Universidad ORT Uruguay

Facultad de Ingeniería

Bernard Wand Polak

Ingeniería de Software en la Práctica

Obligatorio

Docente: Luis Barrague

Nombre: Mauricio Pastorini
Número de estudiante: 190388

Nombre: Franco Gario
Número de estudiante: 195469

Nombre: Fausto Sánchez
Número de estudiante: 187787

Declaración de auditoría

Declaramos ser autores de este documento y sus anexos para un obligatorio de Ingeniería de software en la práctica.

Resumen ejecutivo

Introducción

El presente documento provee una descripción general del producto y una especificación del plan de proyecto para su construcción.

Se utiliza el formato del documento 302 para el formato de la documentación

Proposito

Se solicitó crear un plan de proyecto y ejecutarlo para construir un software a nuestra elección. Nosotros hemos definido el alcance y lo validamos con nuestro profesor. En la siguiente sección se realizará su especificación detallada.

Metodología usada

Elegimos una metodología ágil para planear y ejecutar el proyecto. Los artefactos, eventos y ceremonias utilizadas, están basadas en el libro de Mike Cohn: *Agile Estimating and Planning.*

Referencias

Documento 302 - FORMATO DE TRABAJOS FINALES DE CARRERAS

Índice

2. Análisis del producto

2.1 Análisis del problema

2.1.1 Introducción

En este documento se establece la planificación para el sistema a producir. El sistema se basa en poder regalar artículos a gente que más lo necesite. El sistema va a permitir publicar productos, con sus imágenes y ubicación, para que luego otra persona lo vea y pueda contactarse con la persona que lo publico para ir a buscar el producto.

Todo se basa en la solidaridad de las comunidades. Creemos que con este sistema y el apoyo de las personas podemos ayudar a muchas personas con necesidades.

La idea para desarrollar esta aplicación surge para satisfacer la necesidad de la misma que encontramos dentro de un grupo de facebook que se dedicaba a la publicacion y difusion de productos que los usuarios ya no precisaban y otros no.

2.2.2 Análisis del entorno

Haciendo un análisis del entorno y el funcionamiento del grupo de Facebook previamente mencionado notamos la ineficiencia del mismo. Para poder entrar al grupo para ver las publicaciones o publicar un producto de así desearlo, se debe solicitar un permiso el cual esta demorado meses en ser otorgado. Por

2.2.3 Competencia

2.2 Definición del alcance del producto

2.2.1 Introducción

A continuación se mostraran las historias de usuario que seleccionamos. Estas las creamos en tarjetas con un formato específico según el libro de Kohn.

2.2.2 Recursos claves del dispositivo

GPS, almacenamiento y Cámara

2.2.2 Formato

Las historias de usuario siguen el siguiente formato:

a. Frente
    i. Descripción: Oración de no más de 2 líneas que describa la historia. Cumplimos con INVEST para la especificación de las mismas.
    ii. Nota: Aquí se enumeran temas para discutir sobre las historias en su respectiva ceremonia.

iii. Estimación: Cuanto estimamos que llevaran las historias de usuario.

b. Dorso
   i. Test: Se detallan los criterios de aceptación de las historias de usuario.

2.2.3 Unidades y Escala utilizada para la estimación

2.2.3.1 Unidades de estimacion

Utilizaremos la unidad de un dia ideal.

2.2.3.2 Escala de estimacion

La escala de estimación que utilizaremos es la de fibonacci. Se suele llamar la escala de Cohn ya que no es exactamente Fibonacci y Mike Cohn fue quien la remarcó.

2.2.3 Product Backlog

2.2.3.1 Historia 1

Descripción: Como usuario quiero poder publicar un producto para que lo puedan venir a buscar.
Nota:

- Atributos de la publicación.

Estimacion: 40
Tests:

- Debe quedar el registro en la tabla correspondiente de la base de datos.
- Debe mostrar un mensaje de publicado.
- Si los campos no son correctos, notificarlo.
- Si esta caido el servicio, notificarlo.
- Debe tener nombre, descripción, coordenadas, fotos.

2.2.3.2 Historia 2

Descripción: Como usuario quiero poder ver los productos que hay publicados de mi país para limitarme a mi frontera.
Nota:

- Qué información se muestra de ellos

Estimacion: 3
Tests:

- No se muestren productos fuera del país

2.2.3.3 Historia 3

Descripción: Como usuario quiero poder filtrar los productos para una consulta mas especifica.
Nota:

- Filtros predeterminados

Estimacion: 5

Tests:

- Poder filtrar por categoría de productos
- Poder filtrar por ubicación

2.2.3.4 Historia 4

Descripción: Como usuario quiero poder ver imagenes de los productos publicados para saber sus condiciones y tener mayor confianza.

Nota:

- Donde se van a poder ver.

Estimacion: 5

Tests:

- Se debe poder ver una galería de fotos en los detalles del producto seleccionado.
- Se debe poder ver al menos una foto.

2.2.3.5 Historia 5

Descripción: Como usuario quiero ver en el mapa donde están aproximadamente los productos para ver cuales tengo cerca.

Nota:

- Mostrar iconos o imágenes del producto?

Estimacion: 20

Tests:

- Se deben mostrar los productos en el departamento de mis coordenadas.

2.2.3.6 Historia 6

Descripción: Como administrador quiero moderar los productos que se publican antes que se muestran para tener mayor control sobre que se publica.

Nota:

- Poder eliminar productos?
- Poder modificar títulos y descripciones?

