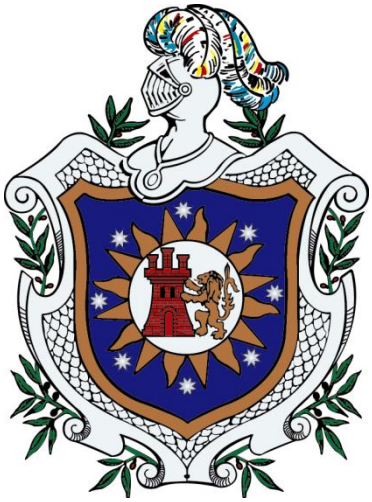
UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA

UNAN – LEÓN

DEPARTAMENTO DE COMPUTACIÓN

MAESTRÍA EN TECNOLOGÍA DE LA INFORMÁTICA EMPRESARIAL



Tema: Desarrollo de una aplicación móvil para dispositivos Android bajo plataforma IDE Eclipse para automatizar la toma de datos y análisis de la información asociada al proceso de control de calidad en la etapa de producción de la empresa TIDE-Manufacturing en periodo enero a julio 2015

.

Autor: Ing. Miguel Ángel Bárcenas Lezama

Tutor: Msc. Julio Cesar González Moreno

Resumen

La zona franca TIDE-Manufacturing ubicada en la ciudad de Managua Nicaragua, es una empresa textilera que realiza productos en base a pedidos internacionales, esta se encarga de elaborar prendas deportivas a diferentes empresas.

TIDE-Manufacturing al igual que muchas empresas trabaja con gran cantidad de información proveniente de todos los procesos de elaboración de dichas prendas, esto genera un problema, ya que la única forma de gestionar correctamente esta información es digitalizarla. Actualmente se ha dado a la tarea de sistematizar gran parte de sus procesos (pedido, inventario, nómina etc.) pero aún hay procesos importantes que no han sido automatizados como el de control de calidad de producción.

El departamento de control de calidad le permite a la empresa ofrecer un producto de calidad a sus clientes, también saber que máquina en producción está defectuosa. Antes de realizar el presente trabajo estas operaciones se realizaban de manera manual lo que implicaba mucho tiempo y errores humanos debido a la gran cantidad de información con la que se trabaja.

Por esta razón es necesario desarrollar una aplicación móvil para agilizar la toma de datos. Esto se logrará ya que los dispositivos son manipulados por el personal de control de calidad, cuando se estén revisando las prendas y en caso de que se encuentre un error de manufactura o cualquier otro daño en las prendas el personal deberá ingresar un registro a través de la App, reduciendo de esta manera el tiempo en que la información es conocida por el área de rectificación de máquinas. El objetivo del desarrollo de esta aplicación consiste en agilizar la toma de datos en la etapa de producción del departamento de calidad.

De esta manera la empresa contará con información reciente y actualizada en todo momento permitiéndole hacer los cambios necesarios en menor tiempo y evitar pérdidas excesivas. A la vez identificar qué tipo de errores son más comunes, con

qué frecuencia suceden, determinar con qué operador ocurren frecuentemente los errores, etc.

Dedicatoria

A mi esposa María Mercedes González Mairena y a mi hija Camila Sofía Bárcenas González, son la motivación a seguir adelante y luchar cada día.

Agradecimientos

A Dios, por darnos la vida y ser fuente de inspiración en el tiempo de nuestros estudios.

A mi esposa por apoyarme incondicionalmente durante todo este tiempo. Sin su apoyo no hubiera culminado este estudio de maestría.

A mis padres por haber confiado en mí a lo largo de estos años.

A todos mis amigos y compañeros de trabajo por sus buenos deseos y el apoyo brindado para salir adelante.

A mi tutor Msc. Julio Cesar González Moreno por su atención y tiempo brindado durante el transcurso de la maestría.

Índice

Índice de ilustraciones

1.Introducción

El presente trabajo está dirigido a mitigar problemas con el manejo de información de la empre TIDE-Manufacturing, la cual es una zona franca textil ubicada en la capital Managua – Nicaragua. Esta empresa cuenta con una gran cantidad de empleados todas sus áreas. El área de interés en este trabajo es el área de calidad que cuenta con 15 auditores que dedican todo su tiempo a revisar las prendas.

Esta área genera gran cantidad de datos los cuales deben de ser almacenados. Primero el auditor revisa las piezas buscando imperfecciones, si encuentra alguna debe anotarlo manualmente en un formato en papel. Luego estos documentos son digitalizados.

Para ello la empresa cuenta con un sistema de información de escritorio para guardar estos datos. Este sistema cuenta con un módulo de informes mostrando información detallada de los datos generados en el área de calidad.

2. Planteamiento del problema

El problema de toda empresa moderna se basa en la eficacia de procesar su información, este problema lo tiene TIDE-Manufacturing en el departamento de control de calidad de producción la información asociada con el proceso de producción demora en llegar hasta la base de datos implicando un retraso en la reparación de las máquinas involucradas.

Otro problema es que al realizar una tarea 100% manual además de ser tedioso, ineficiente y muy tardado, no proporciona integridad en los datos obtenidos, es decir, no se sabe si los datos son correctos. Relacionado con este problema está el hecho que todo proceso está sujeto a errores humanos perjudicando así la calidad de los productos entregados o el tiempo de entrega de los productos. Hasta la fecha de realización del presente trabajo no existe un sistema que realice un análisis sobre el porcentaje de errores generados o que reduzca el tiempo de los proceso del departamento de control de calidad.

Para contribuir con la solución de este problema se debe desarrollar una aplicación móvil para la toma de datos y análisis de la información proveniente de los procesos control de calidad en la producción. Además la App deberá validar datos como código de defecto encontrado, código de ubicación de defecto, código de pedido etc.

Esta aplicación se ejecutará sobre una Tablet con sistema operativo Android [1]y se desarrollara bajo IDE Eclipse. La aplicación mostrará gráficos obteniendo la

[1] Android es un sistema operativo con una plataforma abierta para dispositivos móviles adquirido por Google y la Open Handset Allience, su finalidad es satisfacer la necesidad de los operadores móviles y fabricantes de dispositivos. [11]

información de los procesos de control de calidad apoyando de esta manera en la toma de decisiones de la empresa.

3. Antecedentes

Cuando se requiere tomar datos en un área de trabajo referida a procesos de fabricación es complicado utilizar un computador debido a muchos factores (Peso, carga de energía, movilidad) en estos caso es necesario utilizar otro tipo de dispositivo electrónico que pueda realizar la misma función. Un dispositivo móvil es el más adecuado para este trabajo y es utilizado para tomas datos de **encuestas en campo** en el sector salud, realizar ventas y pedidos, etc.

Dispositivos como teléfonos tienen el potencial de optimizar la recolección de los datos en campo, facilitando la labor de los Operadores/trabajadores/empleados. Pueden permitir además la conformación de bases de datos que posteriormente pueden ser asociadas a la información.

Por otra parte, los dispositivos móviles han tenido un desarrollo importante en los últimos dos o tres años. Hoy es frecuente que cuenten con capacidades de localización e interfaces amigables al usuario –como es el caso de Iphone, Blackberry y los dispositivos con sistema operativo Android– a precios significativamente menores que en el pasado reciente, lo que los pone al alcance de un grupo mayor de la población. Paralelamente, el sistema operativo Android de Google ha forzado a muchos fabricantes a liberar herramientas de desarrollo y, en algunos casos, sistemas operativos –como es el caso de Symbian–, lo que facilita, no solamente el acceso a los dispositivos, sino el desarrollo de aplicaciones complejas sobre ellos. [1]

4. Justificación

Actualmente para la empresa TIDE-Manufacturing es importante mantener su información actualizada principalmente la referida a los procesos de control de calidad. Ya cuenta con un sistema para generar reportes, el problema que esta información tarda mucho tiempo en llegar a la base de datos debido a que todo el proceso de recolección de información es llevada a cabo de forma manual. Este proceso tarda en promedio 3 días lo cual es mucho tiempo perdido.

La empresa cuenta con un sistema informático albergado en un ordenador con sistema operativo Windows y ubicado en la oficina de calidad en el cual se digitalizar los datos, pero el proceso de gestión es muy lento, ya que no se toman los datos desde el momento de la inspección durante la etapa de producción, por lo que, el presente trabajo plantea desarrollar una aplicación móvil para la toma de datos institucionales y que esté relacionado con el sistema de gestión de calidad de la empresa, mediante el uso de la misma base de datos a la vez que se muestran gráficos que permiten analizar el comportamiento referido a los procesos de control de calidad.

El desarrollo de una aplicación móvil para Android vendría a contribuir a una mejor gestión de la información en el departamento de control de calidad en la empresa TIDE-Manufacturing, ya que los datos obtenidos se cargarán directamente a la base de datos empresarial, reduciendo el tiempo que toma este proceso.

Esta aplicación se desarrolla en IDE Eclipse [2], el cual permite desarrollar aplicaciones para sistema operativo Android. Actualmente el 80% de los dispositivos

[2] Eclipse es plataforma de desarrollo abierta formada por marcos extensibles, herramientas y tiempos de ejecución para la construcción, implementación y administración de software a través del ciclo de vida. [4]

en el mercado ejecutan sistema operativo Android, así la empresa tendrá una amplia gama de marcas y tamaños a la hora de elegir el dispositivo a usar.

Contará con formularios para la captura de datos y la validación de datos, configuración del sistema para guardar datos locales o remotos permitiendo trabajar con o sin conexión al servidor en caso de que exista un problema técnico en la empresa.

La App mostrará estadísticas sobre defectos comunes encontrados en el proceso de control de calidad, llevará registro sobre los usuarios que utilizan la App de esta manera se conocerá que datos ha introducido cada empleado. Se necesitará una Tablet con SO Android para realizar pruebas a lo largo del desarrollo de la misma.

5. Objetivos

5.1. Objetivo General

Desarrollar una aplicación móvil para dispositivos Android bajo plataforma IDE Eclipse para automatizar la toma de datos y análisis de la información asociada al proceso de control de calidad en la etapa de producción de la empresa TIDE-Manufacturing en periodo enero a julio 2015

5.2. Objetivos específicos

- Identificar los procesos e información que el departamento de calidad de producción requiere para toda la operación, analizando el sistema de información actual.
- Diseñar las interfaces de la aplicación con IDE Eclipse.
- Diseñar gráficos de barra y lineales para el análisis de la información.

Resultados esperados

El siguiente apartado se describe los resultados que se deben obtener una vez finalizado todas las etapas del proyecto.

- Documento de requisitos de los procesos de control de calidad de producción esto servirá para saber las necesidades del usuario, qué información quiere gestionar y por consiguiente el correcto diseño de las interfaces.
- Diseñar las interfaces de la aplicación las cuales permitirán especificar como va a interactuar la aplicación con el usuario y con el servidor.
- Diseñar una aplicación en Asp.net con funciones públicas que permita comunicar la aplicación móvil con el servidor de base de datos.
- Realizada la codificación de la aplicación, esta se aplicará de acuerdo a las interfaces diseñadas, permitirá interactuar con el usuario para la introducción de datos y con el servidor para validad y guardar estos datos.
- Realizadas pruebas de la aplicación, se realizara un informe donde se especificarán los errores de código y diseño que los usuarios han encontrado. Así como recomendaciones o mejoras (Ver anexos).

6. Marco Teórico

6.1. Android

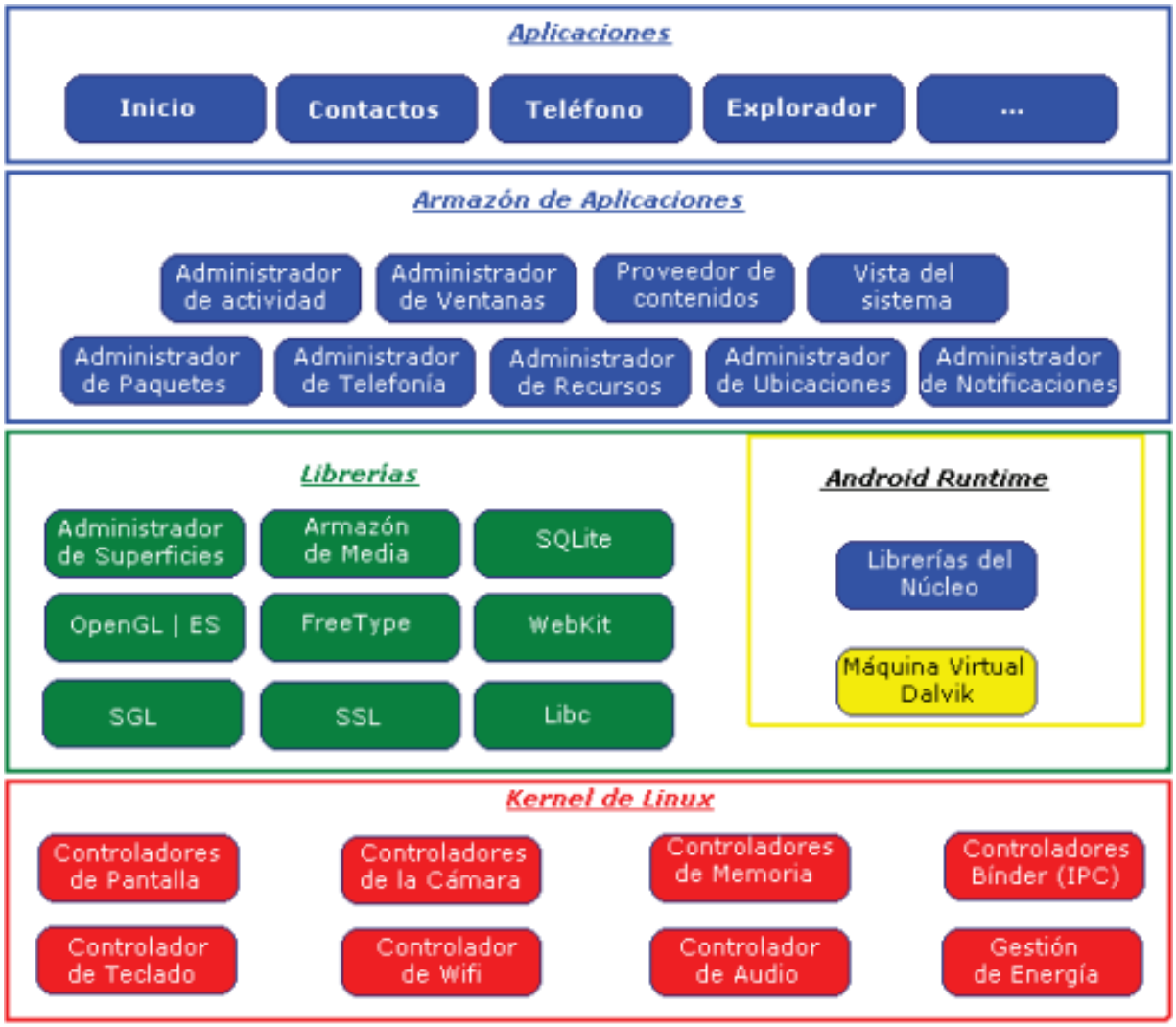
Como hemos comentado, existen muchas plataformas para móviles (iPhone, Symbian, Windows Phone, BlackBerry, Palm, Java Mobile Edition, Linux Mobile (LiMo), etc.); sin embargo Android presenta una serie de características que lo hacen diferente. Es el primero que combina, en una misma solución, las siguientes cualidades:

- **Plataforma realmente abierta** : Es una plataforma de desarrollo libre basada en Linux y de código abierto. Una de sus grandes ventajas es que se puede usar y "customizar" el sistema sin pagar royalties.
- **Portabilidad asegurada** :. Las aplicaciones finales son desarrolladas en Java, lo que nos asegura que podrán ser ejecutadas en una gran variedad de dispositivos, tanto presentes como futuros. Esto se consigue gracias al concepto de máquina virtual.
- **Arquitectura basada en componentes inspirados en Internet**. Por ejemplo, el diseño de la interfaz de usuario se hace en XML, lo que permite que una misma aplicación se ejecute en un móvil de pantalla reducida o en un netbook.
- **Gran cantidad de servicios incorporados**
- **Gran cantidad de servicios incorporados** : Por ejemplo, localización basada tanto en GPS como en redes, bases de datos con SQL, reconocimiento y síntesis de voz, navegador, multimedia, etc.
- **Aceptable nivel de seguridad.** Los programas se encuentran aislados unos de otros gracias al concepto de ejecución dentro de una caja que hereda de Linux. Además, cada aplicación dispone de una serie de

permisos que limitan su rango de actuación (servicios de localización, acceso a Internet, etc.).

