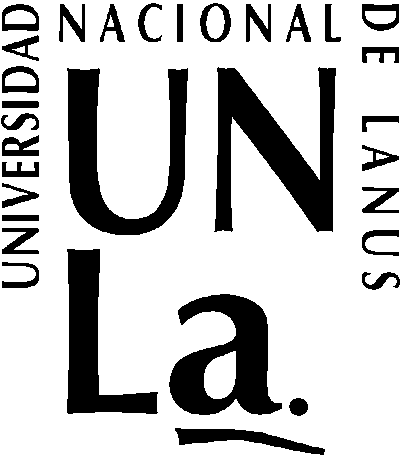
****

**Registración Espacial de Objetos en el Terreno (REOT)**

**Buczacka, Walter**

**González, Gustavo**

**Pérez, Nicolás**

**Cátedra Proyecto de Software**

**Licenciatura en Sistemas**

**2016**

*Registración Espacial de Objetos*

*En El Terreno* *(REOT)*

Pérez, Nicolás –Buczacka, Walter – González, Gustavo

Licenciatura en Sistemas – Departamento de Desarrollo Productivo y Tecnológico Universidad Nacional de Lanús

nperez\_dcao\_smn@outlook.com – gonzalezgustavojavier@gmail.com - buczacka.walter@gmail.com

*Extracto*—En este documento constarán las actividades realizadas por el equipo de desarrollo para la entrega pautada de un primer hito. El mismo constara del análisis preliminar*(Abstracto)*

# **Introducción**

La incorporación de éste proyecto surge a partir de un pedido de la Licenciatura en Gestión Ambiental Urbana. Para incorporar dos mecanismos de relevamiento de árboles en una determinada ciudad. En primera instancia el software será testeado dentro del establecimiento de la UNLa, dicho software constará de dos herramientas, la primera que consiste en la geolocalización de fotografías, y la segunda de la descripción de los árboles fotografiados. A priori no son necesarios el uso simultáneo de dichos segmentos del programa, pero al finalizar el desarrollo ambos funcionarán en simultáneo para que el cliente use los datos que considere pertinente.

La aplicación estará diseñada en 3 capas, en la primer etapa tendremos una aplicación móvil, que sacará una captura de un árbol, dicha captura estará ligada a un par ordenado(Coordenadas), y una descripción, el dispositivo móvil (si el usuario lo desea), enviará toda esta información al servidor alojado en la Universidad; dicho servidor en la segunda capa, direccionará la información a la cpu individual del cliente y guardará toda ésta información en una base de datos; ya en la 3er capa, la cpu del cliente recibirá los datos, con los que podrá generar un excel el cual será exportado a un mapa compuesto SIG, donde cada capa del compuesto reflejan datos recibidos.

Este sistema propone la solución a todo lo mencionado de forma automática.

1. **PROCESO DE SELECCIÓN DE UN MCVS**
   1. **Identificar los posibles MCVS:**

Dedicaremos este primer segmento a la comprensión y entendimiento de por que es necesario construir un ciclo de vida pertinente desde el puntapié inicial de cualquier proyecto de software.

Todo proyecto de ingeniería tiene unos fines ligados a la obtención de un producto, proceso o servicio que es necesario generar a través de diversas actividades. Algunas de estas actividades pueden agruparse en fases porque globalmente contribuyen a obtener un producto intermedio, necesario para continuar hacia el producto final y facilitar la gestión del proyecto. Al **conjunto de las fases empleadas se le denomina *“ciclo de vida”.***

Sin embargo, la forma de agrupar las actividades, los objetivos de cada fase, los tipos de productos intermedios que se generan, etc. pueden ser muy diferentes dependiendo del tipo de producto o proceso a generar y de las tecnologías empleadas.

Es fundamental conocer las principales diferencias de cada ciclo de vida para poder hacer la selección que mas se adecue a nuestro diseño.

Las principales diferencias entre distintos modelos de ciclo de vida están en:

* El **alcance** del ciclo dependiendo de hasta dónde llegue el proyecto correspondiente. Un proyecto puede comprender un simple estudio de viabilidad del desarrollo de un producto, o su desarrollo completo o, llevando la cosa al extremo, toda la historia del producto con su desarrollo, fabricación, y modificaciones posteriores hasta su retirada del mercado.
* Las **características** (contenidos) de las fases en que dividen el ciclo. Esto puede depender del propiotema al que se refiere el proyecto (no son lo mismo las tareas que deben realizarse para proyectar un avión que un puente), o de la organización (interés de reflejar en la división en fases aspectos de la división interna o externa del trabajo).
* La **estructura** de la sucesión de las fases que puede ser lineal, con prototipado, o en espiral. Veámoslo con más detalle:

Ciclo de vida en cascada

Es el más utilizado, siempre que es posible, precisamente por ser el más sencillo. Consiste en descomponer la actividad global del proyecto en fases que se suceden de manera lineal, es decir, cada una **se realiza una sola vez**, cada una se realiza **tras la anterior y antes que la siguiente**. Con un ciclo lineal es fácil dividir las tareas entre equipos sucesivos, y prever los tiempos (sumando los de cada fase).

Requiere que la actividad del proyecto pueda descomponerse de manera que una fase no necesite resultados de las siguientes (realimentación), aunque pueden admitirse ciertos supuestos de realimentación correctiva. Desde el punto de vista de la gestión (para decisiones de planificación), requiere también que se sepa bien de antemano lo que va a ocurrir en cada fase antes de empezarla.

Ciclo de vida con prototipado

A menudo ocurre en desarrollos de productos con innovaciones importantes, o cuando se prevé la utilización de tecnologías nuevas o poco probadas, que las incertidumbres sobre los resultados realmente alcanzables, o las ignorancias sobre el comportamiento de las tecnologías, impiden iniciar un proyecto lineal con especificaciones cerradas.

Si no se conoce exactamente cómo desarrollar un determinado producto o cuáles son las especificaciones de forma precisa, suele recurrirse a definir especificaciones iniciales para hacer un **prototipo**, o sea, un producto parcial (no hace falta que contenga funciones que se consideren triviales o suficientemente probadas) y provisional (no se va a fabricar realmente para clientes, por lo que tiene menos restricciones de costo y/o prestaciones). Este tipo de procedimiento es muy utilizado en desarrollo avanzado.

La experiencia del desarrollo del prototipo y su evaluación deben permitir la **definición de las especificaciones más completas** y seguras para el productodefinitivo.  
   