Estimacion: 5

Tests:

- Eliminada la publicación que se elimine de la Base de Datos
- Modificado el título o descripción de un producto que al verlo desde la aplicación se vean reflejados los cambios.

2.2.3.7 Historia 7

Descripción: Como administrador quiero poder dar de baja un usuario para controlar quienes usan la aplicación.

Nota:

- Borrado lógico o borrado físico?

Estimacion: 3

Tests:

- Dado de baja un usuario, que no pueda loguearse más.

- El borrado debe ser lógico. El registro del usuario debe continuar existiendo en la Base de Datos.

2.2.3.8 Historia 8

Descripción: Como administrador quiero poder ascender a administrador un usuario.
Nota:
- El usuario debe existir?

Estimacion: 1
Tests:
- El usuario debe existir en el sistema.
- Debe poder hacerse de un usuario a la vez

2.2.3.9 Historia 9

Descripción: Como administrador quiero poder hacer todo lo que hace un usuario.
Nota:
- Un administrador es un usuario con más privilegios?

Estimacion: 1
Tests:
- Un administrador debe poder hacer todo lo que puede hacer un usuario.
- Un administrador debe poder moderar productos.
- Un administrador debe poder dar de baja usuarios.
- Un administrador debe poder designar nuevos administradores.

2.2.3.10 Historia 10

Descripción: Como usuario quiero poder registrarme para poder luego loguearme.
Nota:
- Qué tipos de registros? Por facebook? Google?

Estimacion: 13
Tests:
- Los usuarios deben poder registrarse solo por la aplicación

2.2.3.11 Historia 11

Descripción: Como usuario quiero poder loguearme para poder usar el sistema en su totalidad.
Nota:
- N/A

Estimacion: 20
Tests:
- Dado un usuario registrado con nombre de usuario X y contraseña Y, al intentar loguearme con un usuario y contraseña Y el sistema debe dar un mensaje de éxito.
- Dado un usuario registrado con nombre de usuario X y contraseña Y, al intentar loguearme con un usuario y contraseña H el sistema debe dar un mensaje de error.

2.2.3.12 Historia 12

Descripción: Como usuario quiero poder usar la cámara para sacar fotos de mis productos al momento de publicar.
Nota:

- Sacar fotos al momento de publicar?
- Subir fotos del carrete?

Estimacion: 8
Tests:

- Se abre la cámara correctamente
- Se guarda la foto correctamente

2.2.3.13 Historia 13

Descripción: Como usuario quiero poder ver los datos de contacto del publicador una vez que elijo el producto para poder ir a buscarlo.
Nota:

- Qué datos se muestran?
- Se guardan los datos del publicador?

Estimacion: 8
Tests:

- Se deben mostrar el nombre, el celular y el mail.
- Se deben mostrar los datos en un popup
- Se deben enviar al usuario los datos del publicador.

2.2.3.14 Historia 14

Descripción: Como usuario quiero que me notifiquen cuando alguien quiere un producto mío para estar al tanto de la situación de mis productos.
Nota:

- Notificamos vía email? Vía push notifications?

Estimacion: 2
Tests:

- Luego de que alguien eligió un producto mio, recibir el email correctamente.

2.2.3.14 Historia 15

Descripción: Como usuario quiero poder calificar al otro usuario luego de ir a buscar el producto para mostrar mi experiencia con el mismo.
Nota:

- Cuantas estrellas?

Estimacion: 8
Tests

- Se debe poder elegir entre 5 estrellas

3. Gestión del proyecto

3.1 Introducción

En esta sección se detalla cómo realizamos la gestión de nuestro proyecto.

3.1.1 Aclaración importante

Es importante que notemos que según las metodologías ágiles:

1. Los release plannings se realizan antes de cada iteración. En estas ceremonias se realizan las estimaciones de las historias de usuario según las que entren al mismo, esto por cumplir con las condiciones de satisfacción del release.
2. El desglose de las historias de usuario en tareas, se realizan en las Sprint planning antes de comenzar el Sprint, junto con su priorización y estimación.

Como el cliente desea una documentación completa para el plan del proyecto, estas reuniones con sus resultados las realizamos antes de la primera entrega. Esto para tener la mayor cantidad de información posible y así, una más rica documentación.

De todos modos, hay datos como los resultados de los “burndown charts” que se obtienen a medida de la iteración y no se pueden extraer previamente. Por lo que para este tipo de irradiadores de información mostraremos un *template* que se completara al transcurrir la iteración.

3.2 Release Plan

3.2.1 Introducción

En esta sección se mostrarán los criterios que utilizamos para planear nuestros releases y además los resultados de las ceremonias de los “Planning release meeting”.

El proyecto tiene dos importantes releases, el primero que es enfocado a la entrega del plan de proyecto completo y el segundo que es la implementación del mismo con su documentacion e implementacion.