- **Optimizado para baja potencia y poca memoria:** Por ejemplo, Android utiliza la Máquina Virtual Dalvik. Se trata de una implementación de Google de la máquina virtual de Java optimizada para dispositivos móviles.
- **Alta calidad de gráficos y sonido**. Gráficos vectoriales suavizados, animaciones inspiradas en Flash, gráficos en 3 dimensiones basados en OpenGL. Incorpora los códecs estándar más comunes de audio y vídeo, incluyendo H.264 (AVC), MP3, AAC, etc.

6.1.1. Arquitectura Android



Ilustracion 1 Nucleo del sistema operativo Android

El núcleo Linux

El núcleo de Android está formado por el sistema operativo Linux, versión 2.6. Esta capa proporciona servicios como la seguridad, el manejo de la memoria, el multiproceso, la pila de protocolos y el soporte para dispositivos. Esta capa del modelo actúa como capa de abstracción entre el hardware y el resto de la pila. Por lo tanto, es la única que es dependiente del hardware.

Runtime de Android

Está basado en el concepto de máquina virtual utilizado en Java. Dado las limitaciones de los dispositivos donde ha de ejecutarse Android (poca memoria y procesador limitado) no fue posible utilizar una máquina virtual Java estándar. Google tomó la decisión de crear una nueva, la máquina virtual Dalvik, que respondiera mejor a estas limitaciones.

Algunas características de la máquina virtual Dalvik que facilitan esta optimización de recursos son: que ejecuta ficheros Dalvik ejecutables (.dex) (formato optimizado para ahorrar memoria). Además, está basada en registros. Cada aplicación corre en su propio proceso Linux con su propia instancia de la máquina virtual Dalvik. Delega al kernel de Linux algunas funciones como threading y el manejo de la memoria a bajo nivel.

También se incluye en el Runtine de Androidel "core libraries" con la mayoría de las librerías disponibles en el lenguaje Java.

Librerías nativas

Incluye un conjunto de librerías en C/C++ usadas en varios componentes de Android. Están compiladas en el código nativo del procesador. Muchas de las librerías utilizan proyectos de código abierto. Algunas de estas librerías son:

- System C library: una derivación de la librería BSD de C estándar (libe), adaptada para dispositivos embebidos basados en Linux.
- Media Framework: librería basada en PacketVideo's OpenCORE; soporta codees de reproducción y grabación de multitud de formatos de audio, vídeo e imágenes MPEG4, H.264, MP3, AAC, AMR, JPG y PNG.
- Surface Manager: maneja el acceso al subsistema de representación gráfica en 2D y 3D.

- WebKit: soporta un moderno navegador web utilizado en el navegador Android y en la vista webview. Se trata de la misma librería que utiliza Google Chrome y Safari de Apple.
- SGL: motor de gráficos 2D.
- Librerías 3D: implementación basada en OpenGL ES 1.0 API. Las librerías utilizan el acelerador hardware 3D si está disponible, o el software altamente optimizado de proyección 3D.
- FreeType: fuentes en bitmap y renderizado vectorial. • SQLite potente y ligero motor de bases de datos relaciónales disponible para todas las aplicaciones.
- SSL: proporciona servicios de encriptación Secure Socket Layer. [2]

6.2. Componentes de una aplicación

Para diseñar una aplicación en Android, es necesario tener claros los elementos que la componen y la funcionalidad de cada uno de ellos. Ya hemos visto el ejemplo del “Hola Android”, por lo que podemos intuir algunos de ellos. Uno de los aspectos más importantes a tener en cuenta es su funcionamiento.

Android trabaja en Linux, y cada aplicación utiliza un proceso propio. Se distinguen por el ID, un identificador para que solo ella tenga acceso a sus archivos. Los dispositivos tienen un único foco, la ejecución principal, que es la aplicación que está visible en la pantalla, pero puede tener varias aplicaciones en un segundo plano, cada una con su propia pila de tareas. La pila de tareas es la secuencia de ejecución de procesos en Android.

Se componen de actividades que se van apilando según son invocadas, y solo pueden terminarse cuando las tareas que tiene encima están terminadas, o cuando el sistema las destruye porque necesita memoria, por lo que tienen que estar

preparadas para terminar en cualquier momento. El sistema siempre eliminará la actividad que lleve más tiempo parada. En caso de que el sistema necesitase mucha memoria, si la aplicación no está en el foco, puedeser eliminada por completo a excepción de su actividad principal.

Ilustracion 2 Actividades en Android

Una de las características principales del diseño en Android es la reutilización de componentes entre las aplicaciones, es decir, dos aplicaciones diferentes pueden utilizar una misma componente, aunque esté en otra aplicación para así, evitar la repetición innecesaria de código, y la consiguiente ocupación de espacio. Los componentes son los elementos básicos con los que se construye el proyecto. Hay cuatro tipos, pero las aplicaciones se componen principalmente de actividades. Habrá tantas actividades como ventanas distintas tenga la aplicación. Sin embargo, por si solos, los componentes no pueden hacer funcionar una aplicación. Para ello están los intents.

Todos ellos deben declararse en el AndroidManifest.xml (junto con otros elementos que se mostrarán después) con el mismo nombre que lleve la

clase asociada. Por ejemplo, la clase MainActivity, será definida en el AndroidManifest con el mismo nombre.

Actividades

Una actividad (o Activity) es el componente principal encargado de mostrar al usuario la interfaz gráfica, es decir, una actividad sería el equivalente a una ventana, y es el medio de comunicación entre la aplicación y el usuario. Se define una actividad por cada interfaz del proyecto. Los elementos que se muestran en ella deben ser definidos en el fichero xml que llevan asociado (que se guarda en ./res/layout) para poder ser tratados enla clase NameActivity.class, que hereda de la clase Activity.

Dentro del fichero xml asociado a la actividad, se definen los elementos tales como ubicación de los elementos en la pantalla (layouts), botones, textos, checkbox, etc. Las actividades tienen un ciclo de vida, es decir, pasan por diferentes estados desde que se inician hasta que se destruyen. Sus 3 posibles estados son:

Activo: ocurre cuando la actividad está en ejecución, es decir, es la tarea principal

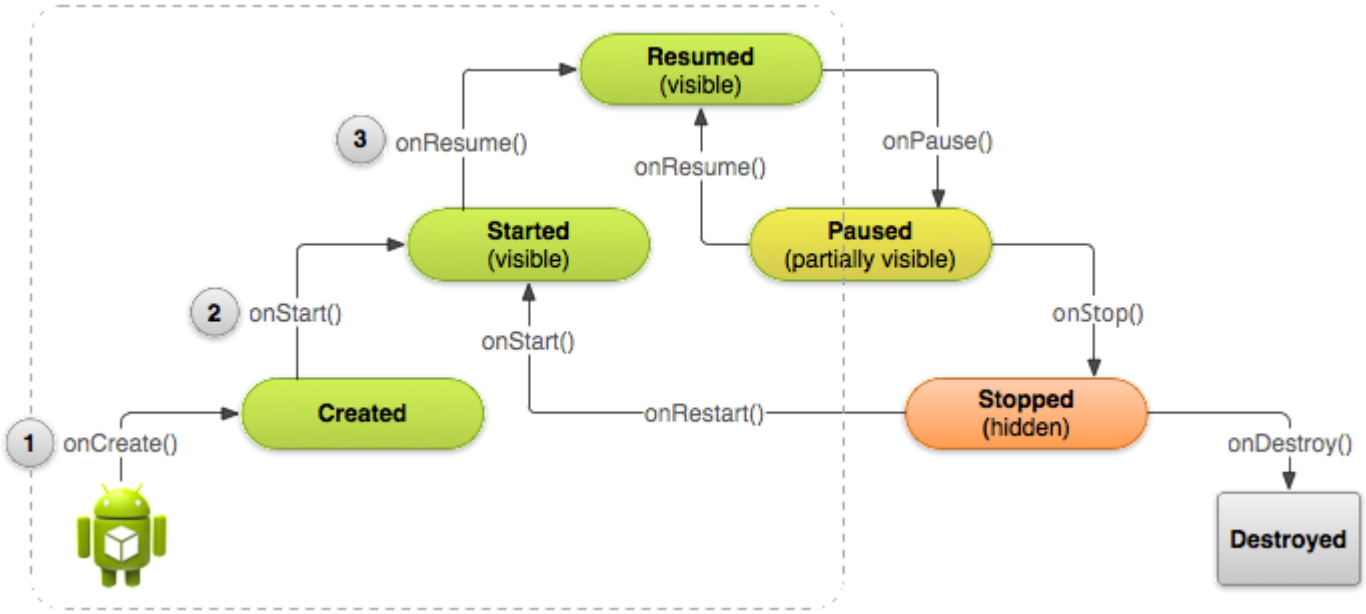
Pausado: la actividad se encuentra semi-suspendida, es decir, aun se está ejecutando y es visible, pero no es la tarea principal. Se debe guardar la información en este estado para prevenir una posible pérdida de datos en caso de que el sistema decida prescindir de ella para liberar memoria.

Parado: la actividad está detenida, no es visible al usuario y el sistema puede liberar memoria. En caso de necesitarla de nuevo, será reiniciada desde el principio.

Una vez definido el ciclo de vida, hay que tener en cuenta qué métodos son importantes en cada uno de ellos. Aquí están los métodos más importantes de una actividad:

- **OnCreate** (Bundle savedInstanceState): es el método que crea la actividad. Recibe un parámetro de tipo Bundle, que contiene el estadoanterior de la actividad, para preservar la información que hubiera, en caso de que hubiera sido suspendida, aunque también puede iniciarse con un null si la información anterior no es necesaria o no existe.
- **OnRestart():** reinicia una actividad tras haber sido parada (si continúa en la pila de tareas). Se inicia desde cero.
- **Onstart():** inmediatamente después de onCreate(Bundle savedInstanceState), o de
- **OnRestart()** según corresponda. Muestra al usuario la actividad. Si ésta va a estar en un primer plano, el siguiente método debe ser onResume(). Si por el contrario se desarrolla por debajo, el método siguiente será onStop(). Es recomendable llamar al método onRestoreInstanceState() para asegurar lainformación

- **OnResume():** establece el inicio de la interactividad entre el usuario y la aplicación. Solo se ejecuta cuando a actividad está en primer plano. Si necesita información previa, el método onRestoreInstanceState() aportarála situación en que estaba la actividad al llamar al onResume(). También puede guardar el estado con onSaveInstanceState().
- **OnPause():** se ejecuta cuando una actividad va a dejar de estar en primer plano, para dar paso a otra. Guarda la información, para poder restaurar cuando vuelva a estar activa en el método onSaveInstanceState(). Si la actividad vuelve a primer plano, el siguiente método será onResume(). En casocontrario, será onStop().
- **OnStop():** la actividad pasa a un segundo plano por un largo período. Como ya se ha dicho, el sistema puede liberar el espacio que ocupa, en caso de necesidad, o si la actividad lleva parada mucho tiempo.
- **OnDestroy():** es el método final de la vida de una actividad. Se llama cuando ésta ya no es necesaria, o cuando se ha llamado al método finish().



Ilustracion 3 Ciclo de vida de las actividades en un proyecto Android

Además de estos métodos, cabe destacar dos más, queson de vital importancia:

OnSavedInstanceState(): guarda el estado de una actividad. Es muy útil cuando se va a pausar una actividad para abrir otra.

OnRestoreInstanceState(): restaura los datos guardados en onSavedInstanceState() al reiniciar una actividad. [3]

6.3. Eclipse IDE

Eclipse es una comunidad de personas y organizaciones que colaboran en el comercio de usar software de código abierto. Sus proyectos se centran en la construcción de una plataforma de desarrollo abierta formada por marcos extensibles, herramientas y tiempos de ejecución para la construcción, despliegue y gestión de software. La Fundación Eclipse es una organización sin fines de lucro, y ayuda a cultivar tanto una comunidad de código abierto y un ecosistema de productos y servicios complementarios.

El Proyecto Eclipse fue creado originalmente por IBM en noviembre de 2001 y con el apoyo de un consorcio de proveedores de software. La Fundación Eclipse fue creada en enero de 2004 como una corporación independiente sin fines de lucro para que actúe como administrador de la comunidad Eclipse. La corporación independiente sin fines de lucro fue creada para permitir que un proveedor neutral y abierto, comunitario transparente para establecerse alrededor de Eclipse. Hoy en día, la comunidad Eclipse consiste en individuos y organizaciones de una sección transversal de la industria del software.

En general, la Fundación Eclipse ofrece cuatro servicios a la comunidad Eclipse: **Infraestructura de TI** , **Gestión IP, Proceso de Desarrollo** , y **Desarrollo del Ecosistema** . Personal de tiempo completo están asociados con cada una de estas áreas y trabajar con la comunidad en Eclipse para ayudar a satisfacer las necesidades de los grupos de interés.

6.3.1. Infraestructura de TI

La Fundación Eclipse gestiona la infraestructura de TI para la comunidad de código abierto Eclipse, incluyendo repositorios Git código, bases de datos Bugzilla, orientadas al desarrollo de listas de correo y foros, descarga web y el sitio web. La infraestructura está diseñada para proporcionar un servicio confiable y escalable

para las committers en desarrollo la tecnología Eclipse y los consumidores que utilizan la tecnología.

6.3.2. Gestión de la Propiedad Intelectual (IP)

Un aspecto importante de Eclipse es el objetivo de facilitar el uso de la tecnología de código abierto en los productos y servicios de software comerciales. Conscientemente promover y alentar a los proveedores de software para utilizar la tecnología de Eclipse para la construcción de sus productos y servicios de software comerciales. Esto es posible por el hecho de que todos los proyectos de Eclipse están licenciados bajo la **Eclipse Public License (EPL)** .