A diferencia del modelo lineal, puede decirse que el ciclo de vida con prototipado repite las fases de definición, diseño y construcción dos veces: para el prototipo y para el producto real.

|  |  |
| --- | --- |
|  |  |

Ciclo de vida en espiral

El ciclo de vida en espiral puede considerarse como una generalización del anterior para los casos en que **no basta con una sola evaluación de un prototipo** para asegurar la desaparición de incertidumbres y/o ignorancias. El propio producto a lo largo de su desarrollo puede así considerarse como una sucesión de prototipos que progresan hasta llegar a alcanzar el estado deseado. En cada ciclo (espirales) las especificaciones del producto se van resolviendo paulatinamente.

A menudo la **fuente de incertidumbres es el propio cliente**, que aunque sepa en términos generales lo que quiere, no es capaz de definirlo en todos sus aspectos sin ver como unos influyen en otros. En estos casos la evaluación de los resultados por el cliente no puede esperar a la entrega final y puede ser necesaria repetidas veces.

El esquema del ciclo de vida para estos casos puede representarse por un bucle en espiral, donde los cuadrantes son, habitualmente, fases de**especificación, diseño, realización y evaluación**  (o conceptos y términos análogos).

En cada vuelta el producto gana en “*madurez*” (aproximación al final deseado) hasta que en una vuelta la evaluación lo apruebe y el bucle pueda abandonarse.

|  |  |
| --- | --- |
|  |  |

* 1. **Seleccionar un modelo para el proyecto:**

Debido a que no se conoce exactamente cómo desarrollar un determinado producto o cuáles son las especificaciones de forma precisa, suele recurrirse a definir especificaciones iniciales para hacer un prototipo, o sea, un producto y provisional. Se elige seguir el modelo de ciclo de vida prototipado. Puede comprenderse las fases de este ciclo en la imagen 1.

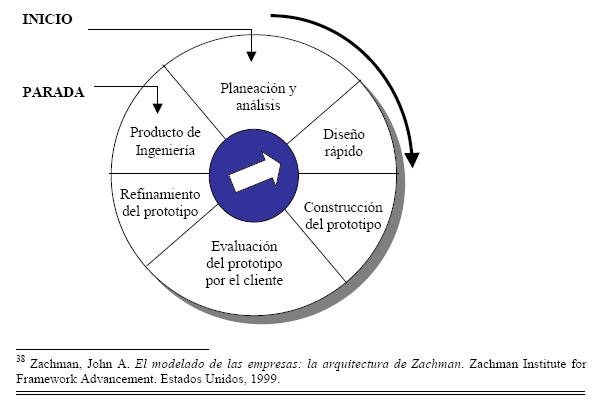


Imagen 1: Ciclo de vida prototipado evolutivo

1. **PROCESO DE INICIACIÓN, PLANIFICACIÓN Y ESTIMACIÓN DEL PROYECTO**
   1. **Establecer la matriz de actividades para el MCVS**

Nuestro diseño constara de 2 maquetas, y 2 prototipos evolutivos, es decir 2 iteraciones en cada una de las fases previamente detalladas, puede comprenderse esto en la imagen 2, mapa de actividades, y comprender los tiempos de ejecución en l imagen 3, Gantt.

* 1. **Asignar los recursos del proyecto**

En esta sección vamos a discriminar el uso de los recursos para la ejecución del proyecto.

**B.1. Recursos humanos:**

El software a desarrollar consta de 3 etapas ligadas entre sí, una etapa de diseño móvil, otra etapa de transición en el servidor y una última de interacción con el usuario en una pc local. Todo lo relacionado con el diseño móvil está a cargo de Perez Nicolas, el ensamblado en el server está dirigido por Buczacka Walter , mientras que la interface visible para el usuario estará bajo la supervisión de Gonzalez Gustavo. Esta redistribución será reforzada por otros miembros en momentos críticos para evitar atrasos muy significativos, la asignacion de tareas podrá observarse también en el Gantt.

**B.2. Recursos tecnológicos:**

La programación móvil estará enfocada a dispositivos con android 4.4 en adelante, y el código será escrito íntegramente en JAVA.

El servidor de la UNLa será escrito en PYTHON 2.7, y los datos serán almacenado con Workbrench 5.1.

Mientras que el software de escritorio, que generara y unificara los archivos GVsig, será escrito en JAVA y C.

* 1. **Definir el entorno del proyecto**

El proyecto está orientado a resolver un problema planteado por la Licenciatura en Gestión Ambiental, que tiene la necesidad de censar arboles en localidades determinadas, dicha información debe estar geolocalizada y puede o no, tener una imagen adjunta para describir aun mejor este árbol. La interface podrá ser utilizada por alumnos, docentes e incluso por cualquier otra persona que desee ayudar en el censado. Por lo que no se tendrá un logueado exigente en la aplicación. Para permitir que dicho software sea lo mas masivo posible, permitiendo solo algunos permisos especiales para los miembros claves del proyecto.

* 1. **Planificar la gestión del proyecto**

El proyecto fue planificado en primera instancia para que sea terminado el 1-11-2016, la distribución de las tareas en cada una de las etapas esta descripta en la imagen 3. En el diagrama de Gantt además vemos plasmados las 4 iteraciones pensadas para el proyecto.

1. **PROCESO DE SEGUIMIENTO Y CONTROL DEL PROYECTO**
   1. **Análisis de riesgos**

En una primera instancia, por medio del análisis del entorno pudimos identificar 3 riesgos, potenciales e inevitables, descritos en la imagen 4. El primero y el segundo consta del GPS, mientras que el tercero es sobre GvSig.

Luego de la construcción de la primer maqueta nos topamos con otro riesgo muy peligroso, es el del mal uso de la cámara de fotos, y no solo capturar imágenes que no sean de arboles si no también sacar la foto desde muy lejos, lo que haría que la ubicación de gps sea aun mas imprecisa.

* 1. **Realizar la Planificación de contingencias**
* El **plan de respaldo**. Para evitar la perdida de datos o para contemplar cambios abruptos en los ideales del cliente, se respaldan todos los archivos y documentos del proyecto en la plataforma GitHub, situada en: <https://github.com/NykolaiPerezVelez/REOT>

Además, no se decidió asignar tareas excluyentes a los integrantes del equipo, porque en caso de que alguno de ellos posea un problema serio sería muy complejo suplantarlo, por ello todos tenemos varias actividades para complementarnos. Contempla las contramedidas preventivas **antes** de que se materialice una amenaza. Su finalidad es evitar dicha materialización.

* El **plan de emergencia**. Suponiendo que alguno de los integrantes no puede cumplimentar alguna de sus tareas diagramamos un grafico de compensaciones de tareas, para paladar los déficits. Ver figura 5.   
  En la misma se expresa quien debe suplantar a cada persona en caso que la misma no pueda cumplir por si sola las tareas asignadas.

