorio, trabajar en línea y localmente.

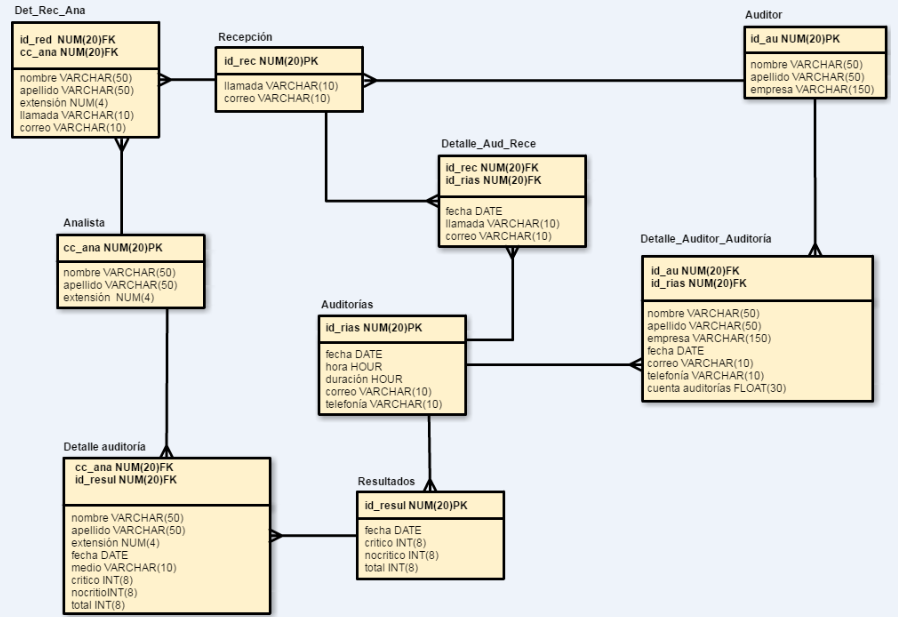
Permitiéndonos alojar tanto el proyecto de software libre de forma gratuita y con gran capacidad.