3.2.2 Formato utilizado para la planificación de un release

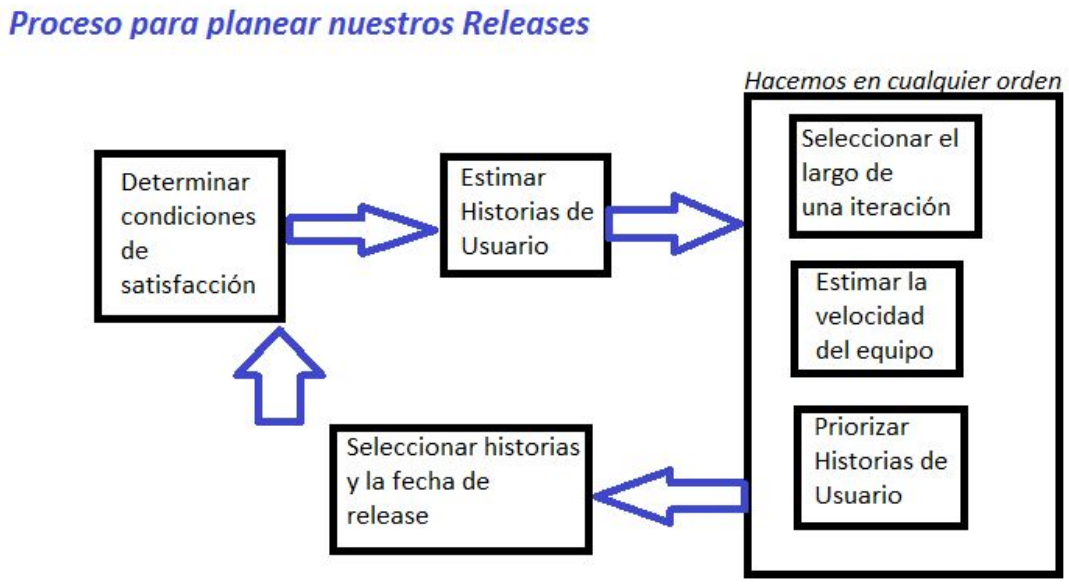
1. **Determinar condiciones de satisfacción:** Expectativas de lo que es el producto, qué características consideramos nosotros que se deben cumplir para que el producto sea satisfactorio.
2. **Estimar historias de usuario:** Estimamos las historias de usuario que entran en el próximo release.
3. **Seleccionar el largo de una iteración:** cuánto tiempo será cada sprint.

4. **Estimar la velocidad:** determinamos según nuestras experiencias y suposiciones, cuanto es aproximadamente la velocidad del equipo.
5. **Priorizar user stories:** priorizamos las historias de usuario por si en el peor de los casos de no llegar al alcance acordado con el cliente, podamos tener las historias más importantes.
6. **Seleccionar historias y fecha de release:** verificamos cuantas historias de usuario podemos incluir en el release según nuestra velocidad y la fecha de release.

   Como la fecha ya está fijada de las entregas, utilizaremos “Date driven” para determinar cuántas historias podemos realizar para el primer release.

$$Cantidad\ de\ días\ ideales\ =\ cantidad\ de\ iteraciones\ \times velocidad\ del\ equipo$$

3.2.3 Etapas del proceso de planeo de Release



3.2.4 Resultado de ceremonia “Release Planning Meeting 1”

3.2.4.1 Introduccion

Esta cerememonia se llevo a cabo el **XXXXXX** y los resultados fueron los que se mencionan en esta sección.

3.2.4.2 Condiciones de satisfaccion

3.2.4.3 Historias de usuario estimadas

3.2.4.4 Largo de cada iteracion

3.2.4.5 Priorizacion de las historias de usuario

3.2.4.6 Seleccionar historias de usuario y fecha de release

3.2.5 Resultado de ceremonia “Release Planning Meeting 2”

3.2.5.1 Introduccion

Esta ceremonia se debería llevar a cabo luego del primer release, pero como el cliente solicita tener una documentación sobre el plan lo más detallado posible, entonces se realiza una reunión previa a la finalización del Release 1.

Se llevó a cabo el **XXXXXX** y los resultados fueron los que se mencionan en esta sección.

3.2.5.2 Condiciones de satisfaccion

3.2.5.3 Historias de usuario estimadas

3.2.5.4 Largo de cada iteracion

3.2.5.5 Priorizacion de las historias de usuario

3.2.5.6 Seleccionar historias de usuario y fecha de release

3.3 Sprint planning

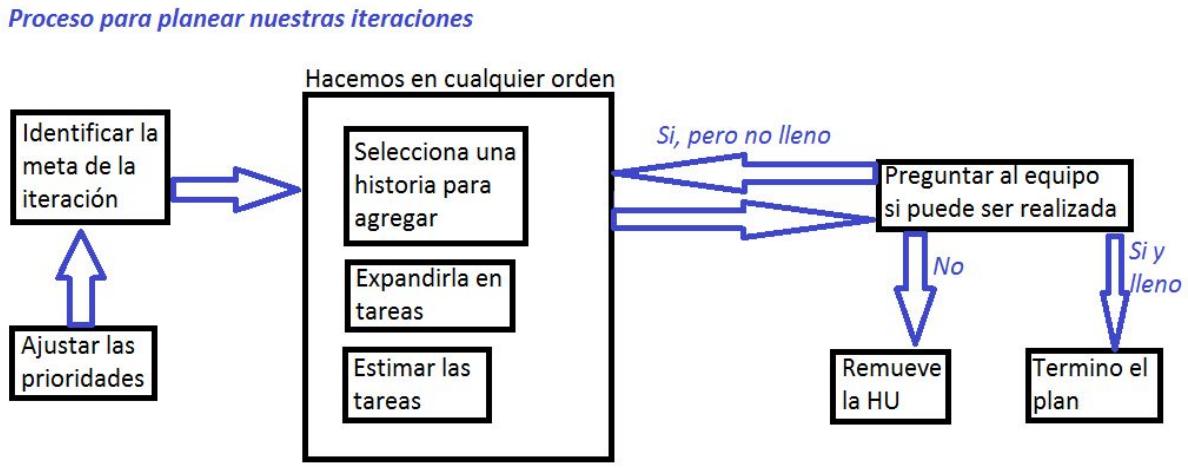
3.3.1 Introducción

En esta sección se detallan el formato para el sprint planning, las etapas del proceso y los resultados de los sprints.

