# Administrador StakeHolders

**Objetivo del proyecto**

Diseñar y desarrollar una herramienta que permita registrar y fomentar la interacción de los stakeholders con un proyecto y sus componentes. La misma debe facilitar la integración de otras aplicaciones para realizar análisis de redes. Además debe ser adaptable para permitir una posterior expansión e integración de nuevos módulos.

## Funcionalidad de la herramienta y características de diseño

La herramienta será una aplicación web (cliente-servidor), que facilitará el acceso a diferentes usuarios a través del explorador para acceder a sus cuentas individuales.

Cada cuenta permitirá diferentes vistas según el usuario que la utilice. Por un lado una vista completa de la red y sus componentes para el administrador y a su vez una vista limitada para stakeholders y/u otro tipo de usuarios. Las vistas se basarán en la teoría en el capítulo 9 del libro “Building the Architecture Documentation”, y se adaptarán a las necesidades del proyecto a lo largo del ciclo de desarrollo.

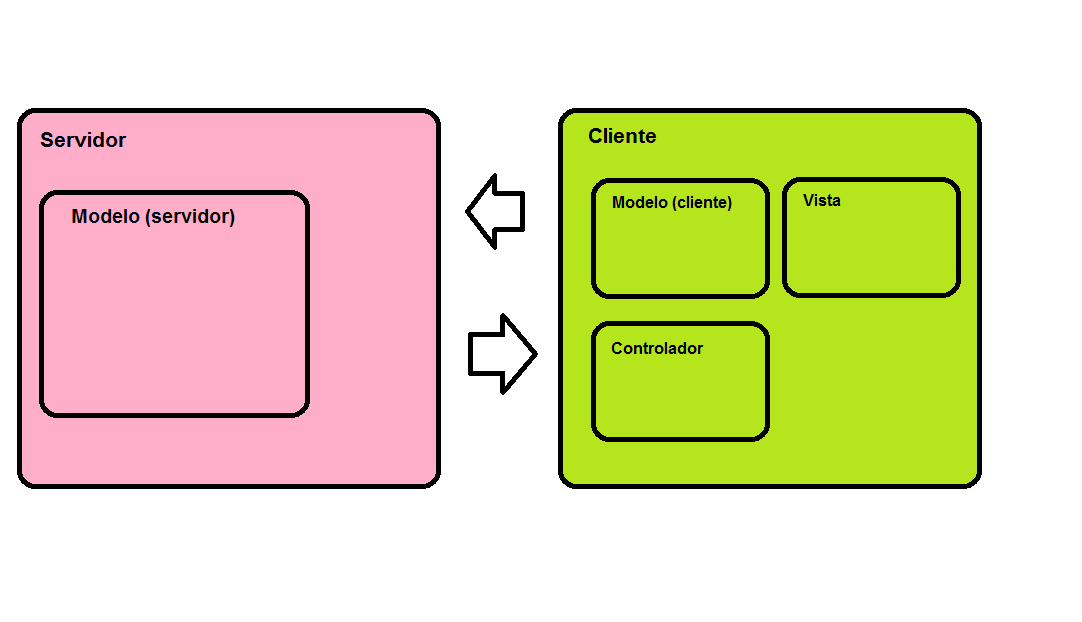
## Arquitectura del sistema ( MVC )

Modelo: contiene todas las estructuras que representan el programa. Las clases que contienen el modelo son las sig:

Vista: contiene todas las clases que permiten representar en pantalla el programa. Las clases son las sig:

Controlador: contiene todas las clases (Handlers) que se encargan de manejar los eventos en pantalla. Las clases son las sig:

## D:\disenioEsparragos\espacioDeTrabajo\informes\documentacion\Arquitectura_incial1.pngArquitectura GWT (cliente servidor)



El componente Modelo(servidor) se comunica con el componente Modelo(cliente).

Serán los encargados de controlar la estructura del programa.

## Modelo(Servidor)

## Estrategia comportamiento de la cuenta de usuario

La cuenta de usuario posee una estrategia de comportamiento que es la encargada de crear la sesión y la pantalla incial(según el tipo de usuario) que vera el usuario con sus distintas opciones ya sea usuario comun o administrador.

La pantalla es la interfaz con la cual se comunica el usuario para utilizar los servicios brindados por la sesión correspondiente.

## Sesion

Esta clase es la encargada de definir los servicios que un usuario puede solicitar al programa.

Define métodos comunes, los cuales son compartidos entre un usuario común y un administrador y posee métodos abstractos que luegos serán definidos por las clases que hereden de esta.

Las clases SesionUser y SesionAdmin heredan de Sesion y se encargan cada una de definir el comportamiento del usuario común y del comportamiento del administrador respectivamente.

La clase SesionAdmin agrega la funcionalidad de validar nodos, asignar nodos e importar un grafo.

## Visitantes Grafo

 Utilizamos el patrón visitor, ya que  es una forma de separar un algoritmo de la estructura de un objeto, en nuestro caso el grafo.

Permite agregar operaciones a la estructura del grafo.

La idea básica es que se tiene un conjunto de clases elemento que conforman la estructura dl grafo. Cada una de estas clases elemento tiene un método aceptar (accept()) que recibe al objeto visitante (visitor) como argumento. El visitante es una interfaz que tiene un método visit diferente para cada clase elemento; por tanto habrá implementaciones de la interfaz visitor de la forma: visitorClase1, visitorClase2... visitorClaseN. El método accept de una clase elemento llama al método visit de su clase.