La Fundación Eclipse también lleva a cabo una serie de medidas para tratar de garantizar el pedigrí de la propiedad intelectual contenida dentro de los proyectos de Eclipse. El primer paso en el proceso de debida diligencia está tratando de garantizar que todas las contribuciones son hechas por el propietario de los derechos que le corresponde y bajo la Licencia Pública Eclipse (EPL). Se requiere que todos committers a firme un **acuerdo de committer** que estipula la to quetalidad de sus contribuciones son su trabajo original y están siendo aportados en el marco del EPL. Si un confirmador es patrocinado a trabajar en un proyecto de Eclipse por una organización miembro, después de que la organización se le pide que firme un **Acuerdo Committer miembros** para asegurar los derechos de propiedad intelectual de la organización son aportados bajo el EPL.

El segundo paso es que el código fuente en relación con todas las contribuciones que se desarrollan fuera del proceso de desarrollo Eclipse se procesan a través de la Fundación Eclipse **proceso de aprobación IP** . Este proceso incluye el análisis de las contribuciones de código seleccionado para tratar de determinar la procedencia del código, y la compatibilidad de licencia con el EPL. Contribuciones que contienen código licenciado bajo licencias no compatibles con el EPL están destinados a ser examinados a través de este proceso de aprobación y por lo tanto no se añade a un proyecto de Eclipse. El resultado final es un nivel de confianza de

que Eclipse es una tecnología de liberación proyectos de código abierto que se puede distribuir de forma segura en los productos comerciales.

6.3.3. Apoyo a la Comunidad para el Desarrollo

La comunidad Eclipse tiene una bien ganada reputación por proporcionar software de calidad de una manera fiable y predecible. Esto se debe al compromiso de los committers y organizaciones que contribuyen a los proyectos de código abierto. La Fundación Eclipse también proporciona servicios y apoyo a los proyectos para ayudarles a alcanzar estos objetivos.

El personal de la Fundación ayuda a implementar el **proceso de desarrollo Eclipse.** Este proceso ayuda a la nueva puesta en marcha del proyecto y asegura que todos los proyectos de Eclipse se ejecutan de una manera abierta, transparente y meritocrático. Como parte de este proceso, la Fundación organiza una comunidad de miembros críticas para proyectos para asegurar la interacción constante entre los proyectos y los miembros en general.

La comunidad Eclipse organiza un tren de lanzamiento anual que proporciona una liberación coordinada de los proyectos de Eclipse que deseen participar. El tren de liberación hace que sea más fácil para los consumidores a adoptar las nuevas versiones de los proyectos porque: 1) todos los proyectos están disponibles en el mismo horario, por lo que no tiene que esperar a programas de proyectos independientes, y 2) un nivel de pruebas de integración se produce antes del lanzamiento final para ayudar a identificar los problemas del proyecto.

6.3.4. Desarrollo de Ecosistemas

Un aspecto único de la comunidad Eclipse y el papel de la Fundación Eclipse es la comercialización activa y la promoción de proyectos de Eclipse y más amplio ecosistema Eclipse. Un ecosistema saludable y vibrante que se extiende más allá de la comunidad Eclipse de código abierto para incluir cosas como productos comerciales basados en Eclipse, otros proyectos de código abierto utilizando

Eclipse, capacitación y servicios de los proveedores, revistas y portales de Internet, libros, etc, son todos clave para el éxito de la comunidad Eclipse.

Para ayudar en el desarrollo del ecosistema Eclipse, Eclipse Foundation organiza una serie de actividades, incluyendo eventos de marketing de cooperación con las empresas miembros, **conferencias comunitarias** , catálogos de recursos en línea (**Eclipse Marketplace** y el **canal de YouTube de Eclipse**), reuniones Miembros bianuales y otros programas para promover toda la comunidad Eclipse.

6.3.5. Un modelo único para el Desarrollo de Código Abierto

La Fundación Eclipse ha sido establecido para servir a los proyectos de código abierto Eclipse y la comunidad Eclipse. Como una corporación independiente sin fines de lucro, la Fundación y el modelo de gobierno Eclipse que garantiza ninguna entidad es capaz de controlar la estrategia, las políticas o las operaciones de la comunidad Eclipse.

La Fundación se centra en crear un entorno para proyectos de código abierto exitosas y promover la adopción de la tecnología de Eclipse en soluciones de código abierto y comerciales., el modelo Eclipse de desarrollo de código abierto es un modelo único y comprobado para desarrollo de código abierto. [4]

6.4. ASP.Net

Según Microsoft, "ASP.NET es una tecnología para la construcción de potentes aplicaciones web dinámicas, y forma parte de .NET Framework".

.NET Es independiente del lenguaje, lo que significa que puede utilizar cualquier lenguaje .NET para el desarrollo de las aplicaciones web .NET . Los lenguajes más comunes para el desarrollo de aplicaciones ASP.NET son C # y VB.NET. Mientras VB.NET se basa directamente VB (Visual Basic), C # se introdujo junto con el marco .NET, y por lo tanto es algo relativamente nuevo .Algunas personas llaman a C # "el lenguaje .NET", pero de acuerdo a Microsoft, se pueden hacer las mismas cosas, no importa si usted está haciendo uso de C # o VB.NET. Los 2 lenguajes no son tan diferentes, y se ha usado uno de ellos, no tendrán ningún problema en absoluto aprendizaje del otro.

6.5. Visual Studio 2010

En el mercado existen muchas herramientas para desarrollar software para la plataforma Microsoft .Net. De cualquier manera, la herramienta por excelencia es la que provee Microsoft bajo el nombre de Visual Studio. Microsoft Visual Studio 2010 cuenta con diferentes sub versiones, cada una de estas orientadas a distintas áreas de la ingeniería y el desarrollo de software, por lo que encontraremos versiones para la realización de pruebas, para la creación de código, o incluso para el desarrollo de la arquitectura de una aplicación.

Es importante elegir una versión adecuada al trabajo que se va a realizar por lo que caso tomaremos la más potente de todas las versiones, la versión Ultimate de Visual Studio 2010. Por otra parte, algo que debe tener en claro es que para poder desarrollar aplicaciones bajo la tecnología Microsoft .Net no es necesario contar con este tipo de herramientas ya que estas sólo son un IDE (Integrated Development Environment o, ambiente integrado de desarrollo) de desarrollo. Esto quiere decir que sólo nos provee atajos para diferentes actividades que podríamos hacer directamente desde una consola de comandos, o mediante cualquier otra herramienta. Por supuesto, si bien la tecnología, como el lenguaje de programación utilizado no está atado al IDE, en la actualidad, la complejidad de los mismos hace que el tener una de estas herramientas simplifique mucho las tareas de programación. [5]

7. Diseño metodológico

La empresa TIDE- Manufacturing que se encuentra ubicada en la ciudad de Managua, Nicaragua es una empresa textil que confecciona ropa deportiva para exportación. Par obtener el producto final la materia prima debe pasar por varios procesos cada uno muy importante. Algunos de estos procesos lo realizan máquinas (Corte) y otros lo realizan el personal de la empresa (Costura). Algunos de estos procesos se pueden automatizar utilizando sistemas de información pero otros que se realizan directamente en la planta no es viable utilizar computadores.

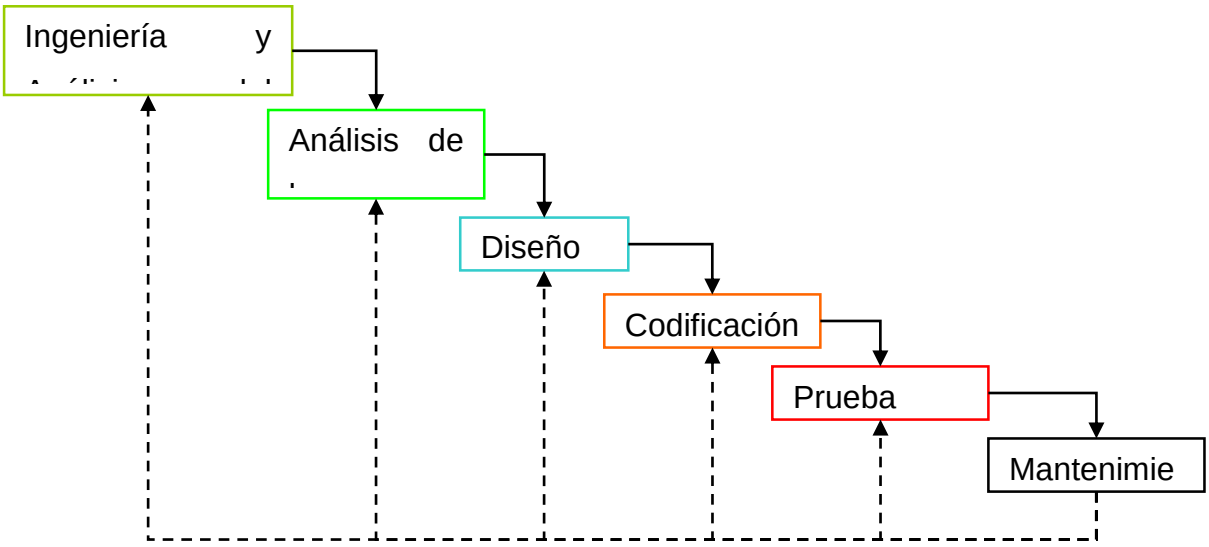
Para estos casos conviene usar dispositivos portátiles (Teléfonos inteligentes, Tablet) ya que pueden estar sin conexión a energía por muchas horas además son más fáciles de manipular y trasladar. Uno de estos procesos es el de control de calidad de producción.

Para realizar el presente trabajo primero se necesita conocer cómo funciona el departamento de control de calidad de producción, así como también cada uno de los subprocesos. Además de eso se necesita saber qué información manipulan y cómo la procesan.

De esta manera tendremos una idea clara de cómo automatizar estos procesos. Una vez que tengamos claro el funcionamiento de este departamento y la información con la que se trabaja, se procederá realizar un documento de **Especificación de requisitos[3].** El cual será una guía a la hora de desarrollar la aplicación.

[3] El ERS son especificaciones para un producto del software en particular, programa, o juego de programas que realizan ciertas funciones en un ambiente específico. El

Luego se procederá diseñar las interfaces tomando como referencia la especificación de requisito. A partir de las interfaces se iniciará con la codificación de aplicación. El desarrollo de la aplicación se realizará siguiendo el modelo en cascada[4], el paradigma del ciclo de vida abarca las siguientes actividades.



Ilustracion 4 Ciclo de vida en cascada

ERS puede escribirse por uno o más representantes del proveedor, uno o más representantes del cliente, o por ambos. [13]

MAESTRÍA EN TECNOLOGÍA DE LA INFORMÁTICA EMPRESARIAL

[4] Este modelo establece que las diversas actividades que se van realizando al desarrollar un producto software, se suceden de forma lineal. Los modelos de ciclo de vida del software describen las fases del ciclo de software y el orden en que se ejecutan las fases. [12]

Debido a que el software es siempre parte de un sistema mayor el trabajo comienza estableciendo los requisitos de todos los elementos del sistema y luego asignando algún subconjunto de estos requisitos al software.

Análisis de los requisitos del software: el proceso de recopilación de los requisitos se centra e intensifica especialmente en el software. El ingeniero de software (Analistas) debe comprender el ámbito de la información del software, así como la función, el rendimiento y las interfaces requerida

Diseño: el diseño del software se enfoca en cuatro atributos distintos del programa: la estructura de los datos, la arquitectura del software, el detalle procedimental y la caracterización de la interfaz. El proceso de diseño traduce los requisitos en una representación del software con la calidad requerida antes de que comience la codificación.

Codificación: el diseño debe traducirse en una forma legible para la maquina. El paso de codificación realiza esta tarea. Si el diseño se realiza de una manera detallada la codificación puede realizarse mecánicamente.

Prueba: una vez que se ha generado el código comienza la prueba del programa. 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.

Mantenimiento: el software sufrirá cambios después de que se entrega al cliente. Los cambios ocurrirán debido a que hayan encontrado errores, a que el software

deba adaptarse a cambios del entorno externo (sistema operativo o dispositivos periféricos), o debido a que el cliente requiera ampliaciones funcionales o del rendimiento.

Una vez desarrollada la aplicación se debe poner a prueba en un entorno real, el cual servirá para identificar algún requerimiento no especificado o error en diseño o código.

El modelo en cascada, a veces llamado ciclo de vida clásico, sugiere un enfoque sistemático y secuencial para el desarrollo del software, que comienza con la especificación de los requerimientos por parte del cliente y avanza a través de planeación, modelado, construcción y despliegue, para concluir con el apoyo del software terminado.

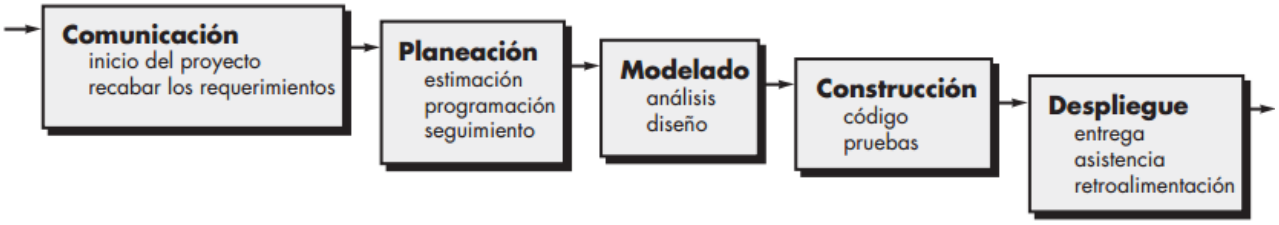
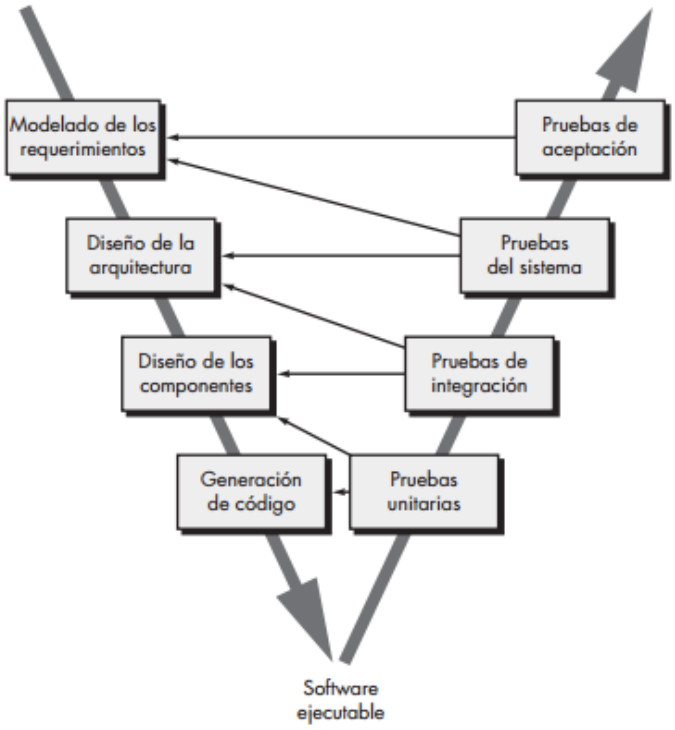


Figura: Modelo en cascada

Una variante de la representación del modelo en cascada se denomina modelo en V. donde se aprecia la relación entre las acciones para el aseguramiento de la calidad y aquellas asociadas con la comunicación, modelado y construcción temprana.