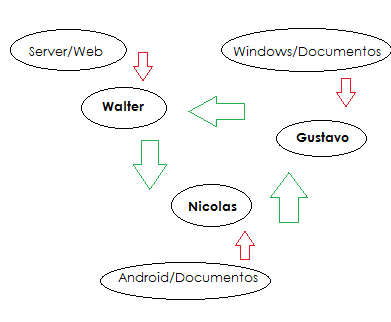


Figura 5: Plan de emergencia de relevos de integrantes

* 1. **Gestionar el proyecto**

Para una mejor gestión y administración, principalmente de los tiempos, se empleo un diagrama PERT desde el momento de la creación de una primer maqueta, en el mismo se aprecian las fechas teóricas pactadas y las fechas reales dadas en el proyecto, además que se distingue el camino critico a transitar para evitar problemas groseros de calendarios. Podemos encontrarlo en la Figura 6.

* 1. **Implementar el sistema de informes de problemas**

1. El primer problema critico con el que nos topamos, fue el poco conocimiento del cliente real al que estará destinado nuestro producto. El mimo fue solucionado parcialmente por el contacto vía mail con el mismo. Dicha solución es parcial por el delay con las respuestas y el bajo nivel de entendimiento que se tiene en este lenguaje de comunicación, se trabajo para contactar varias entrevistas personales, para ver si primer opinión tangible con el maquetado.
2. Luego nos encontramos con la limitación de las versiones de los celulares, vimos que no es para nada despreciable el pensar con que versiones de Android el programa estará disponible. Optamos porque sea de la versión 4.4 en adelante, debido a que son los celulares que se están comercializando en este periodo.
3. Por el ítems anterior surgió la idea, problema, de omitir el uso de android y hacer solo una interfaz online que se pueda ver desde un dispositivo mobil, dicha idea no fue tenida en cuenta para disminuir el tiempo de conexión a internet necesaria para el correcto funcionamiento del sistema.
4. Otro problema no despreciable es el problema de las conexiones a internet a la hora de enviar datos al servidor, para ello se opto por guardar una copia de la información en el celular, hasta que el servidor nos indique que todo ha sido guardado correctamente.
   1. **Archivar Registros**

Todos los registros de avances, o los que describen en profundidad lo ilustrado por este documento puede encontrárselo en el segmento Documentos de Registros de nuestro directorio online,

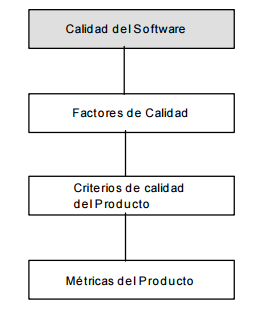
<https://github.com/NykolaiPerezVelez/REOT/tree/master/Docuentacion%20REOT>

1. **PROCESO DE GESTIÓN DE CALIDAD DEL SOFTWARE**

En este segmento nos centraremos en la calidad del producto, servicio o la satisfacción del cliente, sino en los medios para obtenerla. Por lo tanto, la gestión de calidad utiliza al aseguramiento de la calidad y el control de los procesos para obtener una calidad más consistente.

* 1. **Planificar la garantía de calidad del software**

Los modelos de calidad del software vienen a ayudar en la puesta en práctica del concepto general de calidad que vimos en el apartado anterior, ofreciendo una definición más operacional. Unos de los modelos de calidad más antiguos y extendidos es el de McCall [McCall, 1977], y de él han derivado otros modelos, como el de Boehm [Boehm, 78] o el SQM [Murine, 1988] En los modelos de calidad, la calidad se define de forma jerárquica. Es un concepto que se deriva de un conjunto de sub-conceptos, cada uno los cuales se va a evaluar a través de un conjunto de indicadores o métricas. Para el estudio de calidad de nuestro Software nos basaremos en el diagrama descripto en la Figura 7.

****

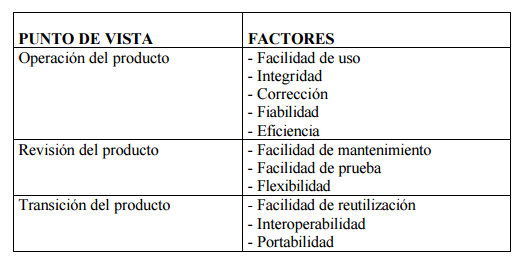
**Figura 7: Estructura de 4 niveles de la calidad del SW**

En el nivel más alto de la jerarquía se encuentran los FACTORES de calidad, que representan la calidad desde el punto de vista del usuario. Son las características que componen la calidad. También se les llama Atributos de Calidad Externos. Cada uno de los factores se descompone en un conjunto de CRITERIOS de calidad. Son atributos que, cuando están presentes, contribuyen al aspecto de la calidad que el factor asociado representa. Se trata de una visión de la calidad desde el punto de vista del producto software. También se les llama Atributos de Calidad Internos. Para cada uno de los criterios de calidad se definen entonces un conjunto de MÉTRICAS, que son medidas cuantitativas de ciertas características del producto que, cuando están presentes, dan una indicación del grado en que dicho producto posee un determinado atributo de calidad.

**Nota 1:** Una desventaja es que aún no ha sido demostrada la validez absoluta de ninguno de estos modelos. Las conexiones que establecen entre características, atributos y métricas se derivan de la experiencia, y de ahí que existan múltiples modelos.

* 1. **Desarrollar métricas de calidad**

Para desarrollar las métricas pertinentes a nuestro proyecto optamos por utilizar el modelo de McCall. El modelo de McCall se basa en 11 factores de calidad, que se organizan en torno a los tres ejes de la siguiente forma: (Figura 8)



**Figura 8: Estructura de las métricas de McCall**

Los factores de McCall se definen como sigue:

1. Corrección: Hasta qué punto un programa cumple sus especificaciones y satisface los objetivos del usuario. Por ejemplo, si un programa debe ser capaz de sumar dos números y en lugar de sumar los multiplica, es un programa incorrecto. Es quizás el factor más importante, aunque puede no servir de nada sin los demás factores.

2. Fiabilidad: Hasta qué punto se puede confiar en el funcionamiento sin errores del programa. Por ejemplo, si el programa anterior suma dos números, pero en un 25% de los casos el resultado que da no es correcto, es poco fiable.

3. Eficiencia: Cantidad de código y de recursos informáticos (CPU, memoria) que precisa un programa para desempeñar su función. Un programa que suma dos números y necesita 2 MB de memoria para funcionar, o que tarda 2 horas en dar una respuesta, es poco eficiente.