3.3.2 Formato utilizado para la planificación de sprints

Como es un equipo que la velocidad es una variable muy inestable, ya que no hay proyectos anteriores en los cuales basarse, elegimos usar el proceso Commitment Driven para el realizar el plan, y no Velocity Driven.

3.3.3 Etapas del proceso de planning del sprint



3.3.2 Largos de sprints

El largo de los sprints los definimos fijos y son de **XXXXX**

3.3.3 Tareas desglosadas de historias de usuario

Las tareas desglosadas las realizamos durante cada iteración del plan

3.3.4 Burndown charts

Para cada sprint realizamos un burndown chart para monitorear la iteración. A medida que se van desarrollando, las vamos actualizando al principio de cada dia.

3.3.4 Resultados de sprints

3.3.4.1 Sprint 1

Fecha Comienzo:
Fecha Fin:
Historias de Usuario:
Tareas:
Burndown Chart:

3.3.4.2 Sprint 2

Fecha Comienzo:
Fecha Fin:
Historias de Usuario:
Tareas:
Burndown Chart:

3.4 Gestión de riesgos

3.4.1 Planificación de riesgos

La Gestión de los riesgos se realizará basándose en el documento Taxonomy-Based Risk Identification del SEI (Software Engineering Institute). Este documento plantea una categorización de riesgos, en la cual tenemos tres primeras principales clases (Ingeniería del producto, Ambiente del desarrollo y Restricciones del producto) y luego un desglose de cada clase. Está metodología para realizar la gestión de riesgos está pensada para metodologías tradicionales. Aun así, decidimos adaptarla para utilizarla en una metodología ágil ya que creemos que es una forma eficiente y correcta de manejar los riesgos. Esta categorización es la utilizada para generar las siguientes tablas:

3.4.1.1. Riesgos en Ingeniería del producto

Categoría	Riesgo	Impacto (I)	Prob. (P)	Magnitud = I x PO	Plan de Respuesta	Plan de Contingencia
Requerimientos	Errores en definiciones de historias	3	0.2	0.6	Realizar una investigación para determinar mejor la historia problemática	Revisar todas las historias nuevamente
Diseño	Modificaciones inesperadas debido a factores externos	3	0.2	0.6	Controlar el ambiente del producto para que no surjan factores externos influyentes	Realizar un nuevo diseño
Codificación	Tardanza en el desarrollo	3	0.2	0.6	Comparar estado actual del proyecto frente a planificaciones y velocidad del equipo	Ajustar tiempos para corregir las tardanzas

	Incumplimiento de estandares de codificacion	2	0.2	0.4	Instruir a los desarrolladores con los estándares	Realizar refactor del código teniendo en cuenta los estándares
Especialidades	Problemas de seguridad	5	0.4	2	Revisar detalles en los cuales deba haber seguridad total	Deshabilitar el producto y revisarlo por completo
	Poca mantenibilidad	2	0.2	0.4	Cumplir con estándares	Realizar refactor del sistema teniendo en cuenta los estándares de mantenibilidad

3.4.1.2. Riesgos del Ambiente de desarrollo

Categoría	Riesgo	Impacto (I)	Prob. (P)	Magnitud = I x PO	Plan de Respuesta	Plan de Contingencia
Herramientas	Herramientas obsoletas	4	0.4	1.6	Informarse sobre actualizaciones	Realizar una actualización de todas las herramientas utilizadas
	Desconocimiento de las herramientas o tecnologías a utilizar	4	0.4	1.6	Definir el problema a implementar basándose en las herramientas y tecnologías que el equipo conoce	Realizar una capacitación técnica (spike) sobre las nuevas herramientas y tecnologías
Ambiente de trabajo	Mala comunicación en el equipo	4	0.2	0.8	Seguimiento de las comunicaciones	Implementar nuevos métodos de comunicación

	Poca cooperación de algunos integrantes del equipo	3	0.2	0.6	Supervisar las actitudes de las personas	Motivar a los integrantes del equipo de distintas maneras

3.4.1.3. Riesgos en Restricciones del proyecto

Categoría	Riesgo	Impacto (I)	Prob. (P)	Magnitud = I x PO	Plan de Respuesta	Plan de Contingencia
Contratos	Eventuales pedidos de aumento de sueldo	2	0.4	0.8	Mantener a los integrantes motivados y conformes con su sueldo	Si hay recursos monetarios, dar los aumentos necesarios. De lo contrario, intentar motivar a las personas para mantener el sueldo actual.
Interfaces	Demora en la entrega del Hardware comprado	3	0.6	1.8	Mantener una comunicación fluida con el proveedor	Solicitar la prisa del proveedor
	Hardware comprado incorrectamente	5	0.4	2.0	Informarse sobre el hardware necesario	Conseguir recursos para comprar nuevo hardware

Notas: 1) La columna plan de respuesta se refiere a acciones para que el riesgo no se convierta en problema. 2) La columna plan de contingencia se refiere a acciones en caso que el riesgo se haya convertido en problema.	**Impacto**: 0 - Ninguno. 1 - Marginal. 2 - Poco importante. 3 - Importante (puede retrasar el proyecto). 4 - Crítica (pude detener el proyecto). 5 - Catastrófica (fracaso del proyecto).	**Probabilidad de ocurrencia**: 0.0 - no probable. 0.2 - poco probable. 0.4 - probable. 0.6 - muy probable. 0.8 - altamente probable. 1.0 - se convierte en problema.