Ilustracion 5 Ciclo de vida Modeo en V

A medida que el equipo de software avanza hacia abajo desde el lado izquierdo de la V, los requerimientos básicos del problema mejoran hacia representaciones técnicas cada vez más detalladas del problema y de su solución. Una vez que se ha generado el código, el equipo sube por el lado derecho de la V, y en esencia ejecuta una serie de pruebas (acciones para asegurar la calidad) que validan cada uno de los modelos creados cuando el equipo fue hacia abajo por el lado izquierdo.

En realidad, no hay diferencias fundamentales entre el ciclo de vida clásico y el modelo en V. Este último proporciona una forma de visualizar el modo de aplicación de las acciones de verificación y validación al trabajo de ingeniería inicial.

El modelo de la cascada es el paradigma más antiguo de la ingeniería de software. Sin embargo, en las últimas tres décadas, las críticas hechas al modelo han ocasionado que incluso sus defensores más obstinados cuestionen su eficacia. Entre los problemas que en ocasiones surgen al aplicar el modelo de la cascada se encuentran los siguientes:

1) Es raro que los proyectos reales sigan el flujo secuencial propuesto por el modelo. Aunque el modelo lineal acepta repeticiones, lo hace en forma indirecta. Como resultado, los cambios generan confusión conforme el equipo del proyecto avanza.
2) A menudo, es difícil para el cliente enunciar en forma explícita todos los requerimientos. El modelo de la cascada necesita que se haga y tiene dificultades para aceptar la incertidumbre natural que existe al principio de muchos proyectos.
3) El cliente debe tener paciencia. No se dispondrá de una versión funcional del (de los) programa(s) hasta que el proyecto esté muy avanzado. Un error grande sería desastroso si se detectara hasta revisar el programa en funcionamiento.

Hoy en día, el trabajo de software es acelerado y está sujeto a una corriente sin fin de cambios (en las características, funciones y contenido de información). El modelo de la cascada suele ser inapropiado para ese tipo de labor. No obstante, sirve como un modelo de proceso útil en situaciones en las que los requerimientos son fijos y el trabajo avanza en forma lineal hacia el final. [6]

8. Especificación de Requisito Software

8.1. Introducción

El siguiente apartado especifica las funciones que tendrá el sistema así como el alcance del mismo.

8.1.1. Propósito

La especificación de requisito software tiene como propósito definir las especificaciones funcionales, no funcionales del sistema para una aplicación móvil que permitirá la gestión de la información en el área de control de calidad de la empresa Tide – Manufactiring, la cual será usada por el personal de la misma área.

8.1.2. Alcance del sistema

En la empresa TIDE-Manufacturing se ha tenido la necesidad de crear una app para captura de datos en el área de calidad. Por lo tanto el objetivo del presente trabajo es desarrollar una aplicación móvil que permita capturar datos de manera rápida y segura, a través de interfaces intuitivas. A la vez que permita al gerente del área de calidad supervisar dicho proceso, es decir, saber cuánto trabajo ha realizado cada trabajador, que tipo de errores o problemas más frecuentes etc.

8.1.3. Definiciones, Acrónimos y Abreviaturas

En el siguiente apartado se especificara y definirá todos los términos usados en la especificación de requisito.

Definición

Auditoria: en el área de calidad cada vez que un empleado (Auditor) revisa una cierta cantidad de cajas y piezas.

Auditor: Personal del área de control de calidad.

Incidencia: Cada vez que el auditor encuentra un error debe guardar esa información, a esto se le llama incidencia.

Interfaces: formularios o ventanas que permiten a los usuarios interactuar con un dispositivo electrónico e ingresar datos a través de la solución diseñada.

Sitio web: Se refiere a la página web desarrollada para interactuar con la aplicación móvil.

CuttinTicket: Orden de compra del pedido.

8.1.4. Visión general del documento

Este documento consta de tres secciones. En la primera sección se realiza una introducción al mismo y se proporciona una visión general de la especificación de recursos del sistema.

En la segunda sección del documento se realiza una descripción general del sistema, con el fin de conocer las principales funciones que éste debe realizar, los datos asociados y los factores, restricciones, supuestos y dependencias que afectan al desarrollo, sin entrar en excesivos detalles.

Por último, la tercera sección del documento es aquella en la que se definen detalladamente los requerimientos que debe satisfacer el sistema.

8.2. Descripción General

Se describirán las funciones que tendrán el sistema, origen de dato de cada funcionalidad, el proceso a seguir y la salida que generara cada función.

8.2.1. Perspectiva del Producto

Este producto se utilizará durante el proceso de control de calidad de las prendas. Cada vez que se encuentre un defecto en la confección de una de las prendas el usuario introducirá los datos correspondientes a este defecto.

8.2.2. Funciones del Producto

La aplicación móvil permitirá realizar lo siguiente:

Registrar Auditoria
Registrar incidencia
Editar incidencia
Eliminar incidencia
Visualizar cajas revisadas por los trabajadores
Visualizar piezas revisadas por los trabajadores
Visualizar la cantidad de un error específico por cada semana
Visualizar la cantidad de cajas que revisa cada trabajador por cada semana.
Visualizar la cantidad de piezas que revisa cada trabajador por cada semana.
Inicio de sesión de usuarios

8.2.3. Características de los usuarios

El sistema estará orientado a usuarios que se están familiarizados a trabajar con dispositivos móviles con pantalla táctil. No se necesitará ningún conocimiento avanzado del dispositivo.

8.2.4. Restricciones

1. La aplicación sólo se podrá ejecutara bajo dispositivos con sistema operativo Android.
2. La aplicación debe contar con conexión a la red local de la empresa (específicamente con el servidor(es) de datos) para su correcto funcionamiento y validación de datos.
3. El dispositivo sólo puede trabajar una cierta cantidad de horas sin corriente eléctrica externa, luego tiene que estar conectado a una fuente de energía para trabajar correctamente.

8.3. Requisitos Específicos

A continuación se detallaran todas las funciones que tendrá la aplicación móvil.

RF1: Registrar Auditoria
RF2: Registrar incidencia
RF3: Editar incidencia
RF4: Eliminar incidencia
RF5: Visualizar cajas revisadas por los trabajadores
RF6: Visualizar piezas revisadas por los trabajadores
RF7: Visualizar la cantidad de un error especifico por cada semana
RF8: Visualizar la cantidad de cajas que revisa cada trabajador por cada semana.

RF9: Visualizar la cantidad de piezas que revisa cada trabajador por cada semana.

RF10: Inicio de sesión de usuarios

8.3.1. Requisitos Comunes de las interfaces

A continuación se especificaran los requisitos de usuario, hardware, software y comunicación con que debe contar la aplicación para su correcto uso.

8.3.1.1. Interfaces de usuario

Las interfaces de usuario están relacionadas con las pantallas, ventanas (formularios) que debe manipular el usuario para realizar una operación determinada. Dicha manipulación se realizara por medio de la pantalla táctil del dispositivo (Tablet).

A continuación se muestra una previa de lo que serán las interfaces de usuario.

El usuario previamente debe tener su cuenta de usuario en el sistema para poder acceder. El personal de informática será el encargado de crear las cuentas de usuario a los auditores.

8.3.1.2. Interfaces de hardware

Pantalla táctil del dispositivo: Todas las funcionalidades de la App se realizarán por medio de la pantalla táctil del dispositivo.

Servidor web y servidor de base de datos, ambos son necesarios para que la aplicación funcione correctamente.

8.3.1.3. Interfaces de software

Ninguno

8.3.1.4. Interfaces de comunicación

Esta aplicación móvil deberá guardar registros en una base de datos SQL Server 2012, el problema es que el sistema operativo Android no cuenta con una librería que permita acceder directamente a este tipo de Sistema Gestor de base de datos. Para esto se ha diseñado una página web que contiene una gran cantidad de métodos web públicos que me permitirán acceder al servidor SQL Server.

Esta página web está diseñada en Asp.net con lenguaje de programación C# y deberá estar alojada en un servidor Web IIS. La aplicación móvil enviará y recibirá información a esta página con el fin de guardar datos, consultar datos para validaciones y consultar datos para la generación de gráficos.

8.3.2. Requisitos Funcionales

A Continuación se describirán a detalle las funciones que tendrá la aplicación. Cada una de estas funciones está representada con un único requisito donde se detalla toda la información necesaria sobre la función que realizara el sistema.

8.3.2.1. Requisito funcional 1

Numero de requisito	RF1
Nombre del requisito	Registrar Auditoria
Tipo	Requisito
Fuente del requisito	Tabla QualityPageDet

Introducción

El sistema debe permitir la entrada de datos como idcuttingTicket, iniciales del auditor, cantidad de piezas y cantidad de cajas etc, a la tabla QualityPageDet.

Entrada

IdcuttingTicket,inicials del auditor, cantidad de cajas(QtyCajas), cantidad de piezas (QtyPiezas), cantidad de defectos (QtyDefectos), CantidadR, QtyPiezasTotal.

Proceso

A medida que se vallan ingresando los datos el sistema validará el idcuttingticket si la validación falla resaltará ese campo para que el usuario lo corrija.

Salida

Un mensaje que indica que el registro fue ingresado correctamente.

8.3.2.2. Requisito funcional 2

Numero de requisito	RF2
Nombre del requisito	Registrar incidencia
Tipo	Requisito
Fuente del requisito	Tabla

Introducción

El sistema contará con un formulario en el cual el usuario puede ir agregando en detalle cada uno de los errores que va encontrando, cada uno de estos errores será un registro en la tabla QualityPageLoca.

Entrada

El sistema tendrá una ventana que permite la inserción de los datos, estos son IdQualityCtrlPage, IdStyle, IdDefect, IdDefectLoca.

Proceso

Todos estos valores deben ser validados para que existan en sus tablas correspondientes. Si uno de los campos no pasa la validación el sistema le informará al usuario que hay un problema con ese campo.

Salida

Un mensaje que indica que el registro fue ingresado correctamente.

8.3.2.3. Requisito funcional 3

Numero de requisito	RF3
Nombre del requisito	Editar incidencia
Tipo	Requisito
Fuente del requisito	Tabla QualityPageLoca

Introducción

El sistema cuenta con ventanas o actividades que sirven para la inserción de datos, estas ventanas deben de servir también para editar esos datos.

Entrada

El sistema tendrá una ventana que permita editar los datos, estos son IdQualityCtrlPage, IdStyle, IdDefect, IdDefectLoca.

Proceso

Cuando se carga la ventana para editar los datos ya insertados ya aparecen en esta, los nuevos valores que se introducen deben de ser validados.

Salida

Un mensaje que indica que el registro fue actualizado correctamente.

8.3.2.4. Requisito funcional 4

Numero de requisito	RF4
Nombre del requisito	Eliminar incidencia
Tipo	Requisito
Fuente del requisito	Tabla QualityPageLoca

Introducción

El sistema cuenta con ventanas o actividades que sirven para la inserción, edición y eliminación de detalle de incidencias.

Entrada

El sistema tendrá una ventana que permita eliminar registro de detalle de incidencia, estos son IdQualityCtrlPage, IdStyle, IdDefect, IdDefectLoca.

Proceso

Cuando se carga la ventana para editar los datos ya insertados ya aparecen en esta, los nuevos valores que se introducen deben de ser validados.

Salida

Un mensaje que indica que el registro fue actualizado correctamente.

8.3.2.5. Requisito funcional 5

Numero de requisito	RF5
Nombre del requisito	Visualizar cajas revisadas por los trabajadores
Tipo	Requisito
Fuente del requisito	Tabla QualityPage

Introducción

Además de la inserción de datos, el sistema contará con vistas para visualizar gráficos, este primer gráfico consiste en obtener la cantidad de cajas que cada trabajador del área de calidad ha revisado.

Entrada

Un valor numérico que indicará el número de semana

Proceso

Este valor se envía a la página web donde esta realizará la consulta correspondiente a la base de datos y devolverá un serie de valores que luego serán analizados para mostrar el gráfico.

Salida

Un gráfico mostrando los datos devueltos por la página web

8.3.2.6. Requisito funcional 6

RF6
Visualizar piezas revisadas por los trabajadores
Requisito
Tabla QualityPage

Introducción

Además de la inserción de datos, el sistema contará con vistas para visualizar gráficos, este primer gráfico consiste en obtener la cantidad de piezas que cada trabajador del área de calidad ha revisado.

Entrada

Un valor numérico que indicara el número de semana

Proceso

Este valor se envía a la página web donde esta realizara la consulta correspondiente a la base de datos y devolverá un serie de valores que luego serán analizados para mostrar el grafico.

Salida

Un gráfico mostrando los valores devuelto por la página web

8.3.2.7. Requisito funcional 7

Numero de requisito	RF7
Nombre del requisito	Visualizar la cantidad de un error especifico por cada semana
Tipo	Requisito
Fuente del requisito	Tabla QualityPage, CatDefect

Introducción

El sistema mostrará gráficos que indique como va aumentado o disminuyendo la cantidad de un error a lo largo de las semanas

Entrada

Un valor numérico que indica el código del error o defecto

Proceso

Este valor se envía a la página web donde esta realizará la consulta correspondiente a la base de datos y devolverá un serie de valores que luego serán analizados para mostrar el gráfico.

Salida

Un gráfico mostrando los valores devuelto por la página web

8.3.2.8. Requisito funcional 8

Numero de requisito	RF8
Nombre del requisito	Visualizar la cantidad de cajas que revisa cada trabajador por cada semana.
Tipo	Requisito
Fuente del requisito	Tabla QualityPage

Introducción

El sistema mostrará gráficos que indique como va aumentado o disminuyendo la cantidad de cajas que revisa cada trabajador a lo largo de las semanas

Entrada

Un valor de tipo cadena que indica el nombre del trabajador

Proceso

Este nombre se envía a la página web donde esta realizará la consulta correspondiente a la base de datos y devolverá un serie de valores que luego serán analizados para mostrar el gráfico.

Salida

Un gráfico mostrando los valores devuelto por la página web

8.3.2.9. Requisito funcional 9

Numero de requisito	RF9
Nombre del requisito	Visualizar la cantidad de piezas que revisa cada trabajador por cada semana.
Tipo	Requisito
Fuente del requisito	Tabla QualityPage

Introducción

El sistema mostrará gráficos que indique como va aumentado o disminuyendo la cantidad de piezas que revisa cada trabajador a lo largo de las semanas

Entrada

Un valor de tipo cadena que indica el nombre del trabajador

Proceso

Este nombre se envía a la página web donde esta realizará la consulta correspondiente a la base de datos y devolverá un serie de valores que luego serán analizados para mostrar el gráfico.

Salida

Un gráfico mostrando los valores devuelto por la página web

8.3.2.10. Requisito funcional 10

Numero de requisito	RF10
Nombre del requisito	Inicio de sesión de usuarios
Tipo	Requisito
Fuente del requisito	Tabla QualityPage

Introducción

Al iniciar la aplicación siempre pedirá al usuario que introduzca su nombre de usuario y contraseña, con el objetivo de brindar a la aplicación seguridad ya que los datos que se introducen son muy importantes.