4. Integridad: Hasta qué punto se controlan los accesos ilegales a programas o datos. Un programa que permite el acceso de personas no autorizadas a ciertos datos es poco íntegro.

5. Facilidad de uso: El coste y esfuerzo de aprender a manejar un producto, preparar la entrada de datos e interpretar la salida del mismo.

6. Facilidad de mantenimiento: El coste de localizar y corregir defectos en un programa que aparecen durante su funcionamiento.

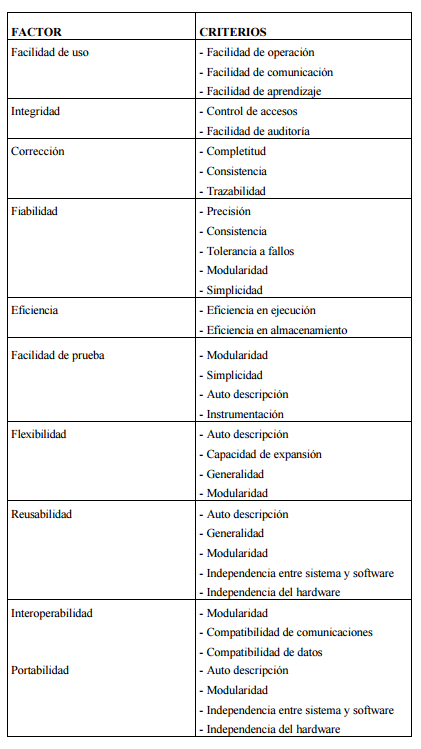
7. Facilidad de prueba: El coste de probar un programa para comprobar que satisface sus requisitos. Por ejemplo, si un programa requiere desarrollar una simulación completa de un sistema para poder probar que funciona bien, es un programa difícil de probar.

8. Flexibilidad: El coste de modificación del producto cuando cambian sus especificaciones.

9. Portabilidad (o Transportabilidad): El coste de transportar o migrar un producto de una configuración hardware o entorno operativo a otro. 10. Facilidad de Reutilización: Hasta qué punto se puede transferir un módulo o programa del presente sistema a otra aplicación, y con qué esfuerzo.

11. Interoperabilidad: El coste y esfuerzo necesario para hacer que el software pueda operar conjuntamente con otros sistemas o aplicaciones software externos.

Además estos factores poseen 23 criterios para la correcta evaluación. Los mismos están plasmados en la figura 9.



**Figura 9: Factor – Criterio**

Las métricas que desarrollaremos en este proyecto son las siguientes:

Fiabilidad = 1 - (número de errores/ número de líneas de código)

Facilidad de mantenimiento = 1 - 0.1 (número medio de días-hombre por corrección)

Portabilidad = 1 - (esfuerzo para portar / esfuerzo para implementar)

Flexibilidad = 1 - 0.05 (número medio de días-hombre por cambio)

Trabajando con las dos primeras iteraciones de nuestro maquetado, obtuvimos los siguientes índices.

Numero de errores= 21  
Número de líneas de código = 2092  
Numero medio de días hombre por corrección = 3  
Esfuerzo = 2  
Esfuerzo para implementar = 10  
número medio de días-hombre por cambio = 15

Dado estos rangos, obtuvimos las siguientes métricas:

Fiabilidad = 0,9899

Facilidad de mantenimiento = 0,7

Portabilidad = 0,8

Flexibilidad = 0,75

* 1. **Gestionar la calidad del software**

**NO SE QUE PONER ACA**

* 1. **Identificar necesidades de mejora de la calidad.**

Con el primer maquetado se identificaron 3 necesidades de mejoras de calidad para los modelos funcionales:

* Quitar las letras claves y poner las palabras completas para un mejor entendimiento y minimizar los errores del usuario
* Permitir a los usuarios que puedan editar los campos en todo momento, salvo luego del guardado en la memoria
* Recordarle al usuario por medio de un mensaje en pantalla que debe estar cerca del árbol para calcular la distancia.

1. **PROCESO DE EXPLORACIÓN DE CONCEPTOS**
   1. **Identificar las ideas o necesidades  
        
      Necesidades a satisfacer:**

* Censado rápido y móvil de arboles en municipios
* Archivar censos en servidor y poderlo visualizar desde cualquier pc
* Comprensión de datos por medio de georeferenciacion usando gvsig

|  |  |  |  |
| --- | --- | --- | --- |
| ***Decripción*** | ***Impacto*** | ***Probabildad*** | ***Accion a Seguir*** |
| -  Dependencia absoluta con GPS | El dispositivo no cumplirá las solicitudes si no posee conexión de GPS en el instante de capturar la imagen | Alta | Incorporar ubicación manual del usuario, si el cliente lo desea |
| -   Error de entre 30 y 100 metros, error controlado y causado por el Departamento de defensa de lo EEUU | Las imágenes tomadas podrán estar hasta una cuadra distante a lo dicho por el GPS, cifra muy significativa para el ideal del SW | Media | Usar convergencia de serie bidimensional en un entorno acotado para disminuir el error en (n/n+33) |
| - Fuertes de incertidumbres causada por la escasa documentación del GvSig | Desconocimiento parcial de teoría detrás del uso de segmentos del GvSig | Baja | Ver que usos necesita y usa el usuario |

**Figura 4**: Gestión de Riesgos.

**Ideas del equipo de software:**

* Crear una aplicación para android, disponible para cualquier usuario que quiera cumplir las necesidades del cliente.
* Utilizar la posición administrada por el GPS, corrigiendo los errores utilizando convergencia de puntos
* Crear una interface web para conseguir los datos desde cualquier pc
  1. **Formular las soluciones potenciales**

Para combatir los riesgos descriptos en la imagen 4, proponemos las siguientes soluciones.

* En caso que la conexión GPS no esté disponible dejar que el usuario seleccione la posición por medio de un mapa en forma manual. Esta solución teniendo en cuenta que no todos los lugares son claros a la hora de visualizar desde el GPS, la descartamos ya que podría ser una solución con daños colaterales inmanejables. Un ejemplo claro es censar arboles en una plaza o parque grande, como las calles y numeraciones estarán muy lejos será difícil ubicarnos de forma manual.
* Si el GPS funciona, tiene un error de entre 30 y 100 mts, para minimizar este error, proponemos la solución de convergencia de puntos, para bajar este error en un (, descripto en la referencia” a”.
* Diseñar en archivo autoejecutable que interprete los datos y los cargue en GvSig sin que el usuario deba hacer nada mas que un doble click.