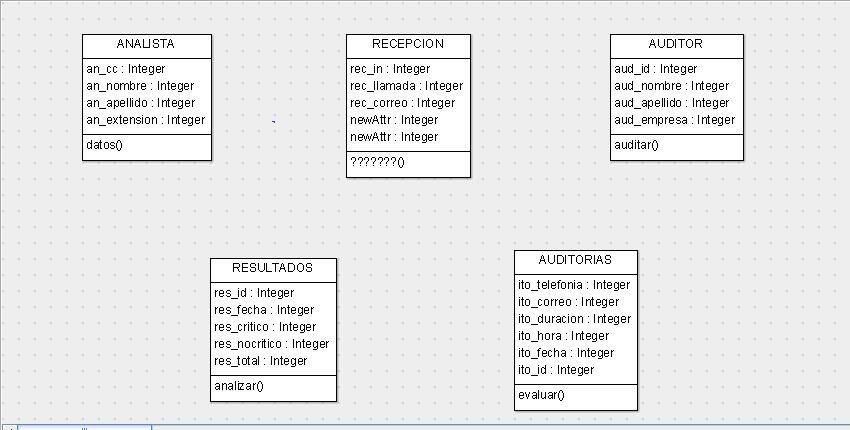
**MODELO ENTIDAD – RELACIÓN**



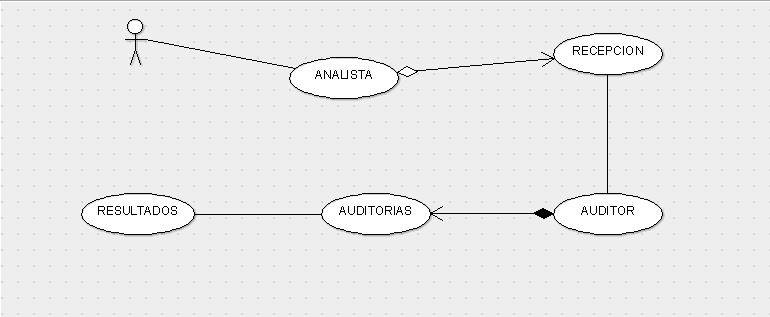
**MODELO RELACIONAL**



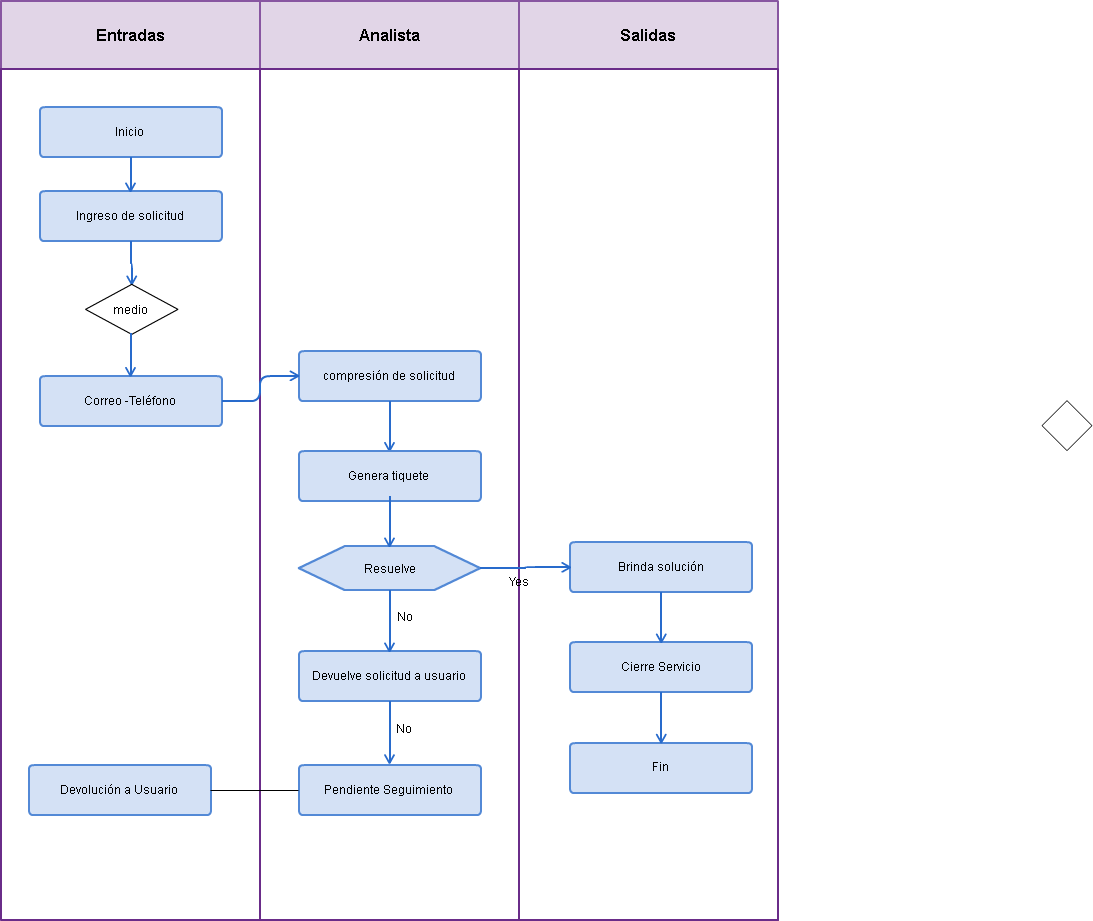
**DIAGRAMA DE CLASES**

****

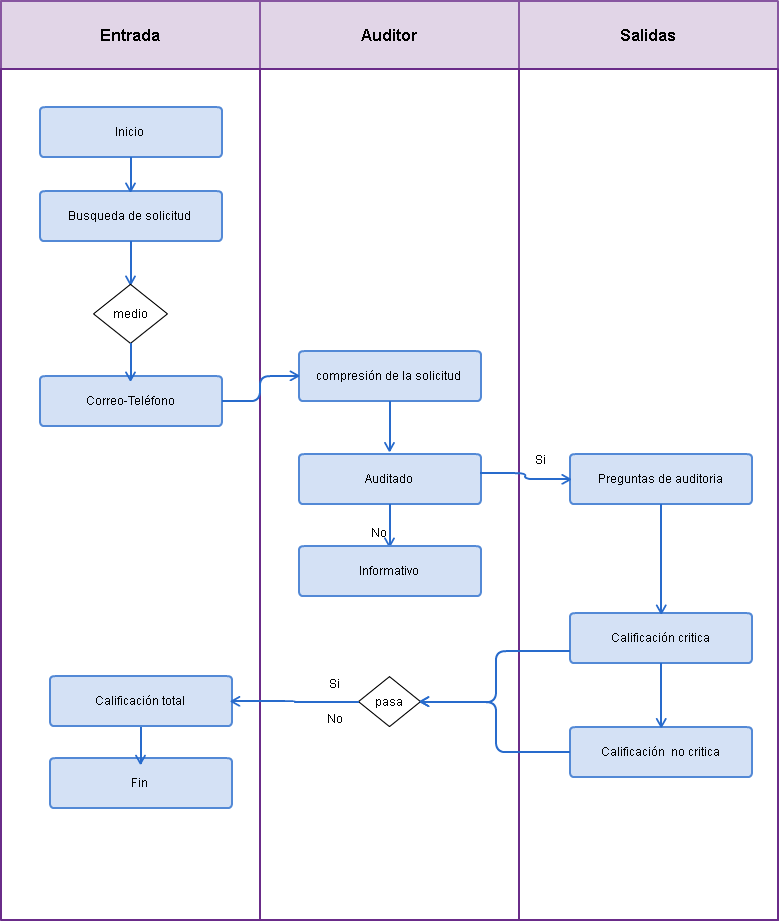
**CASOS DE USO**

****

**DIAGRAMA DE PROCESO GENERAL**

****

**DIAGRAMA AUDITIRAS**

****

**WEBGRAFÍA**

* <http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf>
* <http://users.dsic.upv.es/asignaturas/facultad/lsi/ejemploxp/index.html>
* <http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm>
* <http://repository.unimilitar.edu.co/bitstream/10654/13235/1/Proyecto%20investigacion%20-%20Claudia%20Perez.pdf>
* <http://www.cyta.com.ar/ta0502/v5n2a1.htm>
* <http://blog.leanmonitor.com/es/que-son-las-metodologias-agiles/>
* <http://csandoval.net/files/ejemplo.pdf>
* <https://procesosdesoftware.wikispaces.com/METODOLOGIA+XP>
* <http://ccia.cujae.edu.cu/index.php/siia/siia2008/paper/viewFile/1174/246>
* <http://www.i2btech.com/blog-i2b/tech-deployment/5-beneficios-de-aplicar-metodologias-agiles-en-el-desarrollo-de-software/>
* <http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf>
* <https://git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones>
* <https://hipertextual.com/archivo/2014/04/sistema-control-versiones/>
* <https://tortoisesvn.net/docs/release/TortoiseSVN_es/tsvn-qs-basics.html>
* <http://www.genbetadev.com/sistemas-de-control-de-versiones/kit-basico-de-herramientas-para-desarrollar-en-equipo-i-control-de-versiones>
* <http://www.portalprogramas.com/descargar/control-versiones>