Entrada

Nombre de usuario y contraseña, ambos de tipo cadena.

Proceso

El nombre de usuario y contraseña son enviado al sitio web y este verifica si este usuario con esa contraseña existe, si es así , el sitio web devolverá un valor booleano verdadero, si el usuario no existe o su contraseña es incorrecta el sitio web devolverá falso. En caso de que el valor devuelto sea verdadero el sistema mostrar la siguiente actividad, en caso falso le volverá a pedir el usuario y contraseña.

Salida

Un mensaje indicando al usuario si el inicio de sesión fue exitoso o no.

8.3.3. Requisitos de Rendimiento

Se necesita que el dispositivo esté conectado a una red para poder acceder al servidor web y a los datos. Además se necesita una fuente de energía para que el dispositivo pueda funcionar durante muchas horas.

9.Diagrama de Secuencias

9.1. Diagrama de Secuencia 1: Logueo de un usuario

La ilustración anterior muestra la interacción de la aplicación con el sitio web y la interacción de este con el servidor de base de datos. Al iniciar la aplicación el usuario debe elegir que acción quiere realizar (Registrar incidencia o mostrar gráficos).En ambos casos la aplicación pedirá usuario y contraseña del usuario, una vez insertado los datos el sistema ejecuta una tarea asíncrona para cual envía información al sitio web. Luego este con la información obtenida por la tarea asíncrona realiza una consulta a la base de datos para ver si existe este usuario. Dependiendo de la información que el sistema gestor de base de datos retorne al sitio web, el sitio web enviara a la tarea asíncrona un valor booleano (True , False). Con esta información obtenida la aplicación le dará o no acceso al sistema.

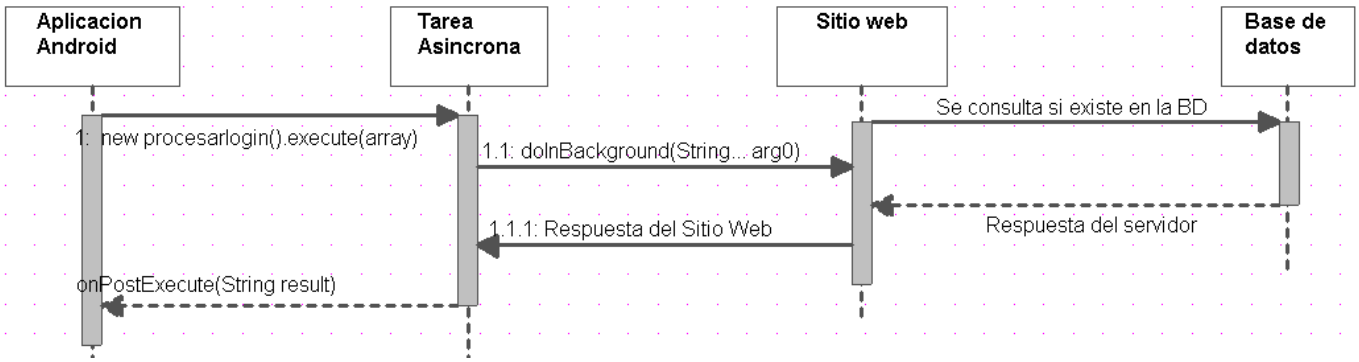


Ilustración 6 Diagrama de secuencia – Logueo de un usuario

9.2. Diagrama de secuencia 2: Registrar Auditoria

La ilustración anterior nos muestra como interactúa la aplicación con el sitio web y este con el servidor, esto se hace cuando un auditor revisa muestras de prenda la información debe guardarse aunque no se encuentre ningún defecto, la información que se almacena es la cantidad de cajas, piezas, fecha.

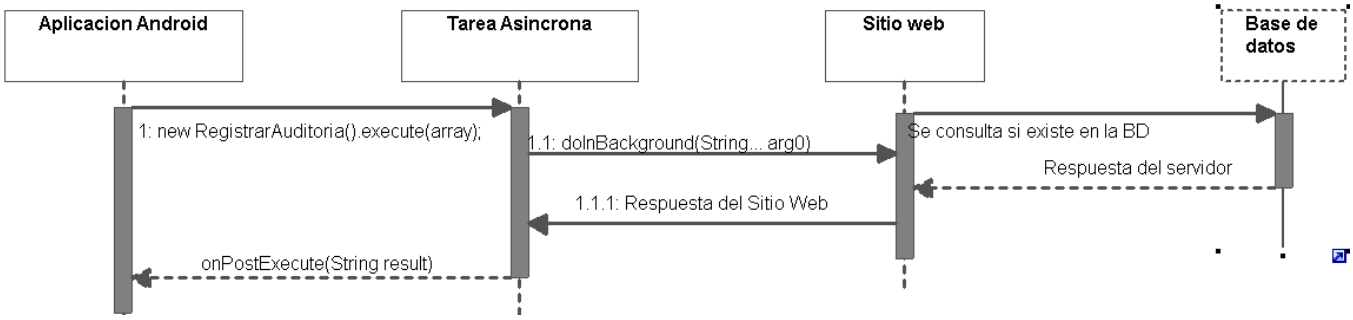


Ilustración 7 Diagrama de secuencia – Registrar Auditoria

9.3. Diagrama de secuencia 3: Registrar una Incidencia

La ilustración anterior nos muestra como interactúa la aplicación cuando se quiere registrar una incidencia, esto se hace cuando el auditor esta llevando a cabo un auditoria de las prendas. Si el auditor encuentra un defecto en una prenda, este defecto encontrado debe quedar registrado, esta información se guardara en la tabla QualityPageLoca. El auditor

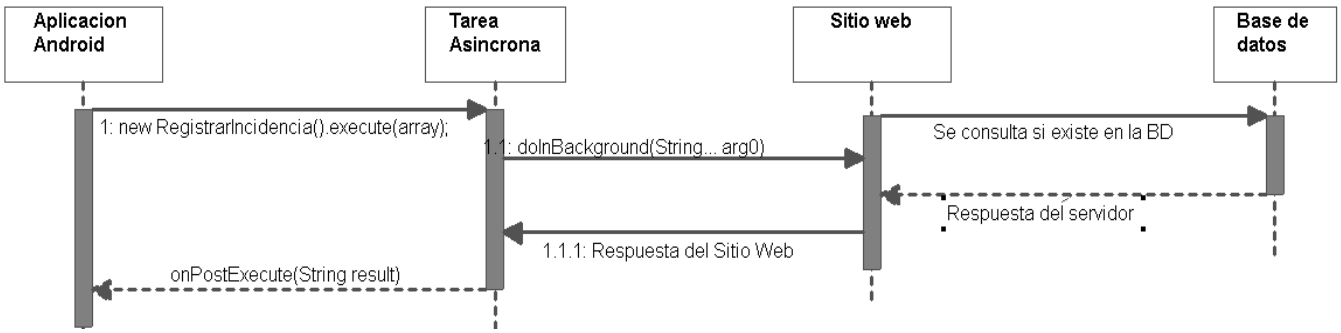
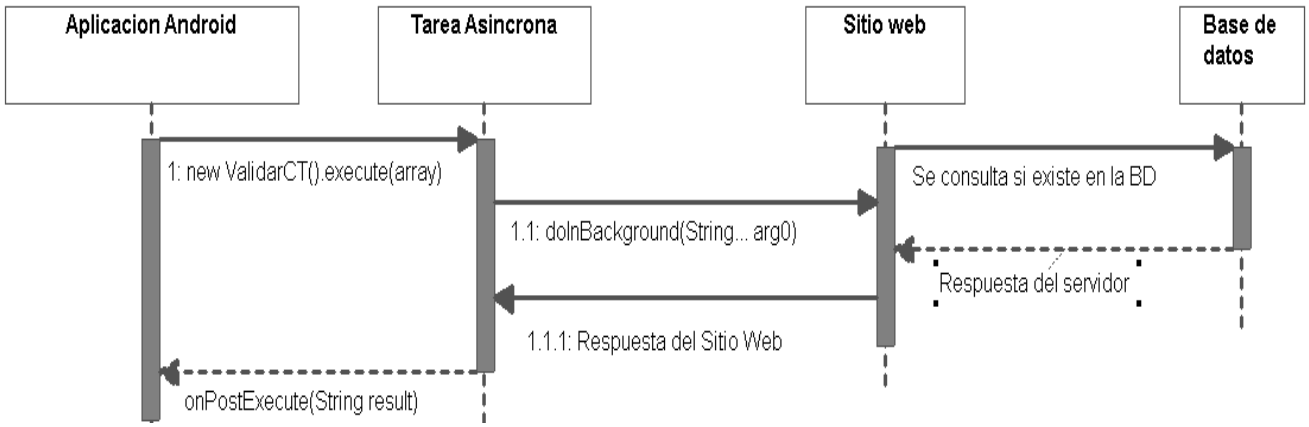


Ilustración 8 Diagrama de secuencia – Registrar Incidencia

9.4. Diagrama de secuencia 4: Validar CuttingTicket

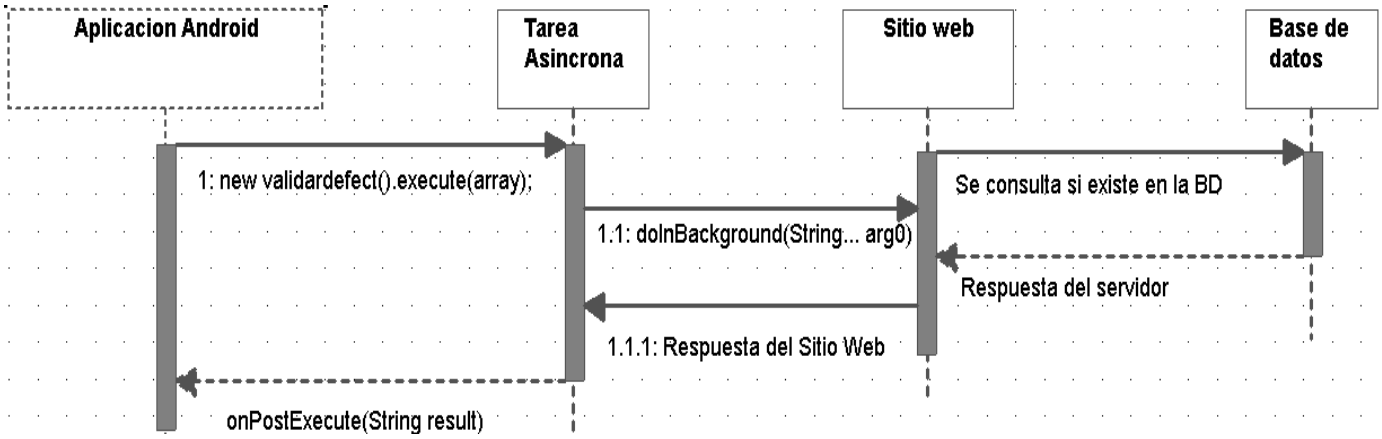
Cuando se ingresa información al sistema, este debe contar con un mecanismo para garantizar que la información a ingresar sea correcta. Cuando se ingrese una incidencia uno de los valores a introducir es el cuttingTicket, el cual debe ser validadado. El sistema lo validara cuando la caja de texto donde se ingresa el cutting ticket pierda el foco.



Ilustracion 9 Diagrama de secuencia para Validar Cutting Ticket

9.5. Diagrama de secuencia 5: Validar código de defecto

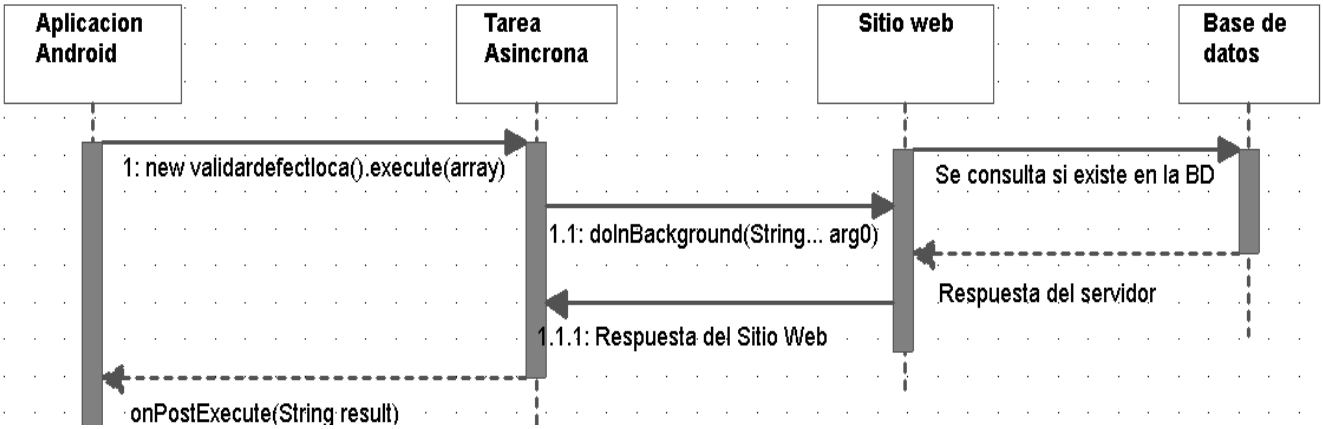
Cuando se registra una incidencia se debe guardar el tipo de defecto encontrado. Este valor es un código que se encuentra en la tabla CatDefect así que cuando se introduce un valor en la caja y esta pierde el foco se validara si existe el código introducido.



Ilustracion 10 Diagrama de secuencia para validar código de defecto

9.6. Diagrama de secuencia 6: Validar código localización del defecto

Cuando se registra una incidencia se debe guardar la localización donde se encuentra el defecto. Este valor es un código que se encuentra en la tabla CatDefectLoca así que cuando se introduce un valor en la caja y esta pierde el foco se validara si existe el código introducido.



Ilustracion 11 Diagrama de secuencia para validar localización del defecto

10. Conclusiónes

Una vez analizada la información, dise;ada y codificada la aplicacion en el presente trabajo se ha llegado a las siguientes conclusiones:

- La arquitectura de la aplicación permite integrarse con las tecnología usada en la empresa (servidor web IIS, Sistema Gestor de Base de dato SQL Server) permitiendo envió y recepción de los datos.
- El diseño de la aplicación permite llevar a cabo actividades del área de calidad como registrar auditoria, incidencia.
- La combinación de mecanismos para generación de gráficos e interfaces, Permite visualizar la información de manera interactiva para su correcto análisis.