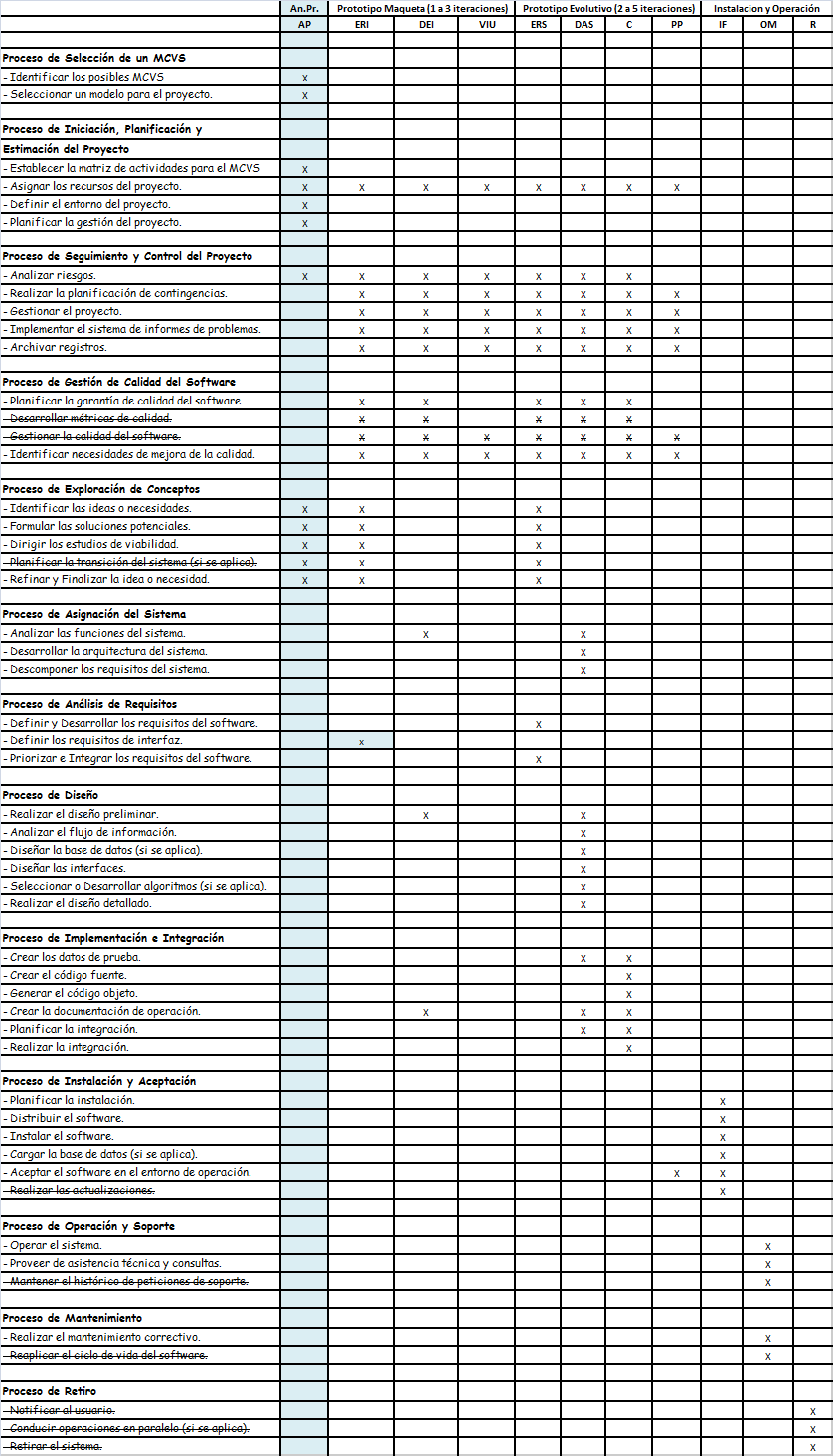


Imagen 2: Mapa de actividades

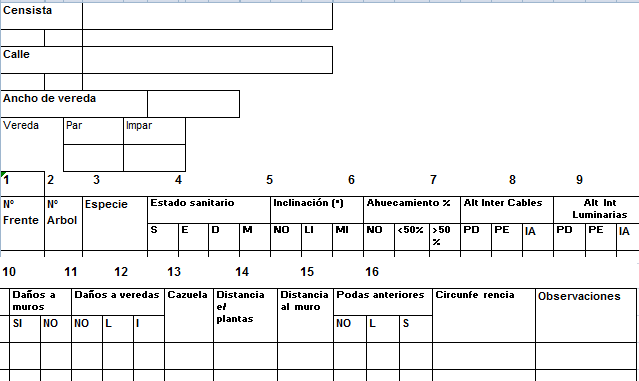
* 1. **Dirigir los estudios de viabilidad**

Los riesgos descriptos en la imagen 4 y las solucione planteada en el apartado anterior, se consideran viables para el proyecto, por los siguientes ítems:

* **Viabilidad operacional: Los lenguajes de programación a implementar, el tiempo disponible, y las condiciones de producción se consideran totalmente viables por los integrantes del equipo, por el conocimiento previo de todas las partes.**
* **Viabilidad del mercado: El proyecto tiene una ventaja que muchos otros proyectos desearían, el mismo no será sumergido a un mercado desconocido o inestable, el software será diseñado previo a un pedido, por lo que posee un mercado seguro esperando el desarrollo.**
* **Viabilidad conceptual: Si bien hay riesgos, y los requisitos fueron, en primera instancia muy volátiles, el comprendimiento del problema y la factibilidad de la solución han sido comprendidas por los encargados de este programa.** 
  1. **Refinar y Finalizar la idea o necesidad**

Uno de los principales problemas sobre el entendimiento del problema a resolver, fue desde un primer momento, la comprensión de lo datos a almacenar sobre el arbolado, por nuestro desconocimiento sobre el tema. Este problema fue resuelto, y refino nuestras ideas sobre la solución por medio de las respuestas del usuario, descriptas en el enlace 1, y plasmadas en la imagen 5.