3.4.2 Seguimiento de Errores

En está sección determinamos una metodología a seguir para el reporte de errores y el seguimiento de los mismos. Para realizar el “Bug Tracking” utilizaremos la herramienta Trello configurada para realizar esta tarea.

3.4.2.1 Configuracion de tablero

La configuración a utilizar será la definicion de un tablero con las “boards” siguientes:

Reported by Team: En está “board” se colocaran los bugs que fueron encontrados y reportados por miembros del equipo.

Reported by Users: En está “board” se colocaran los bugs que fueron encontrados y reportados por usuarios del producto.

Accepted: Una vez que los bugs fueron verificados y realmente son un bug, se pasan a está “board” para que los miembros del equipo encargados de los bugs sepan que pueden corregirlo.

In Progress: Una vez que un desarrollador elige corregir un bug, se pasa la tarjeta del bug a está “board”. Aquí se puede ver que desarrollador está trabajando en el bug.

To be Validated: Una vez que el desarrollador determina que el bug está corregido, se pasa la tarjeta del bug a está “board”. Una vez que las tarjetas están en está “board” los encargados del área de calidad del equipo pueden validar que el bug fue corregido correctamente. De validarse, se pasan a la “board” llamada “Done”. En caso de no ser validados, se especifica en la descripción de la tajera, el porque no fue validado y vuelve a la “board” llamada “Accepted” para que los desarrolladores puedan encargarse de solucionarlo.

Done: Aquí se colocan las tarjetas de los bugs solucionados.

3.4.2.2 Manejo del tablero

Se definirá un miembro del equipo que se encargará de manejar el tablero.

3.4.2.2.1 Acciones

Las acciones que se le pueden pedir a este miembro son las siguientes:

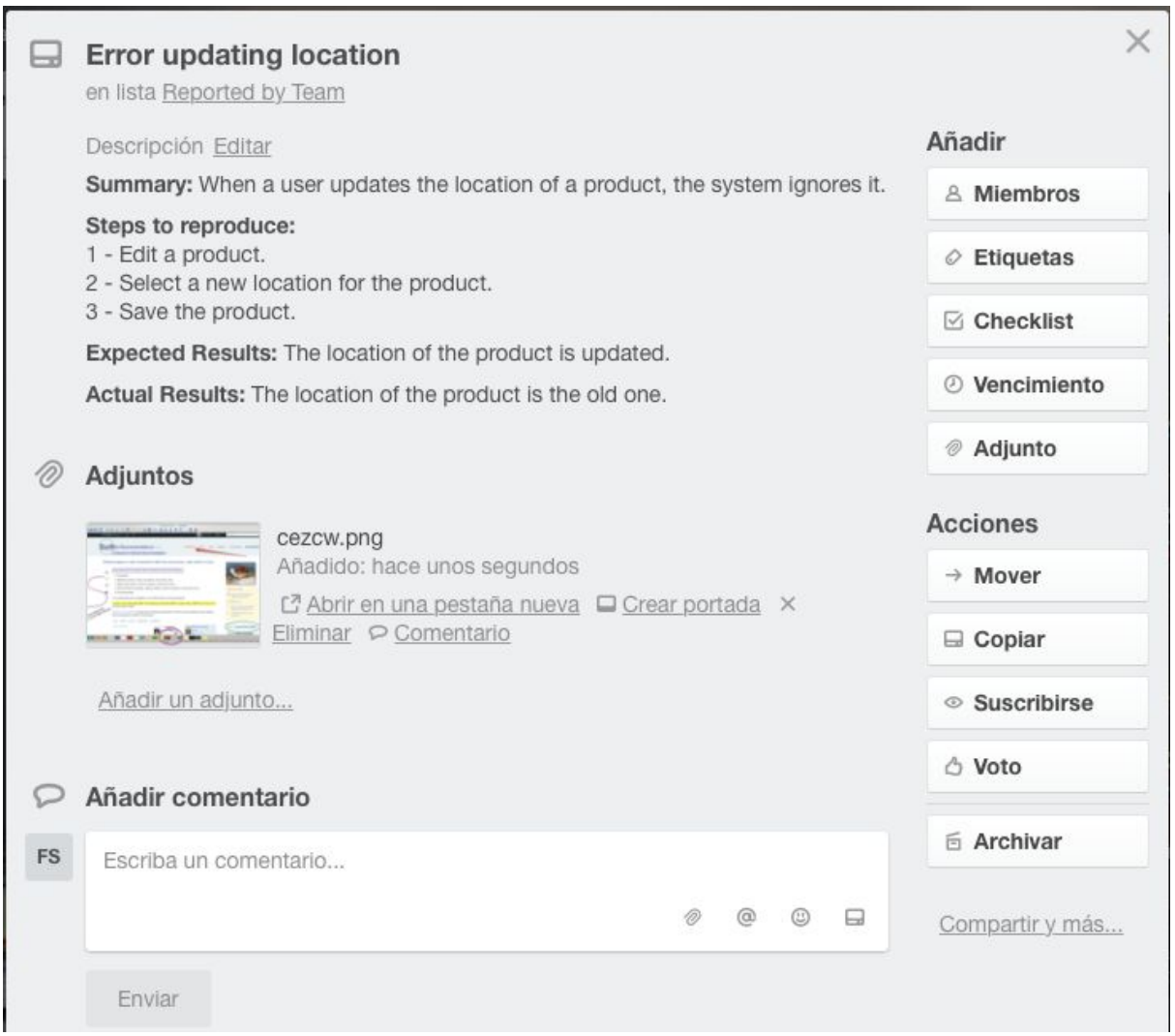
Reportar error: Se le envía un correo a este miembro para reportar un error. Dentro del correo se define una estructura para el reporte. El encargado del tablero luego define la tarjeta del error en el tablero de Trello. Este correo lo pueden enviar miembros del equipo como también usuarios del sistema.