11. Anexos

11.1. Segmento de código o algoritmo de mayor relevancia

11.1.1 Clase Asíncrona para guardar Auditoria

```
private class RegistrarAuditoria extends AsyncTask<String, Void, String>
        {

                @Override
                protected String doInBackground(String... arg0) {
                        // TODO Auto-generated method stub
                HttpClient httpClient = new DefaultHttpClient();
                HttpPost post = new HttpPost("http://"+
arg0[0].toString()+"/Maestria/Default.aspx/guardar_det");
                //Construimos el objeto cliente en formato JSON

                JSONObject dato = new JSONObject();
                try {
                        dato.put("IdQualityCtrlPage", arg0[1].toString());
                        dato.put("IdCt", arg0[2].toString());
                        dato.put("Initials", arg0[3].toString());
                        dato.put("QtyCajas", arg0[4].toString());
                        dato.put("QtyPiezas", arg0[5].toString());
                        dato.put("QtyDefectos", arg0[6].toString());
                        dato.put("StatusD", arg0[7].toString());
                        dato.put("CantidR", arg0[8].toString());


                        //Toast.makeText(getBaseContext(), "Mensaje 2",
Toast.LENGTH_SHORT).show();
                        } catch (JSONException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }

                ////////////////////
                //https://danlaho.wordpress.com/2011/11/12/android-y-json/
                ////////////////////

                StringEntity entity;
                String respStr = null;

                        try {
```

```
                                        //entity=new StringEntity(jar.toString());
                                        entity = new StringEntity(dato.toString());
                                        post.setEntity(entity);
                                        post.setHeader("Accept", "application/json");
                                        post.setHeader("content-type", "application/json");
                                        HttpResponse resp = httpClient.execute(post);
                                        respStr = EntityUtils.toString(resp.getEntity());

                                } catch (UnsupportedEncodingException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                } catch (ClientProtocolException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                } catch (IOException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
                        //Toast.makeText(getBaseContext(), dato.toString(),
Toast.LENGTH_SHORT).show();
                                //Toast.makeText(getBaseContext(), respStr.toString(),
Toast.LENGTH_SHORT).show();
                                return respStr;
                        }



                        protected void onPostExecute(String result)
                        {
                                Toast.makeText(getBaseContext(), result.toString(),
Toast.LENGTH_SHORT).show();
                        }
                }
```

11.1.2 Clase Asíncrona para guardar incidencia

```
private class RegistrarIncidencia extends AsyncTask<String, Void, String>
        {

                @Override
                protected String doInBackground(String... arg0) {
                        // TODO Auto-generated method stub
                HttpClient httpClient = new DefaultHttpClient();
                HttpPost post = new
HttpPost("http://"+arg0[0].toString()+"/Maestria/Default.aspx/guardar_pageloca");

                //Construimos el objeto cliente en formato JSON

                JSONObject dato = new JSONObject();
                try {
                        dato.put("IdQualityCtrlPage", arg0[1].toString());
```

```
                        dato.put("IdStyle", arg0[2].toString());
                        dato.put("IdDefect", arg0[3].toString());
                        dato.put("IdDefectLoca", arg0[4].toString());



                        //Toast.makeText(getBaseContext(), "Mensaje 2",
Toast.LENGTH_SHORT).show();
                        } catch (JSONException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }

                ////////////////////
                //https://danlaho.wordpress.com/2011/11/12/android-y-json/
                ////////////////////

                StringEntity entity;
                String respStr = null;

                        try {
                                //entity=new StringEntity(jar.toString());
                                entity = new StringEntity(dato.toString());
                                post.setEntity(entity);
                                post.setHeader("Accept", "application/json");
                                post.setHeader("content-type", "application/json");
                                HttpResponse resp = httpClient.execute(post);
                                respStr = EntityUtils.toString(resp.getEntity());

                        } catch (UnsupportedEncodingException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        } catch (ClientProtocolException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        } catch (IOException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                        return respStr;
                }



                protected void onPostExecute(String result)
                {
                        Toast.makeText(getBaseContext(), result.toString(),
Toast.LENGTH_SHORT).show();
                }
        }
```

11.1.3 Clase Asíncrona para validar Style

```
public class validarStyle extends AsyncTask<String, Void, String>
{

	@Override
	protected String doInBackground(String... arg0) {
		// TODO Auto-generated method stub
	HttpClient httpClient = new DefaultHttpClient();
	HttpPost post = new
HttpPost("http://"+arg0[0].toString()+"/Maestria/Default.aspx/validarStyle");
	httpClient.getParams().setIntParameter("http.connection.timeout",3000);

	//Construimos el objeto cliente en formato JSON

	JSONObject dato = new JSONObject();
	try {
		dato.put("valor", arg0[1].toString());


		//Toast.makeText(getBaseContext(), "Mensaje 2", Toast.LENGTH_SHORT).show();
		} catch (JSONException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}



	StringEntity entity;
	String respStr = null;
	JSONObject js;

		try {
				entity = new StringEntity(dato.toString());
			post.setEntity(entity);
			post.setHeader("content-type", "application/json");
			HttpResponse resp = httpClient.execute(post);
			respStr = EntityUtils.toString(resp.getEntity());


		} catch (UnsupportedEncodingException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (ClientProtocolException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

		return respStr;
	}


	protected void onPostExecute(String result)
	{
		if(result==null)
```

```
                {
                        return;
                }
                else
                {
                        try {
                                JSONObject jsonObj = new JSONObject(result);
                                String va=jsonObj.get("d").toString();
                                Toast.makeText(getBaseContext(), va,
Toast.LENGTH_SHORT).show();


                                if(va.equalsIgnoreCase("\"False\""))
                                {

        ctestilo.setBackgroundResource(R.drawable.backwithborder);
                                        validar_cajas_style=false;

                                }
                                else
                                {
                                        validar_cajas_style=true;

        ctestilo.setBackgroundResource(R.drawable.lastborder);
                                }



                                } catch (JSONException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }

                }



        }
}
```

11.1.4 Código para Editar incidencia

```
public void  Editar_Defecto(final int pos)
        {
                final Dialog dlg = new Dialog(this);
                dlg.setContentView(R.layout.activity_agregar_defecto);
                dlg.setTitle("Agregar un defecto");
                dlg.setCancelable(false);

                Button btAceptar = (Button)dlg.findViewById(R.id.btAddDefectoOK);
                Button btCancel = (Button)dlg.findViewById(R.id.btAddDefectoCancel);
```

```
                final QualityPageLoca defect=new QualityPageLoca();

                //Captuda de valore
                final EditText ctcantidad=(EditText)dlg.findViewById(R.id.etCantidadDet);
                final EditText ctestilo=(EditText)dlg.findViewById(R.id.etEstiloDet);
                final EditText ctdefecto=(EditText)dlg.findViewById(R.id.etDefectoDet);
                final EditText ctubicacion=(EditText)dlg.findViewById(R.id.etUbicacionDet);

                ctcantidad.setText(String.valueOf( Array_pageloca.get(pos).getIdQualityCtrlPage()));
                ctestilo.setText(String.valueOf( Array_pageloca.get(pos).getIdStyle()));
                ctdefecto.setText(String.valueOf(Array_pageloca.get(pos).getIdDefect()));
                ctubicacion.setText(String.valueOf(Array_pageloca.get(pos).getIdDefectLoca()));


                Log.e("Error", "Entrando a interfaz");
                btAceptar.setOnClickListener(new OnClickListener(){
                        @Override
                        public void onClick(View arg0) {

                                // TODO Auto-generated method stub
                                Log.e("Error", "Entrando al click ok");
                                defect.setIdDefect(Integer.parseInt(  ctdefecto.getText().toString()));

        defect.setIdDefectLoca(Integer.parseInt(ctubicacion.getText().toString()));
                                defect.setIdQualityCtrlPage(1);
                                defect.setIdStyle(Integer.parseInt(ctestilo.getText().toString()));
                                //fdetalle.aq.add(defect);
                                //Array_pageloca.add(defect);
                                Array_pageloca.set(pos, defect);
                                fdetalle.aq.set(pos+1, defect);
                                fdetalle.cargar();

                                Toast.makeText(getBaseContext(), "Defecto Actualizado",
Toast.LENGTH_SHORT).show();
                                Log.e("Error", "Entrando al click ok1111");
                                //fdetalle.cargar();
                                //inter_defe.InsertarDefecto(defect);

                                dlg.cancel();

                                MainActivity.this.cargarDatosDetalle();
                        }});

                btCancel.setOnClickListener(new OnClickListener (){
                        @Override
                        public void onClick(View arg0) {
                                // TODO Auto-generated method stub
                                dlg.cancel();
                        }});
                dlg.show();




        }
```

11.1.5 Código de clase asíncrona para obtener las piezas revisadas por trabajador

```
private class ObtenerPiezasRevisadas extends AsyncTask<String, Void, String>
	{

		@Override
		protected String doInBackground(String... arg0) {
			// TODO Auto-generated method stub
		HttpClient httpClient = new DefaultHttpClient();
		HttpPost post = new
HttpPost("http://"+arg0[0].toString()+"/Maestria/Default.aspx/cantidad_piezas_revizadas");

		//Construimos el objeto cliente en formato JSON

		JSONObject dato = new JSONObject();
		try {
			dato.put("valor", arg0[1].toString());


			//Toast.makeText(getBaseContext(), "Mensaje 2",
Toast.LENGTH_SHORT).show();
			} catch (JSONException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}


		StringEntity entity;
		String respStr = null;
		JSONObject js;

			try {
				//entity=new StringEntity(jar.toString());
				entity = new StringEntity(dato.toString());
				post.setEntity(entity);
				//post.setHeader("Accept", "application/json");
				post.setHeader("content-type", "application/json");
				HttpResponse resp = httpClient.execute(post);
				respStr = EntityUtils.toString(resp.getEntity());
				//HttpEntity httpentity=resp.getEntity();


			} catch (UnsupportedEncodingException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			} catch (ClientProtocolException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			} catch (IOException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		//Toast.makeText(getBaseContext(), dato.toString(),
```

```
Toast.LENGTH_SHORT).show();
                        //Toast.makeText(getBaseContext(), respStr.toString(),
Toast.LENGTH_SHORT).show();
                        return respStr;
                }


                protected void onPostExecute(String result)
                {


                        Gson gs=new Gson();
                        try {
                        JSONObject jsonObj = new JSONObject(result);
                                JSONArray ar=new JSONArray(jsonObj.getString("d"));
                                List<String> nombre=new ArrayList<String>();
                                List<String> ncajas=new ArrayList<String>();
                                for(int i=0;i<ar.length();i++)
                                {
                                        JSONObject oneObject = ar.getJSONObject(i);
                                //      Toast.makeText(getBaseContext(),
oneObject.getString("nombre"), Toast.LENGTH_SHORT).show();
                                        nombre.add(oneObject.getString("nombre"));
                                        if(boolCajas==true){
                                        ncajas.add(oneObject.getString("ncajas"));}
                                        else {
                                                ncajas.add(oneObject.getString("npiezas"));}

                                }
                                generar_reporte_cajasRevisadas(nombre, ncajas);

                        } catch (JSONException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }

                }
        }
```

11.1.6 Código para generar gráfico de piezas revisadas por trabajador

```
void generar_reporte_cajasRevisadas(List<String> nombre,List<String> ncajas)
        {
                graph=(GraphView) findViewById(R.id.idgraw);
                //Toast.makeText(getBaseContext(), nombre.get(1),
Toast.LENGTH_SHORT).show();
                graph.setTitle("Cantidad de cajas revisadas por empleado");
                DataPoint[] dp=new DataPoint[nombre.size()];
                for(int i=0;i<nombre.size();i++)
                {
                        dp[i]=new DataPoint(i,Integer.parseInt(ncajas.get(i)));
                }

                BarGraphSeries<DataPoint> series=new BarGraphSeries<DataPoint>(dp);
                series.setSpacing(200);
                series.setDrawValuesOnTop(true);

                StaticLabelsFormatter staticLabelsFormatter = new StaticLabelsFormatter(graph);

                //Agregamos los nombre para ponerlos como etiquetas
                String[] nom=new String[nombre.size()];
                for(int i=0;i<nombre.size();i++)
                {
                        //nom[i] =nombre.get(i)+" "+ncajas.get(i);
                        nom[i] =nombre.get(i);
                }

                //staticLabelsFormatter.setVerticalLabels(nom);

                staticLabelsFormatter.setHorizontalLabels(nom);
                graph.getGridLabelRenderer().setLabelFormatter(staticLabelsFormatter);
                graph.removeAllSeries();

                graph.addSeries(series);

                series.setSpacing(30);
                series.setValueDependentColor(new ValueDependentColor<DataPoint>() {
                   @Override
                   public int get(DataPoint data) {
                   return Color.rgb((int) data.getX()*255/4, (int) Math.abs(data.getY()*255/6), 100);
                   }
                });


                //series.setDrawValuesOnTop(true);
                series.setValuesOnTopColor(Color.BLACK);



        }
```

11.2. Métodos web públicos en página web

11.2.1. Cantidad de piezas revisadas

```
    [WebMethod]
    public static string cantidad_piezas_revizadas(string valor)
    {
        QualitiPageDet qp = new QualitiPageDet();
        DataTable dat = new DataTable();
        string json = string.Empty;


        JavaScriptSerializer serializer = new JavaScriptSerializer();


        dat = qp.cantidad_piez_revisada(valor);
        Object[] myArray = new Object[dat.Rows.Count];
        for (int i = 0; i < dat.Rows.Count; i++)
        {
            myArray[i] = new { nombre = dat.Rows[i][0].ToString(), ncajas =
dat.Rows[i][1].ToString(), npiezas = dat.Rows[i][2].ToString() };
        }
        json += JsonConvert.SerializeObject(myArray, Formatting.Indented);
        return json;

    }
```

11.2.2. Cantidad de piezas revisadas por semana y por trabajador

```
    [WebMethod]
    public static string Obtener_cajas_x_cada_trabajador(string valor)
    {
        QualitiPageDet qp = new QualitiPageDet();
        DataTable dat = new DataTable();
        string json = string.Empty;


        JavaScriptSerializer serializer = new JavaScriptSerializer();
```

```
        dat = qp.Obtener_cajas_x_cada_trabajador(valor);
        Object[] myArray = new Object[dat.Rows.Count];
        for (int i = 0; i < dat.Rows.Count; i++)
        {
            myArray[i] = new { semana = dat.Rows[i][0].ToString(), ncaja =
dat.Rows[i][1].ToString(), npieza = dat.Rows[i][2].ToString() };
        }
        json += JsonConvert.SerializeObject(myArray, Formatting.Indented);
        return json;


    }
```

11.2.3. Obtener nombre de todos los auditores

```
    [WebMethod]
    public static string get_NameAuditor(string valor)
    {
        QualitiPageDet qp = new QualitiPageDet();
        DataTable dat = new DataTable();
        string json = string.Empty;


        JavaScriptSerializer serializer = new JavaScriptSerializer();


        dat = qp.Obtener_nombre_trabajadores();
        Object[] myArray = new Object[dat.Rows.Count];
        for (int i = 0; i < dat.Rows.Count; i++)
        {
            myArray[i] = new { nombre = dat.Rows[i][0].ToString() };
        }
        json += JsonConvert.SerializeObject(myArray, Formatting.Indented);
        return json;


    }
```

11.2.4. Cantidad de errores por semana

```
[WebMethod]
    public static string get_cantidad_Errores_x_semana(string valor)
    {
        QualitiPageDet qp = new QualitiPageDet();
        DataTable dat = new DataTable();
        string json = string.Empty;
        JavaScriptSerializer serializer = new JavaScriptSerializer();
```

```
        dat = qp.cantidad_errores_x_Semana(valor);
        Object[] myArray = new Object[dat.Rows.Count];
        for (int i = 0; i < dat.Rows.Count; i++)
        {
            myArray[i] = new { semana = dat.Rows[i][1].ToString(), cantidad =
dat.Rows[i][2].ToString() };
        }
        json += JsonConvert.SerializeObject(myArray, Formatting.Indented);
        return json;


    }
```