Link1: <https://docs.google.com/forms/d/1duxmYBjoJQfQu0hXVa6AqDGU1VJIHG-53ZO6XSiB2dI/edit>



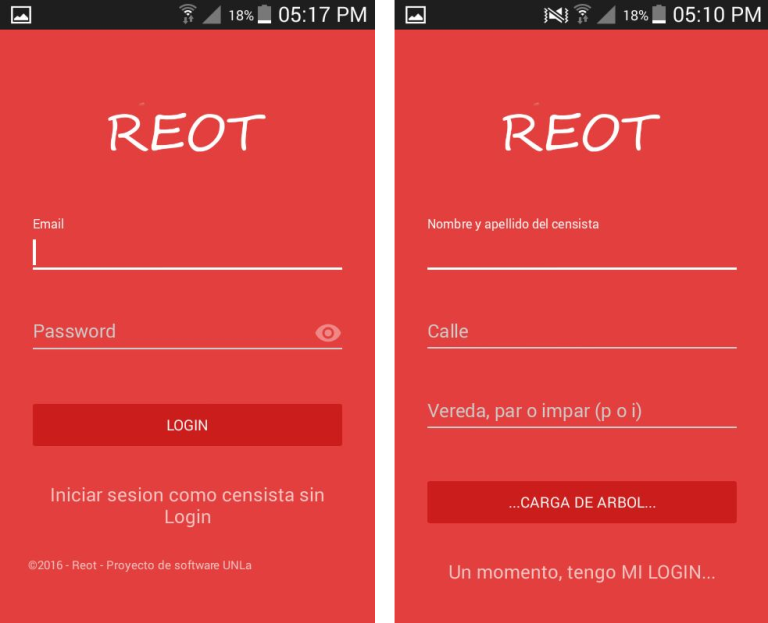
**Imagen 5:** Necesidad del usuario, datos que necesita el censista

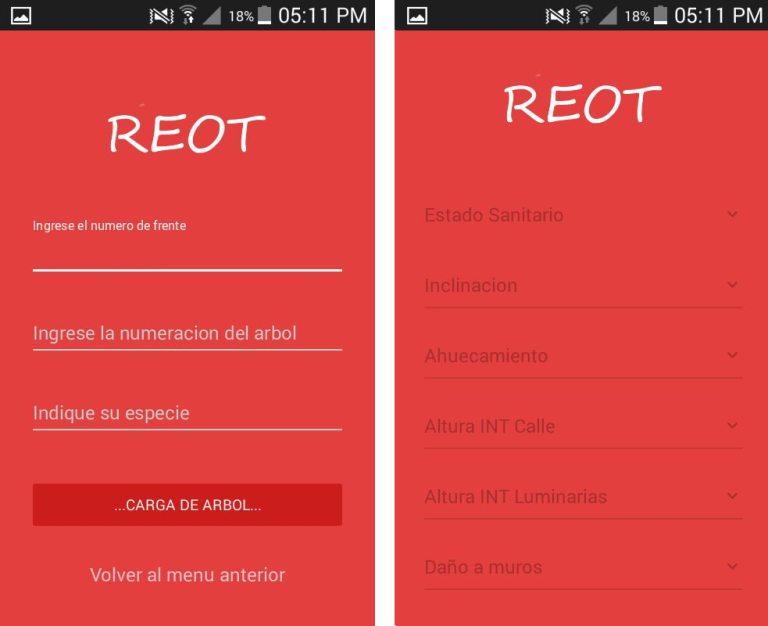
1. **PROCESO DE ASIGNACIÓN DEL SISTEMA**
   1. **Analizar las funciones del sistema**

En este apartado analizaremos las funciones que posee la segunda iteración del maquetado. Y Haremos algunas menciones a la primer maqueta, que solo poseía una función (la de sacar fotos). Haremos este recorrido mostrando las funciones operativas que tiene la aplicación, en este segundo bucle de operaciones de software.

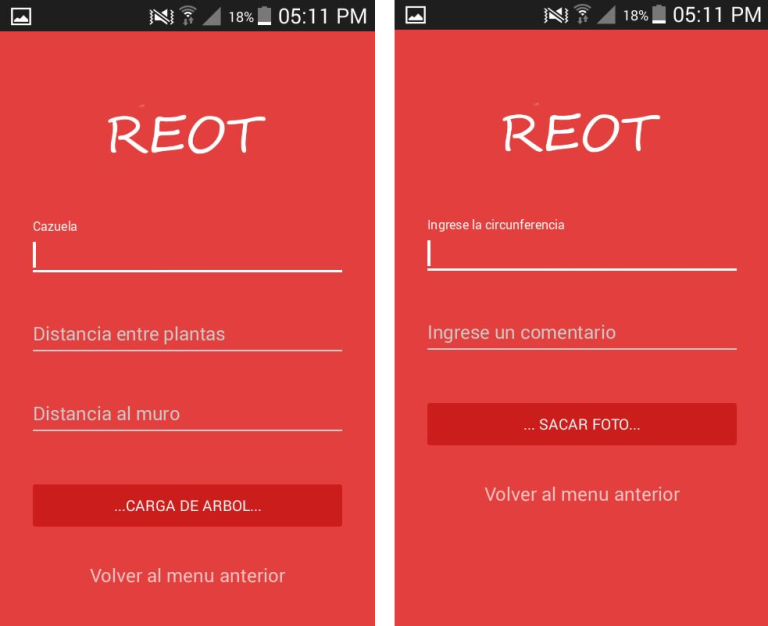
El programa REOT, consta con una primer pantalla (Inicio), donde uno puede elegir si hacer o no Login, el software fue diseñado para que cualquier usuario pueda contribuir con información, por lo que sospechamos que la mayor cantidad de clientes no tendrá login, los que así lo tengan solo evitaran hacer la carga de alguno de sus datos. Figura 8.

Luego se le pide al cliente sus datos personales y el de la calle a censar (Figura 9). A posteriori la aplicación pide el ingreso de datos pertinentes para el arbolado, todos los datos pueden ser nulos, salvo los esenciales, como el de la calle (Figura 10 a 13)