Estructura de reporte de error:

Summary: <descripción breve del error>
Steps to reproduce: <pasos a seguir para reproducir el error>
Expected results: <resultados esperados>
Actual results: <resultados obtenidos>

Archivos adjuntos al correo: Imagenes de capturas de pantalla del error con flechas o referencias que indiquen el error.

Teniendo está información el encargado del tablero definirá una tarjeta en trello (tener en cuenta que trello utiliza el lenguaje “Markup” para escribir descripciones) de la siguiente manera:



Aceptar error: Luego de que los miembros del área de calidad acepten uno de los errores reportados, se le envía un correo al encargado del tablero con la siguiente estructura.

Bug: <nombre de la tarjeta de trello> fue aceptado por área de calidad.

Corregir error: Cuando un desarrollador elige un bug para corregir, se le envía un correo al encargado del tablero con el siguiente formato:

Bug: <nombre de la tarjeta de trello> va a ser corregido por <nombre del desarrollador a corregir el bug>.

Cuando el encargado del tablero recibe este correo, mueve la tarjeta hacia la “board” llamada “In Progress” y a su vez asigna el desarrollador correspondiente a la tarjeta.

Finalizar error: Cuando el desarrollador finaliza su tarea de corregir un bug, se le envía un correo al encargado del tablero con el siguiente formato:

Bug: <nombre de la tarjeta de trello> fue finalizado.

Cuando el encargado recibe este correo mueve la tarjeta a la “board” llamada “To be validated”

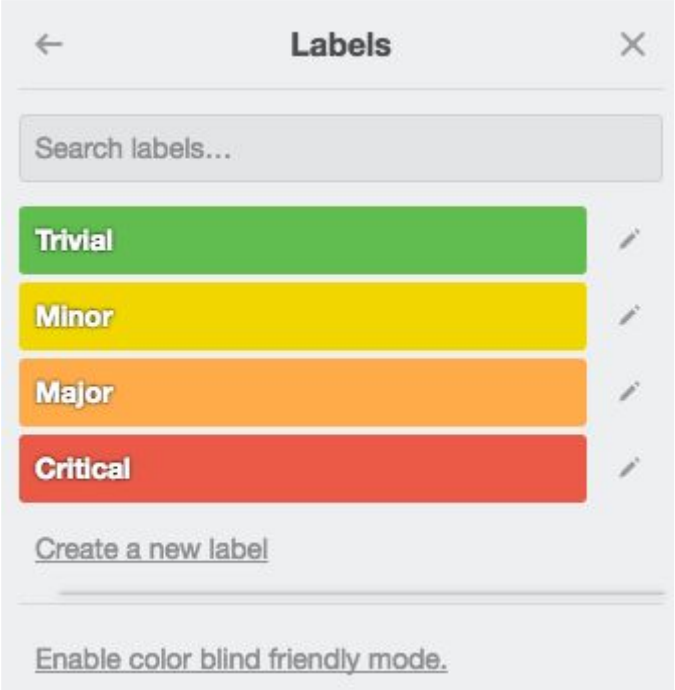
Validar error: Cuando el equipo de calidad valida un error se le envía un correo al encargado del tablero con el siguiente formato:

Bug: <nombre de la tarjeta de trello> fue validado.
o
Bug: <nombre de la tarjeta de trello> no fue validado por los motivos: <texto describiendo los motivos por los cuales el error no fue validado>.

Cuando el encargado recibe este correo, si el error fue validado, mueve la tarjeta a la “board” llamada “Done”. Si el error no fue validado, se agregan los motivos a la descripción de la tarjeta y se mueve la tarjeta a la “board” llamada “Accepted”.

3.4.2.2.2 Etiquetas

Para manejar las prioridades de los bugs se define una lista de etiquetas para las tarjetas. Estas etiquetas son las siguientes:



El encargado del tablero, será el encargado de priorizar los bugs, esto lo realizará mediante las etiquetas que luego las asignará a cada uno de los bugs.
Estas etiquetas servirán para que luego los desarrolladores eligan los bugs a corregir de una forma priorizada.

3.4.2.3 Comentarios

Cuando hablamos del encargado del tablero, nos referimos a un miembro del equipo asignado para manejar el tablero de trello. Este miembro debe tener conocimientos de trello y del manejo de está herramienta.

Cuando hablamos de equipo de calidad o área de calidad nos referimos a uno o varios miembros del equipo designados para verificar la calidad del producto. Estos miembros deben tener claro el proceso de Aseguramiento de la calidad.

4. Gestión de la configuración

4.1 Introducción

En esta sección detallaremos el proceso y herramientas utilizadas para la gestión de configuración de nuestro proyecto.

4.2 Control de cambios y de versiones

Para el control de cambios y de versiones utilizamos la tecnología GIT. Creamos un equipo donde pertenecemos los 3 integrantes de este trabajo y creamos un repositorio para este proyecto.

Además, utilizamos la metodología GIT Flow para el trabajo con GIT branches, commits, etc. De este modo los integrantes del equipo tenemos un mayor entendimiento de la gestión de la configuración.

4.3 Registros de defectos

Utilización y evidencia de uso de una herramienta de registro de defectos.