11.2.5. Validar CuttingTicket

```
[WebMethod]
    public static string validadCT(string valor)
    {
        QualitiPageDet qp = new QualitiPageDet();
        //string v = "valor:" + qp.validarCT(valor).ToString();
      //  return JsonConvert.SerializeObject(v, Formatting.Indented);
        return JsonConvert.SerializeObject(qp.validarCT(valor).ToString(),
Formatting.Indented);

    }
```

11.2.6. Validar Style

```
[WebMethod]
    public static string validarStyle(string valor)
    {
        QualitiPageDet qp = new QualitiPageDet();
        //string v = "valor:" + qp.validarCT(valor).ToString();
        //  return JsonConvert.SerializeObject(v, Formatting.Indented);
        return JsonConvert.SerializeObject(qp.validarStyle(valor).ToString(),
Formatting.Indented);

    }
```

11.2.7. Validar Defect

```
[WebMethod]
    public static string validarDefect(string valor)
    {
        QualitiPageDet qp = new QualitiPageDet();
        //string v = "valor:" + qp.validarCT(valor).ToString();
```

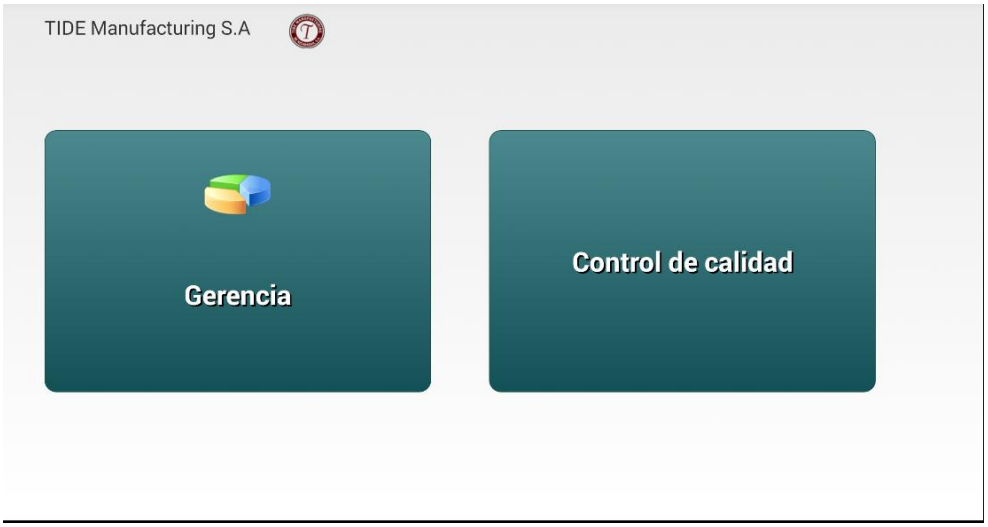
```
        //  return JsonConvert.SerializeObject(v, Formatting.Indented);
        return JsonConvert.SerializeObject(qp.validarDefect(valor).ToString(),
Formatting.Indented);

    }
```

11.2.8. Validar DefectLoca

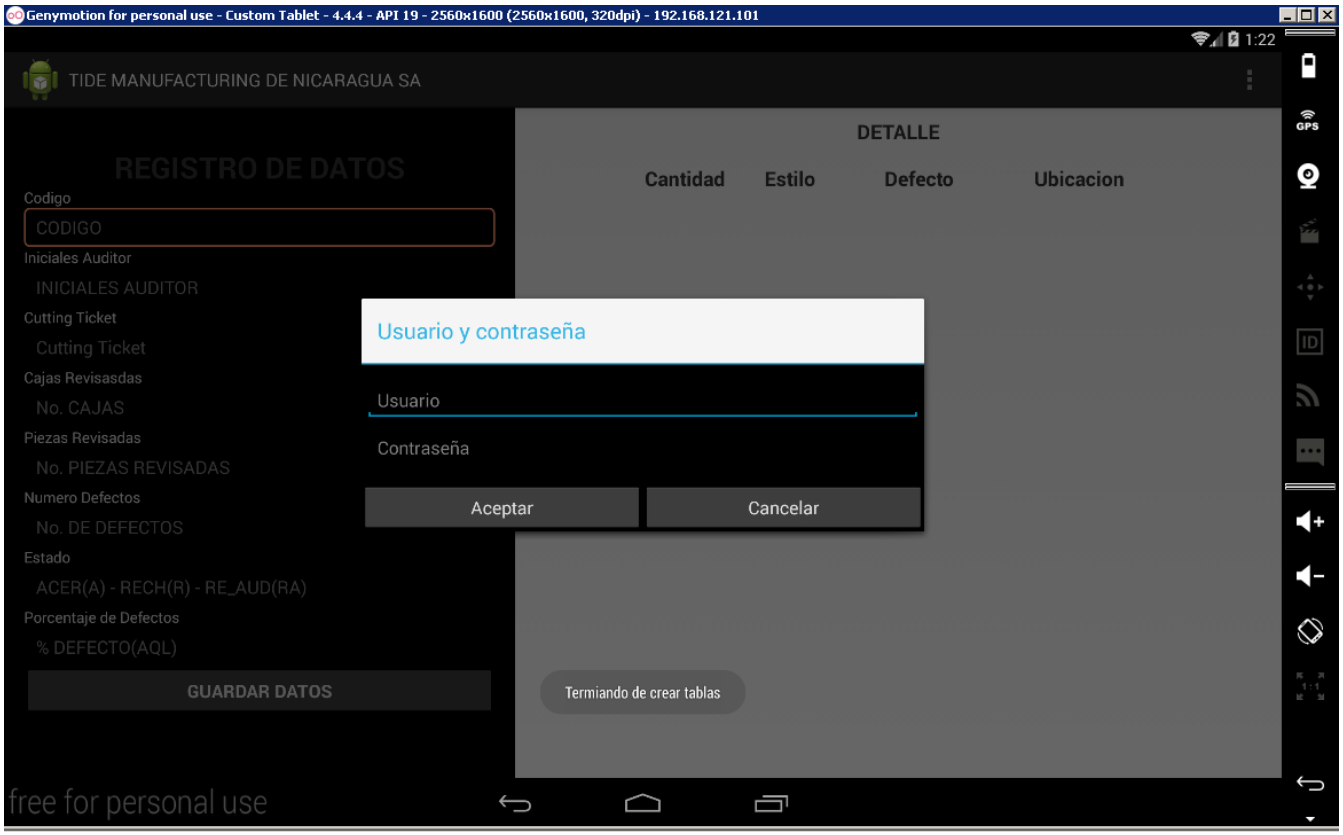
```
[WebMethod]
    public static string validarDefectLoca(string valor)
    {
        QualitiPageDet qp = new QualitiPageDet();
        //string v = "valor:" + qp.validarCT(valor).ToString();
        //  return JsonConvert.SerializeObject(v, Formatting.Indented);
        return JsonConvert.SerializeObject(qp.validarDefectloca(valor).ToString(),
Formatting.Indented);

    }
```

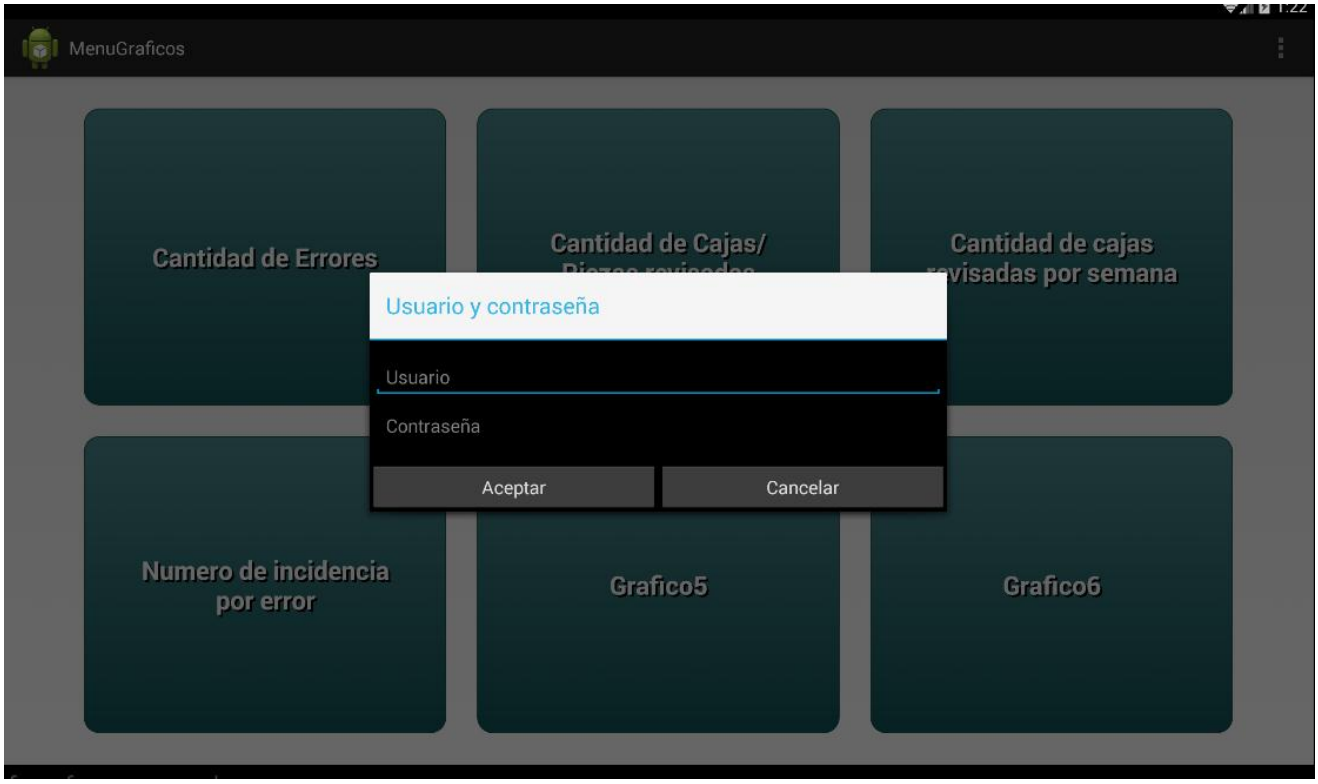
11.3. Interfaces de la aplicación

11.3.1. Actividad Principal

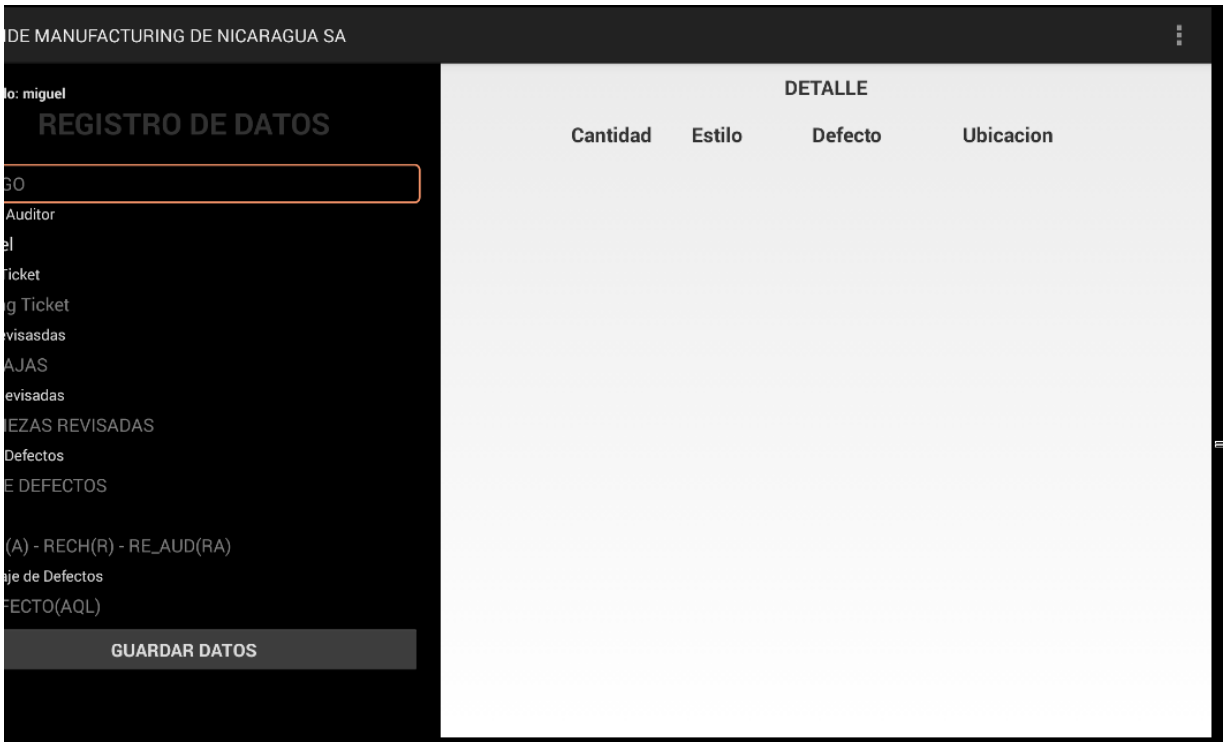


11.3.2. Logueo

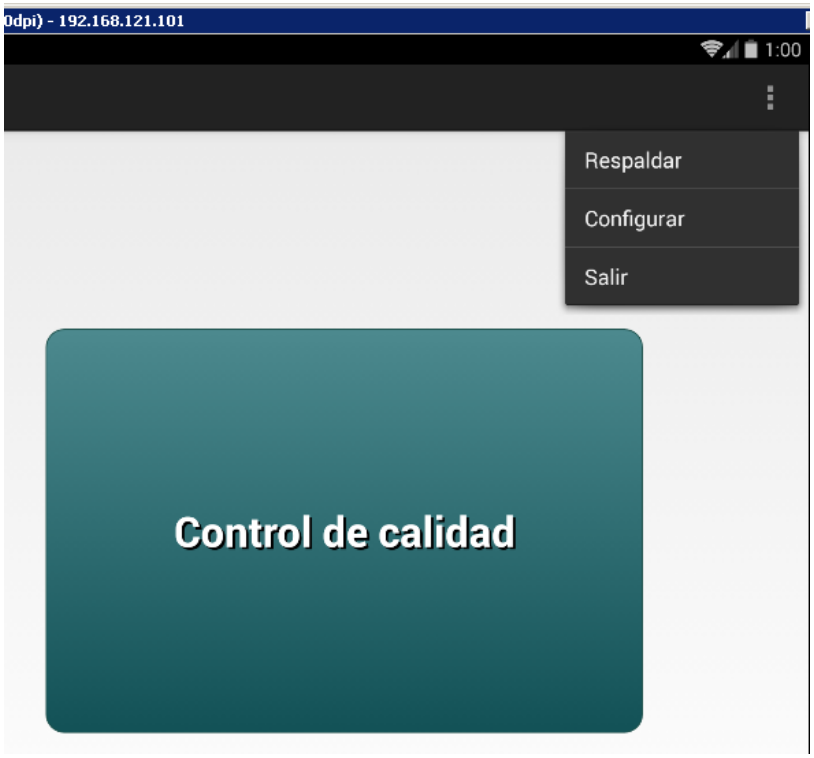

Genymotion for personal use - Custom Tablet - 4.4.4 - API 19 - 2560x1600 (2560x1600, 320dpi) - 192.168.121.101
1:22
TIDE MANUFACTURING DE NICARAGUA SA
REGISTRO DE DATOS
Codigo
CODIGO
Iniciales Auditor
INICIALES AUDITOR
Cutting Ticket
Cutting Ticket
Cajas Revisasdas
No. CAJAS
Piezas Revisadas
No. PIEZAS REVISADAS
Numero Defectos
No. DE DEFECTOS
Estado
ACER(A) - RECH(R) - RE_AUD(RA)
Porcentaje de Defectos
% DEFECTO(AQL)
GUARDAR DATOS
DETALLE
Cantidad
Estilo
Defecto
Ubicacion
Usuario y contraseña
Usuario
Contraseña
Aceptar
Cancelar
Termiando de crear tablas
free for personal use