 **Figura 8: Login Figura 9: Datos personales**

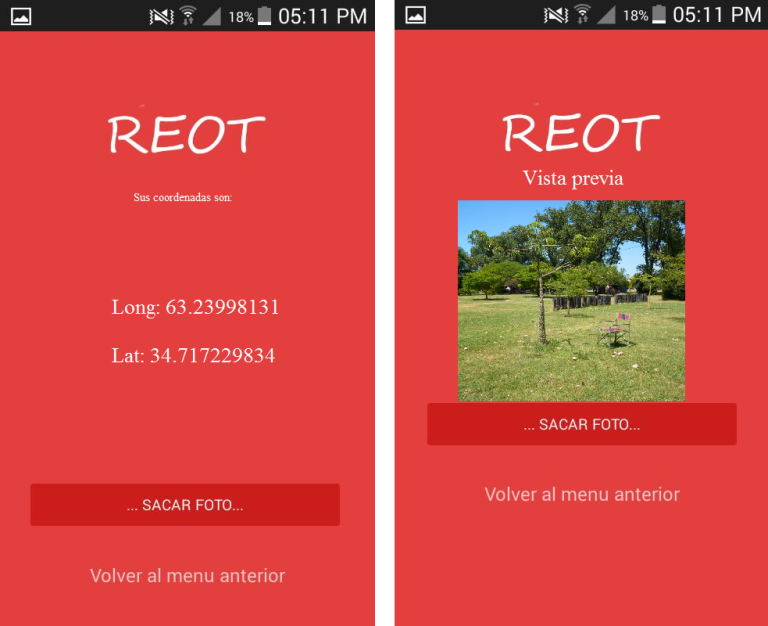
****

**Figura 10 y 11: Datos de arbolado 1 y 2**

****

**Figura 12 y 13: Datos de arbolado 3 y 4**

Luego tenemos la ubicación de GPS, que el usuario elige en que momento calcularla, debe ponerse lo mas cerca del árbol posible (Figura 14) y por ultimo en la Figura 15 tiene la opción de sacar fotos e ir viendo sus vistas previas, antes del guardado.



# **Proceso de análisis de Requisitos**

* 1. ***Requisitos de Usuario***

**RU1**. Integrar datos excel con las herramientas gvSIG, hospedadas en la cpu personal del Usuario.

**RU2.** Capturar imágenes de árboles desde un dispositivo móvil

**RU3**. Completar una planilla con los datos pertinentes correspondientes a la imagen capturada.

**RU4**. El usuario puede cancelar el proceso en cualquier momento.

* 1. ***Requisitos del Sistema***

**RS1.** El sistema móvil, enviará las fotos georeferenciadas y etiquetadas al servidor de la UNLa

**RS2**. El sistema intermedio, alojado en el servidor, enviará los datos al usuario y guardará una copia de los datos en una base de datos Workbrench 5.5

**RS3.** El sistema que el usuario dispondrá en su pc personal recibirá los datos del servidor, los transformará en una planilla excel, y ese excel será codificado para la lectura gvSig

* 1. ***Requisitos Funcionales***

A continuación se enumera la lista de Requisitos Funcionales de la aplicación, que indican las funcionalidades que formarán parte del sistema:

**RF1.** El sistema permitirá capturar fotografías y adjuntar una descripción

**RF2.** El usuario podrá elegir si la fotografía merece o no formar parte del relevamiento de datos de arbolado.

**RF3.** El sistema georeferenciará las imágenes capturadas

**RF4.** El sistema se conectará con el servidor de la UNLa y enviará los datos al mail del usuario.

**RF5.** El sistema permitirá la interpretación de los datos por medio del motor gvSig

**RF6.** El sistema tendrá una interface web para la administración de datos y la vista de los mismos.

**RF7.** El sistema permitirá y exigirá un login para el uso.

**RF8.** El sistema determinará si las fotografías sacadas son o no de árboles.

* 1. ***Requisitos No Funcionales***

El siguiente apartado tendrá el objetivo de indicar restricciones sobre los Requisitos No Funcionales:

**RFN1.** El usuario no verá ni participará de los procesos

internos de conversión y obtención de datos.

**RFN2.** El software es íntegramente dependiente de la red GPS, en los momentos donde ésta no esté funcionando la aplicación no será utilizable.

**RFN3.** El sistema necesita conexión de internet inalámbrica.

**RFN4.** El sistema tendrá una aplicación android (superior a version 4.4) o posterior.

**RFN5.** El sistema tendrá una aplicación de escritorio (Windows)

**RFN6.** El sistema se desarrollará en JAVA (Android), Python (Servidor y Windows).

**RFN7.** El servidor contará con el Sistema Operativo Debian Jessie 10.0

**RFN8.** El sistema móvil no se encargará de la vista de información previamente subida.

1. **PROCESO DISEÑO PRELIMINAR**

Aquí se definen los pasos realizados para la creación de las dos maquetas o mejor dicho sus dos iteraciones.

1. **Realizar el diseño preliminar**

Para el diseño preliminar mobil, se utilizo la aplicación Android Studio, y las herramientas de Material Desing para Android. En la primer maqueta se construyo una aplicación que simplemente capturaba imágenes y conseguía la posición. Luego en el 2do maquetado, posterior al contacto con los clientes y la adquisición de los verdaderos valores buscados por el cliente, se construyo la 2da maqueta con los datos necesarios y pedidos en primera instancia.

La misma cuenta con una interfaz simple y amigable para cualquier persona que este medianamente familiarizada con la plataforma Android.

Es muy simple su utilización, solo campos vacios que deben ser rellenados, o incluso algunos con opciones cerradas que facilitan aun mas al usuario el uso. La aplicación está disponible en:

<https://github.com/NykolaiPerezVelez/REOT/tree/master/REOT>

El uso y el entendimiento de cómo funcióna la aplicación se lo puede ver de las figuras 9 a 13, donde se ve pagina por pagina el maquetado.

El manual de la maqueta esta en:

<https://github.com/NykolaiPerezVelez/REOT/Maqueta/REOT>

1. **PROCESO DE IMPLEMENTACIÓN E INTEGRACIÓN**
2. **Crear la documentación de operación**

Todo el software, se lo puede encontrar documentado y versionado en:

<https://github.com/NykolaiPerezVelez/REOT/tree/master/Docuentacion%20REOT>

1. **PROCESO DE VERIFICACIÓN Y VALIDACIÓN**

La verificación y validación es una actividad que juega un papel importante en la consecución de procesos y productos de calidad. Por ello y dado los problemas que históricamente han surgido en los desarrollos software, su estudio y mejora supone un avance importe para la futura evolución de las tecnologías de la información.

En el presente documento se establece la evolución de dicha actividad y su actual impacto en los distintos tipos de organizaciones software que las implementan, con el fin de poder extraer algunos de los conceptos más importantes en su implantación y desarrollo, y así elaborar una guía de buenas prácticas que permita seleccionar las mejores opciones para realizar las pruebas de software en función de las características de una organización.

1. **Planificar la verificación y validación**

Hoy en día existen dos paradigmas principales que permiten la aplicación de las actividades de V&V en un proyecto, el análisis estático y el dinámico. El análisis estático necesita de la ayuda de una herramienta que analice la representación estática del sistema (diagramas de flujo, casos de uso, código fuente, otros) con el fin de descubrir errores.

