**CONTROL DE VERSIONES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Realizado Por** | **Fecha** | **Motivo** |
| **1.0** | Líder Desarrollo | 27/03/2014 | Versión Inicial Estrategia |

# ESTRATEGIA:

**Ciclos y funciones:**

El desarrollo del producto se realizará en un proceso cíclico TSP donde el número de ciclos decidido por el instructor es 2.

En el primer ciclo se debe crear un diseño, implementación y pruebas de funciones básicas de nuestro producto. Para realizar las pruebas se creará una clase Main que nos permita ejecutar el programa y comprobar los resultados. En el segundo ciclo se crearán más funcionalidades al producto con una interfaz definida y amigable al usuario. Las funcionalidades definidas a desarrollar por cada ciclo se encuentran en la forma STRAT1.

El producto producido en cada uno de los ciclos debe ser construido de una forma que permita integrar nuevas funcionalidades en los ciclos siguientes, para ellos se debe analizar y dividir el desarrollo en funciones que cumplan objetivos específicos.

Las prioridades en el desarrollo serán las comparaciones del programa modificado, para ello se debe analizar el uso de librerías existentes que permitan facilitar el trabajo de los desarrolladores.

**Arquitectura:**

El desarrollo de la aplicación se debe basar en capas que permitan en algún momento dado realizar cambios sin afectar la totalidad del producto. Las capas consideradas en las que se dividirá el producto serán:

* **Commons:** Contendrá las clases utilitarias propias y específicas de la aplicación que no se consideren reutilizables y no deban estar en librería.
* **Business:** Contiene las clases de lógica de negocio del producto.
* **Presentation:** Contiene clases relacionadas con la interfaz a mostrar al usuario final.

Al utilizar capas los integrantes del equipo podrán dividir desarrollos y trabajar en forma paralela sin verse afectados o en conflicto

**Reusabilidad:**

Alcrear funciones que puedan ser reusadas en siguientes ciclos se deben implementar en un paquete externo considerado como una librería de utilidades para el equipo de trabajo. Un ejemplo de las utilidades es crear métodos para manejo de archivos que puede ser utilizado en diferentes clases o ciclos.

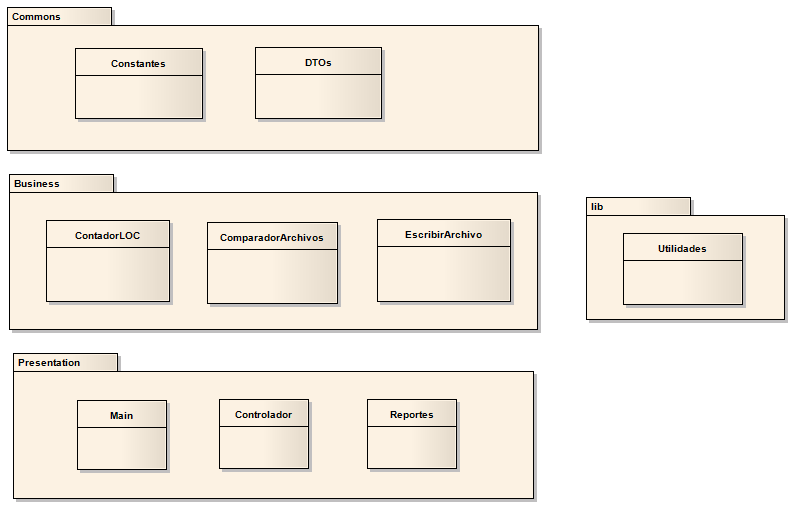
**Pruebas:**

Antes de realizar las pruebas integrales del sistema, cada integrante del equipo que realice cambios en el código debe realizar pruebas unitarias de los métodos implementados utilizando assert, esto con el fin de corregir errores en una etapa temprana de desarrollo y evitar retrasos en la entrega del producto.

**Documentación:**

Cada acción sobre el desarrollo del producto debe ser comentado en el código y debe tener un comentario claro al realizar commit de sus cambios para que los demás integrantes puedan entender con facilidad el código fuente del producto y realizar las mejoras eficientemente en posteriores ciclos.

# DIAGRAMA CONCEPTUAL:



El diseño conceptual representa las capas a las que hace referencia la arquitectura de la estrategia: commons, business y presentation.

En Commons se considera necesario crear constantes donde se puedan identificar etiquetas y demás palabras reservadas de la aplicación, y DTOs para clases que devolverán las funciones entre paquetes.

En Business para la lógica del negocio se considera necesario crear Clases que permitan contar LOC, Comparar los archivos para observar las líneas agregadas y eliminadas y clases que permitan escribir en los archivos las etiquetas o encabezados necesarios prueba.

En Presentation se considera necesario crear clases GUI realizadas en Swing, clases para controlar eventos en la interfaz y clases para realizar reportes solicitados.

Las otras clases que permitan ser reutilizadas en otros ciclos se almacenaran en un paquete externo (libreria).

# Estimación de Tiempos por Ciclo:

Para la estimación de tiempo y tamaño en LOC de los desarrollos por cada ciclo se utilizara el menor LOC/HOUR (12 LOC/HOUR), asociado a los cursos de PSP de los integrantes del equipo.