5. Aseguramiento de calidad

5.1 Proceso y producto

Una de las tareas claves para el aseguramiento de la calidad del código producido será el uso de convenciones de desarrollo seleccionadas en conjunto por el equipo. Para esto nos basamos en el uso de las prácticas de Clean Code.

Evidencia Clean Code y justificación de codificación

En esta sección se harán evidencias y restricciones que tomamos en nuestro equipo de desarrollo para comprender, manejar y mantener mejor el código, utilizando prácticas recomendadas por clean code.

Reglas utilizadas

Nombres

a. Nombre de variables = empieza minúscula
b. Nombre de properties = empieza Mayúscula
c. Nombre de métodos = empieza Mayúscula
d. Nombre de clases = empieza Mayúscula
e. Nombre de constantes = todo mayúscula
f. Métodos y variables booleanas empiezan con is, are, has o exists
g. Variables o propiedades que auxilian a otra propiedad: *“<PropiedadAuxiliada>Aux”*

Estructura del código de una clase

a. Indentación y espaciado utilizada recomendada por VisualStudio al indentar la clase.
b. Un espacio vertical entre métodos
c. Estructura de una clase:
    1. Declaración de enums
    2. Declaración de constantes
    3. Declaración de propiedades sin comportamientos agregados en get y set
    4. Declaración de propiedades con comportamientos agregados en get y set, con su variable o propiedad que la auxilia arriba.
    5. Declaración de variables (Excepto las utilizadas como auxiliares de las propiedades, estas van arriba de la propertie que auxilian, mirar el ejemplo)
    6. Espacio en blanco
    7. Constructores
    8. Métodos estáticos que llaman a los constructores(*“CreateWithNameComission(...)”*)
    9. Métodos

Ejemplo:

```
public class ExampleClass
{
    public enum EnumeratorExample : int
    {
        OptionA = 1,
        OptionB = 2
    }
    public const string CONSTANT = "This is a constant value.";
    public int FirstProperty { get; set; }
    private int SecondPropertyAux;
    public int SecondPropertyAux
    {
        get
        {
            return SecondPropertyAux
        }
        set
        {
            if (value > 5)
            {
                SecondPropertyAux = value;
            }
            else
            {
                SecondPropertyAux = 0;
            }
        }
    }

    public ExampleClass(){}
}
```

Estructura del código de una interfaz

a. Indentación y espaciado utilizada recomendada por VisualStudio al indentar la clase.
b. Sin espacios entre declaración.

Ejemplos de aplicación de Clean Code

Nombres de variables

```
int comission;
double earnings = 0;
IPaymentService paymentService = new PaymentService();
```

Nombres de properties

```
public int ClientId { get; set; }
public string Address { get; set; }
public int IdentityDocument { get; set; }
public string Name { get; set; }
public string Phone { get; set; }
public int Points { get; set; }
```

Nombres de clases

```
public class Client
```

Nombres de constantes

```
public const int MIN_DOCUMENT_ID = 10000000;
public const int MAX_DOCUMENT_ID = 99999999;
```

Comentarios

Se utilizaron las reglas anteriores y se intenta seguir el planteamiento del libro “Clean Code”. Quizás por cuestiones de tiempo, no se invierte demasiado en este apartado, aunque se intenta programar utilizando este planteo.
En el caso de que se tuviera más tiempo, una instancia fuertemente viable, sería un refactor del código, realizando ajustes de Clean Code, para dejar un codigo fuente más limpio y entendible para futuros programadores sobre esta aplicación.

5.1.1. Cualidades deseadas del software

Las cualidades que las actividades de aseguramiento de calidad deben cumplir son las especificadas en el ISO 9126. Estas cualidades son: Funcionalidad, Confiabilidad, Eficiencia, Usabilidad, Mantenibilidad y Portabilidad.

5.1.2. Actividades de Gestión de Calidad durante el proyecto

Se realizarán revisiones para las actividades de aseguramiento de la calidad. Estas revisiones se realizarán en los momentos definidos en este plan de calidad. Las revisiones a realizar serán de las siguientes metodologías:

Ad-Hoc: Se realizarán de forma informal. No es necesario realizar un registro de estas revisiones. Para todos los entregables.

Revisión de pares: Se realizarán de forma informal. Se realizará un pequeño registro de quienes participaron de la revisión y de los defectos encontrados. Para todos los entregables.

Revisión de pares múltiples: Idem Revisión de pares. Para todos los entregables.

Walkthrough: Se realizará formalmente. El líder de SQA debe organizar la reunión. Se documentará la presentación y los defectos encontrados por parte de los presentes. Para todos los entregables.

Inspección: Se realizarán Inspecciones formales en los siguientes artefactos: Diseño arquitectónico y Código fuente del software. Estas inspecciones serán documentadas dejando registrado los participantes, los defectos, las posibles soluciones y la solución elegida. También se debe realizar una validación y documentarla.

5.1.3. Roles y responsabilidades

Rol	Efectuado por	Responsabilidad
Líder de SQA	Persona 1: Gerente de proyectos	-Planificar actividades de SQA -Participar como moderador en inspecciones y revisiones
Participe de SQA	Persona 2: Arquitecto	-Realizar los ajustes necesarios y/ó rehacer el trabajo para cumplir con el producto definido.