Este tipo de análisis implica la no ejecución de modelos, ni del código fuente para determinar las posibles fallas. Se trata por lo tanto de verificar los “productos” obtenidos durante el desarrollo del software. Para llevarlos a cabo es habitual es uso de técnicas de inspección, éstas no resultan normalmente un reemplazo de las pruebas de software, sino que las complementan, enfocándose en aquellas diferencias que no son posibles detectar usando únicamente pruebas.

Por el contrario, el análisis dinámico implica normalmente la ejecución del código fuente del software con datos de prueba, con el fin de evaluar en tiempo real los diferentes escenarios soportados por el programa.

Las técnicas de inspección comprenden el análisis automático del código fuente y la verificación formal. Por el contrario, las técnicas estáticas sólo pueden comprobar la correspondencia entre un programa y su especificación (verificación), no pudiendo demostrar que el software es operacionalmente útil.

En nuestro proyecto hemos planificado la V&V en 2 segmentos, técnicas estáticas, y técnicas dinámicas, donde:

* **Las técnicas estáticas** son las primeras comprobaciones que se aplican al software. Son técnicas cuyo objetivo es el de mejorar la calidad de los productos del software, ayudando a los ingenieros a reconocer y arreglar sus propios errores en etapas tempranas del proceso de desarrollo sin ejecuta el código. Para ello, buscan defectos sin ejecutar el código, llevándose a cabo una vez se encuentra escrito el código. Estas técnicas tienen que ver con análisis y el control de las representaciones del sistema, es decir con los diferentes modelos construidos durante el proceso de desarrollo de software, tales como los documentos de requerimientos, los diagramas de análisis y de diseño y el código fuente. En esta categoría entran, por ejemplo, las revisiones, las inspecciones de programas, la verificación formal (basada en el uso de notaciones formales con base matemática) y las herramientas de análisis estático (muchas de ellas provistas hoy en día como parte del compilador, como por ejemplo la detección de variables no utilizadas, código inalcanzable, etc.).
* **Las técnicas dinámicas**, también conocidas como pruebas, se basan en ejercitar una implementación. Por lo tanto, sólo pueden ser aplicadas si existe una versión operativa o ejecutable del producto. A pesar de que las técnicas de verificación estáticas son usadas cada vez más, la prueba de los programas sigue siendo la técnica predominante en la validación y verificación. La prueba involucra ejecutar el programa proveyéndole entradas con el fin de detectar la mayor cantidad de defectos posibles. Son diseñadas con el propósito de revelar la presencia de defectos, por ello se dice que un caso de prueba es exitoso cuando los logra poner en evidencia. Además, son consideradas como la única técnica de validación para los requerimientos no funcionales, debido a que el software debe de ser ejecutado para ver su comportamiento, proporcionando confianza en el software.

1. **Ejecutar las tareas de verificación y validación**

A continuación se realizaran las técnicas estáticas y dinámicas explicadas más arriba.

B.1. Manejo estático:

* Revisión informal del código.

1. La revisión del código, que consiste en una forma de evaluación sistemática del código, normalmente elaborada por el propio desarrollador. Se realiza de forma individual estableciendo una depuración de un determinado código. Se realizo por medio de Android Studio, donde se dejo de usar el celular y se opto por el uso del emulador, para ver la compatibilidad con otros sistemas operatives Android.
2. La programación en parejas, que consiste en una revisión del código en el que dos personas trabajan juntas, en colaboración, para revisar de forma periódica un determinado producto. Allí se unieron los dos proyectos, la maqueta 1, y la maqueta dos, para lograr obtener la ultima iteración del maquetado. Se realizo la verificación de ambos códigos y se solucionaron problemas de compilación que surgieron en la unificación de los módulos.

* Revisión formal del código.

Las revisiones formales siguen una secuencia de actividades basada en gran medida en la inspección del software, las cuales son detalladas a continuación, **no fueron realizadas aun, se espera realizarlas cuando el cliente vea el software por primera vez en 15 días.**

1. Planificación: La planificación se produce cuando el desarrollador completa su producto, entonces se forma un grupo de inspección y se designa un moderador. El moderador debe de asegurarse que el producto satisfaga el criterio de inspección. Por último, se le asignan diferentes roles a las personas que integran el grupo de inspección, así como la planificación de tiempos y recursos necesarios.
2. Familiarización: El paso de familiarización sólo se realiza si los inspectores no conocen el desarrollo del proyecto. Es opcional pero no por ello menos importante, ya que en esta etapa se dará al grupo de inspección el contexto a cubrir por las inspecciones.
3. Preparación: Los inspectores se preparan individualmente para la evaluación en la reunión, estudiando los productos y el material relacionado. En este paso es aconsejable la utilización de listas de chequeos para ayudar a encontrar defectos comunes. El tiempo que pueda llevar esta etapa va a depender de la familiarización del inspector con el trabajo que debe analizar.
4. Examen: En esta etapa, los inspectores se reúnen para analizar su trabajo individual de forma conjunta. El moderador deberá asegurarse que todos los inspectores se encuentran suficientemente preparados. La persona designada como lector presenta el producto interpretando o parafraseando el texto, mientras que cada participante observa en busca de defectos. Es recomendable que este examen no dure más de 2 horas ya que la atención en busca de defectos va disminuyendo con el tiempo. Al terminar con la reunión, el grupo determina si el producto es aceptado o se debe volver a realizar para una posterior inspección.
5. Rehacer: El autor corrige todos los defectos.
6. Seguimiento: El moderador chequea las correcciones del autor. Si el moderador está satisfecho, la inspección está formalmente completa, y el producto es puesto bajo el control de configuración.
7. **Recoger y analizar los datos de las métricas**
8. **Planificar las pruebas**

Se ejecutaran las pruebas de verificación y validación, acordes a lo acordado en el inicio XI.B. Detallando lo ocurrido en la revisión formal del código. Esta tarea sea plasmada con las siguientes especificaciones de casos de prueba: (se recuerda que en 15 días se realizaran más informes formales). (Ver Figura 9).

1. **Desarrollar las especificaciones de las pruebas**

En la Figura 9 puede apreciarse los casos de prueba diseñados y ejecutados a la fecha. Restan otros casos de prueba, como:

- Envió al servidor de los datos cargados por el usuario

- Testeo de excepción en caso de carencias en el internet

- Guardar imágenes y datos en el celular sin enviar

- Ir atrás en cualquier parte de la aplicación y verificar si los datos quedan o no guardados.

- Probar aplicación sin conexión GPS.

1. **Ejecutar las pruebas**

Varias de las pruebas fueron ejecutadas, los resultados se los aprecian a continuación:



**Figura 9: Especificación de casos de prueba**

Imagen 3 Diagrama de Gantt