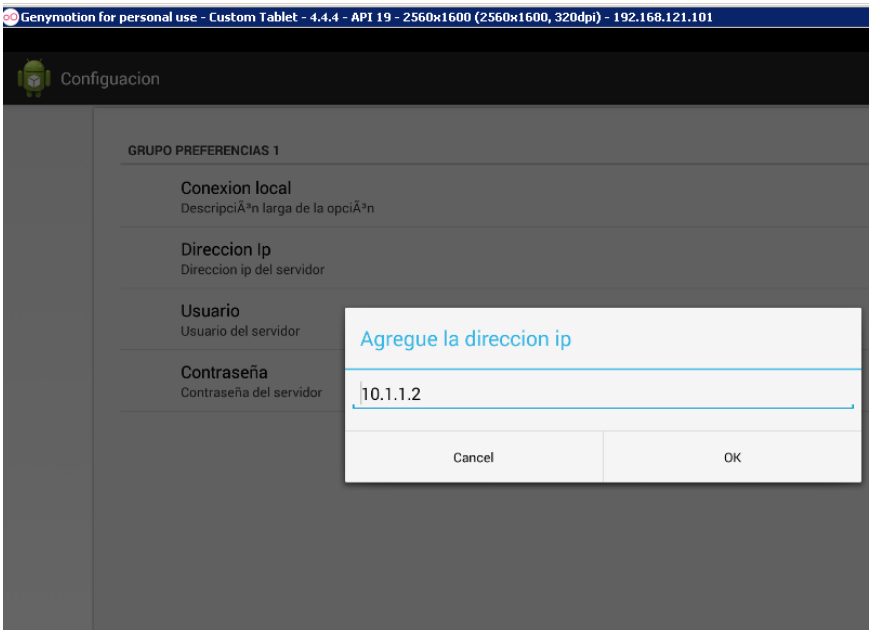

MenuGraficos
Cantidad de Errores
Cantidad de Cajas/
Cantidad de cajas
revisadas por semana
Usuario y contraseña
Usuario
Contraseña
Aceptar
Cancelar
Numero de incidencia
por error
Grafico5
Grafico6

11.3.3. Registro de auditoria e incidencias



11.3.4. Configuración inicial de la aplicación

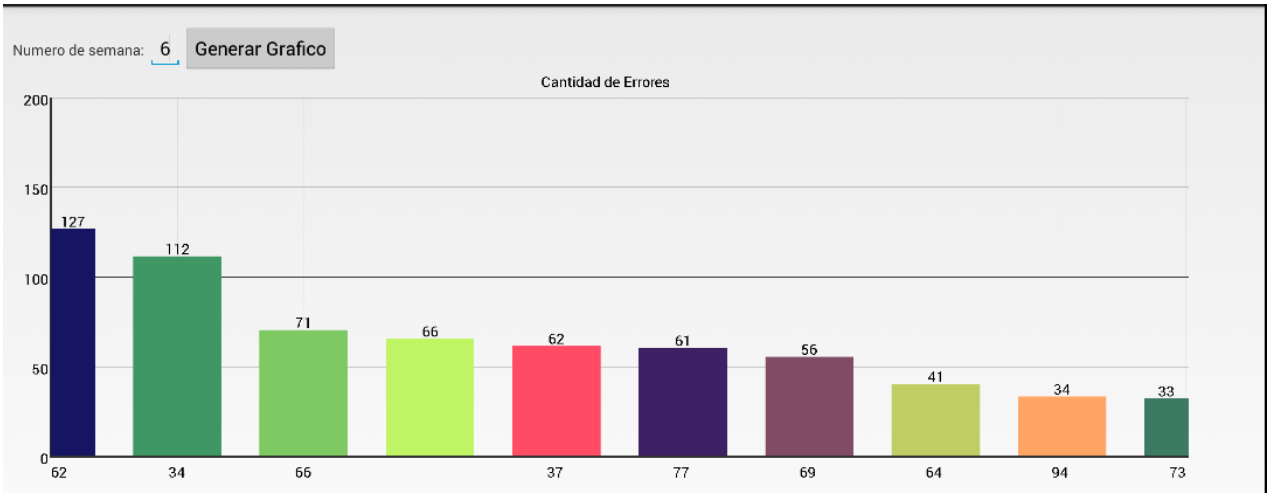




11.3.5. Ventana de gráficos

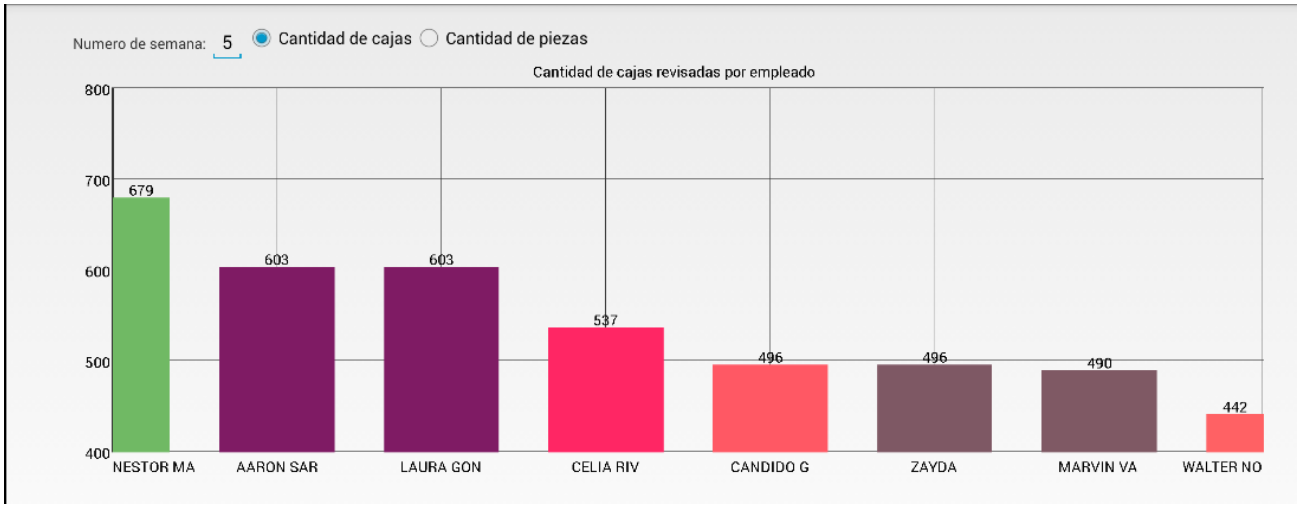
11.3.6. Grafico 1: Cantidad de errores por semana

Este grafico muestra la cantidad de cada error en una semana determinada, muestra los 10 errores mas frecuentes.



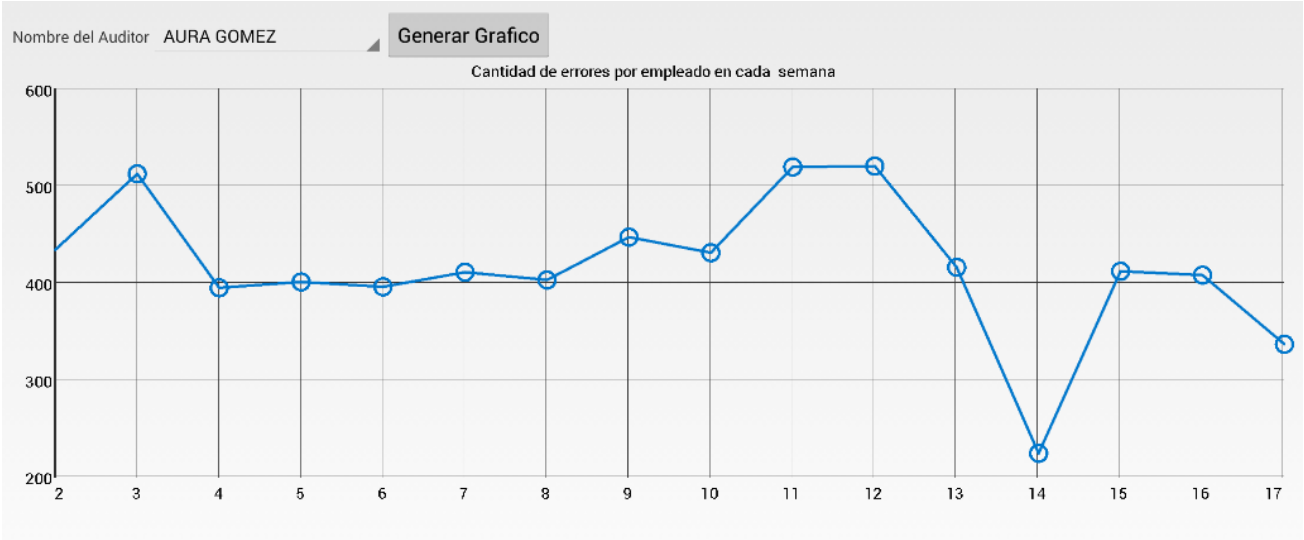
11.3.7. Grafico 2: Cantidad de piezas revisada por empleado en una semana determinada

Este grafico muestra la cantidad de piezas y cajas que cada empleado ha revisado, muestra los 10 empleados que revisaron más cajas y piezas



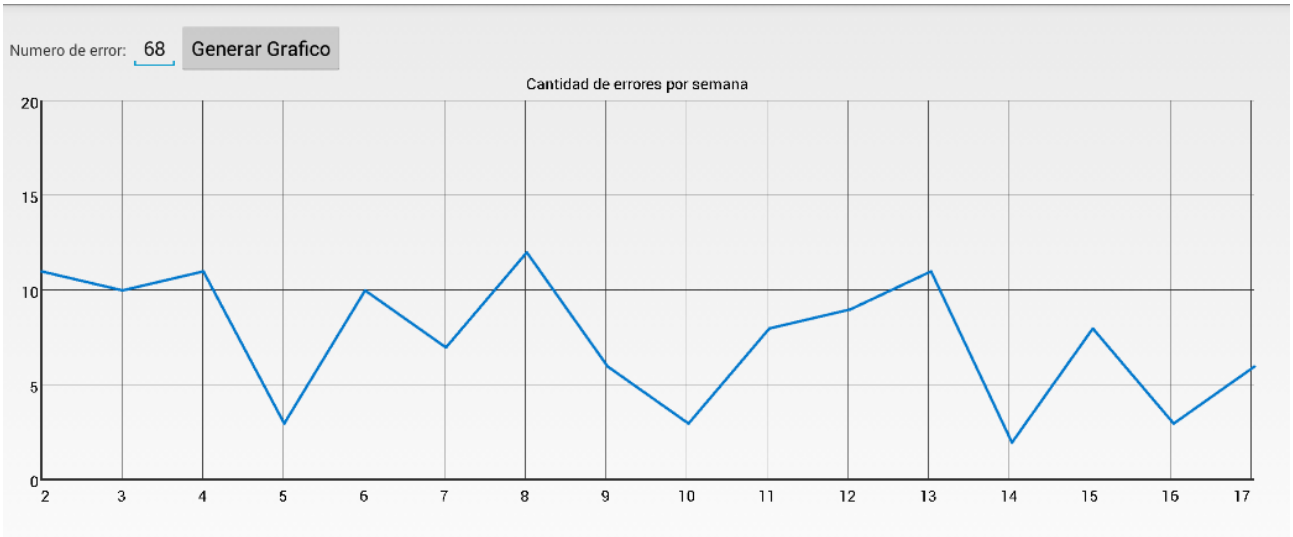
11.3.8. Grafico 3: Cantidad de piezas revisadas por empleado

Este grafico muestra información más detallada que el anterior, se debe seleccionar al empleado y al darle Click en botón Generar Grafico mostrara cuantos errores ha encontrado a lo largo de las semanas.



11.3.9. Grafico 4: Cantidad de un error por cada semana

Este grafico muestra la cantidad de ocurrencia de un error por cada semana. Se debe introducir el código del error y al dar click en el botton Generar Grafico mostrara la cantidad del error por cada semana.



11.4. Encuesta a realizar a personal de control de calidad sobre las pruebas de la aplicación.

Nombre del Auditor: _____

Fecha: _____

Tipo de error :

- Conexion
- Aplicacion

Descripción del error:

_____

_____

Sugerencia de mejoras en la interfaz:

_____

_____

Sugerencia de mejora en funcionalidad:

_____

_____

Referencias Bibliográficas

[1] D. Baeza , A. F. Aguirre , N. C. Andrés y C. Ocampo, «Software para toma y procesamiento de datos de campo en programas de control de dengue,» 2012.

[2] J. T. Girones, El gran libro de Android, Barcelona, España : Alfaomega Grupo Editor, S.A., 2012.

[3] Á. B. J. C. C. M. G. F. H. D. P. J. R. d. L. D. S. S. P. T. Z. Manuel Báez, Introduccion a Android, E.M.E. Editorial.

[4] F. Eclipse, «https://www.eclipse.org/,» [En línea].

[5] M. S. Iacono, ASP.net Ejercicios para compreder la tecnologia.

[6] R. S. Pressman, Ingenieria del sotfware Un enfoque practico.

[7] N. D. F. Miguel de Icaza, «www.xamarin.com/,» Attachmate, 2011. [En línea].

[8] Stephanie Falla Aroche, . I. E. Mendoza y A. Catalán, «Curso Android: Desarrollo de aplicaciones móviles,» Junio 2011.

[9] I. F. Darwin, Android Cookbook, Sebastopol: O'Reilly Media, Inc, 2011.

[10] R. S. Presuman, «Ingeniería del Software: Un enfoque practico,» pp. Pag. 26-30.

[11] K. M. Polanco y J. L. Beauperthuy Taibo, «"ANDROID" GOOGLE'S OPERATING,» *Scientific e-journal of Management,* 2011.

[12] L. N. d. C. d. S. d. INTECO, «INGENIERÍA DEL SOFTWARE: METODOLOGÍAS Y CICLOS DE VIDA,» 2009.

[13] I. C. Society, «Espeficicacion de Requisitos segun el estandar de IEEE 830,»

1993.

[14] «http://developer.android.com,» [En línea].

[15] A. Inc, «https://www.apple.com/,» [En línea].

[16] M. Corporation, «http://www.windowsphone.com,» [En línea].

[17] X. Inc., «http://xamarin.com/,» [En línea].