5.1.4. Artefactos sobre los cuales se efectuarán las actividades

1 - Diseño arquitectónico
2 - Código fuente del software
3 - Casos de prueba

5.1.5. Momentos en lo que se van a llevar a cabo las actividades

El control de calidad no debe realizarse al final de cada iteración, sino en **todos** los aspectos de la producción. El aseguramiento de la calidad forma parte del proceso de desarrollo. Por lo tanto, se establecerán momentos para realizar las actividades de aseguramiento de calidad.

Las revisiones de Ad-Hoc, Revisión de a pares y Revisión de a pares múltiple se realizarán de forma diaria.

Las revisiones de Walkthrough se realizarán cuando existan versiones medianamente completas y completas de cada uno de los entregables.

Las inspecciones se realizarán cuando estén las versiones finales de cada entregable.

5.1.6. Estándares

Artefacto	Estándar
Diseño arquitectónico	Template de EDS.doc
Código fuente del software	Google Java Style
Casos de prueba	IEEE 830

5.2 Plan de Métricas

Para desarrollar el plan de métricas utilizamos la metodología GQM (Goal Question Metrics).La siguiente tabla explica esta metodología fijando objetivos de la medición (comprender, controlar, predecir o mejorar). Luego hacerse preguntas (para alcanzar los objetivos) y por último definiendo métricas que contestan las preguntas.

Este plan de métricas será utilizado durante el desarrollo del backend del sistema. No se seguirá el mismo durante el desarrollo de la aplicación mobile.

Objetivo	Pregunta	Métrica
Mejorar y disminuir la cantidad de defectos.	¿Se puede reusar código?	Código duplicado.
Asegurar la correctitud del programa.	¿Cuál es el índice de cobertura del código?	Porcentaje de cobertura de código.
Controlar la complejidad del código.	¿Qué tan complejo es el código?	Complejidad ciclomática.
Mejorar el entendimiento y la comprensión del código.	¿Qué es lo que hace el código?	Pruebas unitarias.

Nombre	Código duplicado		
Info que brinda.	Cantidad de codigo duplicado.	**Para quién es de utilidad**	Equipo
Entidad	Codigo	**Escala**	Discreta
Unidad	Métodos	**Tolerancia**	Métodos fuera del dominio
Formula	Cantidad de duplicados del proyecto = Σ cantidad de métodos por clase.	**Atributos a recolectar**	1. Nombre de clase involucrada 2. Métodos duplicados.

	Metodo	**Responsable**	**Frecuencia**	**Registro**
Recolección	Correr Source Code Metrics	Desarrollador	Semanalmente	
Analisis	Comparación con tolerancia	Gerente del Proyecto	Mensualmente	Plan de Proyecto.

Notas	Debe verificarse cada vez que se crea un método que no sea código duplicado. Dado que esto es complicado se correrá esta métrica semanalmente para asegurarse que no se introdujo código duplicado.

Nombre	Cobertura de Pruebas		
Info que brinda.	Porcentaje de cobertura de pruebas.	**Para quién es de utilidad**	Equipo
Entidad	Código y sus pruebas.	**Escala**	Porcentual
Unidad	Porcentaje	**Tolerancia**	Arriba del 80% del código.
Formula	Interna de la herramienta utilizada.	**Atributos a recolectar**	1. Porcentaje de cobertura total, por clase y por métodos. 2. Líneas de código visitadas. 3. Cantidad de caminos de los métodos recorridos.

	Metodo	**Responsable**	**Frecuencia**	**Registro**
Recolección	Correr herramienta de medición.	Desarrollador	Semanalmente	
Analisis	Comparación con tolerancia	Gerente del Proyecto	Mensualmente	Plan de Proyecto.

Notas	Esta medición de métrica es imprescindible para el correcto desarrollo del software ya que asegura que el proyecto está yendo por buen camino.

Nombre	Complejidad Ciclomática		
Info que brinda.	Promedio de caminos por método	**Para quién es de utilidad**	Equipo
Entidad	Código	**Escala**	Continua
Unidad	-	**Tolerancia**	Debajo de 2
Formula	Complejidad ciclomática del proyecto = (Σ cantidad de caminos por método)/cantidad de métodos	**Atributos a recolectar**	1. Se recolecta un indicador.

	Metodo	**Responsable**	**Frecuencia**	**Registro**
Recolección	Correr Simple Code Metrics.	Desarrollador	Semanalmente	
Analisis	Comparación con tolerancia	Gerente del Proyecto	Mensualmente	Plan de Proyecto.

Notas	Esta métrica es importante ya que tiene una relación directa con la mantenibilidad del código.

5.3 Pruebas de Software

Para las pruebas de software del backend se realizarán pruebas unitarias y pruebas funcionales.

Para la aplicación mobile se realizará testing de funcionalidades únicamente. Para este proceso se utilizará una aplicación Android que lleva, a medida que se utiliza la aplicación, un registro de las fallas que esta presenta. Dado que la aplicación tiene una licencia de 15 días, se realizará el testing de la aplicación dentro de esa ventana de tiempo. Se realizará una instancia de Spike, para la investigación de otras herramientas que permitan realizar lo mismo.

7. Arquitectura

7.1 Modelo de arquitectura utilizado

Se utilizara el modelo de capa Cliente - Servidor. Siendo el servidor encargado del Backend y el cliente el frontend.

7.2 Backend

Se realizará una API mediante el uso de ASP.Net. Se utilizara el lenguaje de programación C# en el IDE Visual Studio. Esta fue una restricción presentada por el cliente.

7.2.2 Diagrama Conceptual

7.3 Frontend

Se realizará una aplicación mobile para el sistema operativo android, usando el IDE Android Studio.

7.3.1 Navegabilidad

Mockups

8. Bibliografía

9. Anexos