Las clases concretas de un visitante pueden entonces ser escritas para hacer una operación en particular.

Definimos 3 clases:

**VisitanteValidadorSimple**: Se encarga de validar todos los nodos y arcos cuyo estado es invalido. Basicamente es un ejemplo de cómo validar los cambios realizados en el grafo por los usuarios. Se pueden definir distintos validadores con diferentes criterios para validar.

**VisitanteFiltradoGrafo**: Utilizando un filtro de arcos y un filtro de nodos, esta clase se encarga de recorrer el grafo entero y solo devolver los nodos y arcos que cumplan dichos filtros.

**VisitanteExportarGML**: Clase encargada de recorrer el grafo y presentar en el formato gml la información que este contiene.

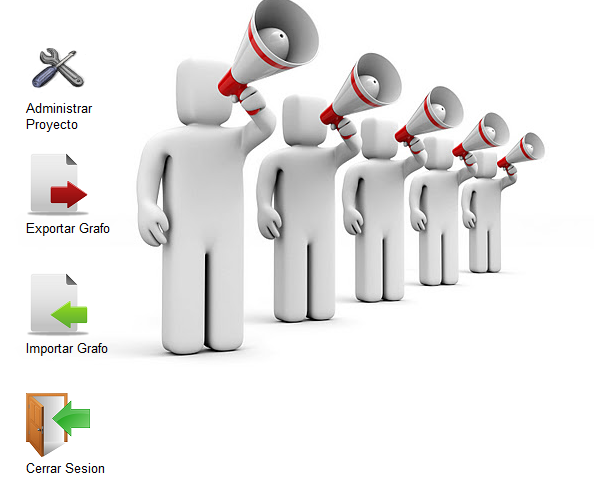
## Modelo(Cliente)

Posee una cuenta y un grafo que le entrega el servidor.

La métodos con los cuales invoca los servicios del servidor son los sig:

## Vista

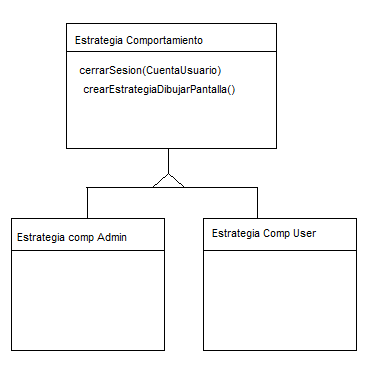
Esta conformada por una pantalla principal, que ofrece los servicios a los que puede acceder el usuario o administrador. Presenta una apariencia similar a la del escritorio del Sistema operativo Windows.



### Patrón strategy

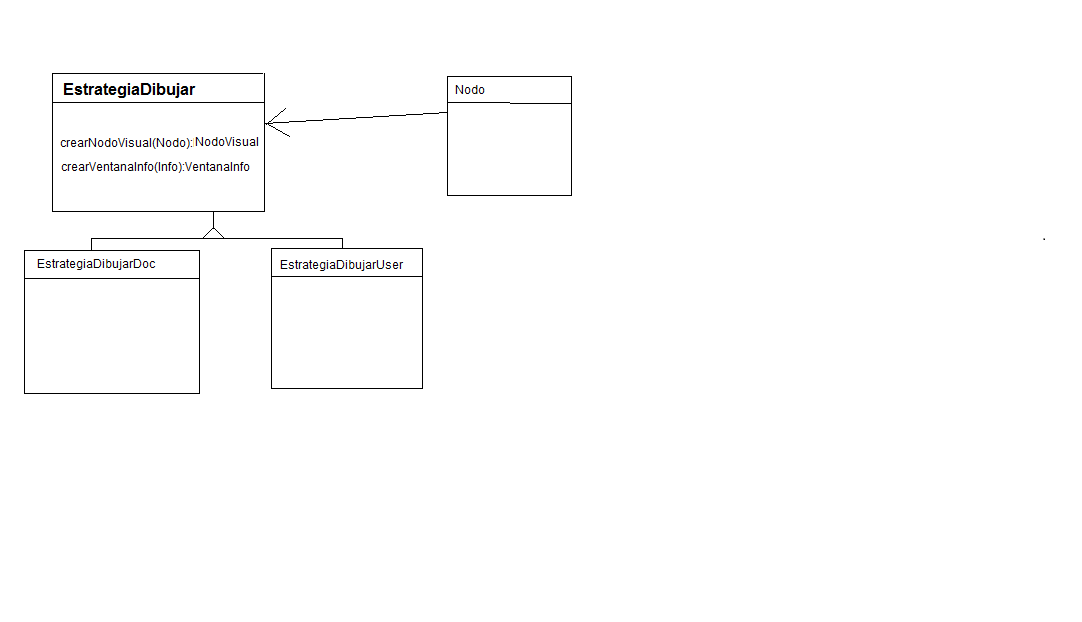
### Estrategia Comportamiento

Cada Cuenta de Usuario posee una Estrategia comportamiento que permite mostrar la pantalla correspondiente según sea un administrador o un usuario



### Estrategia Dibujar

Cada nodo posee la estrategia correcta para dibujarse en pantalla, ya sea un usuario o un documento.

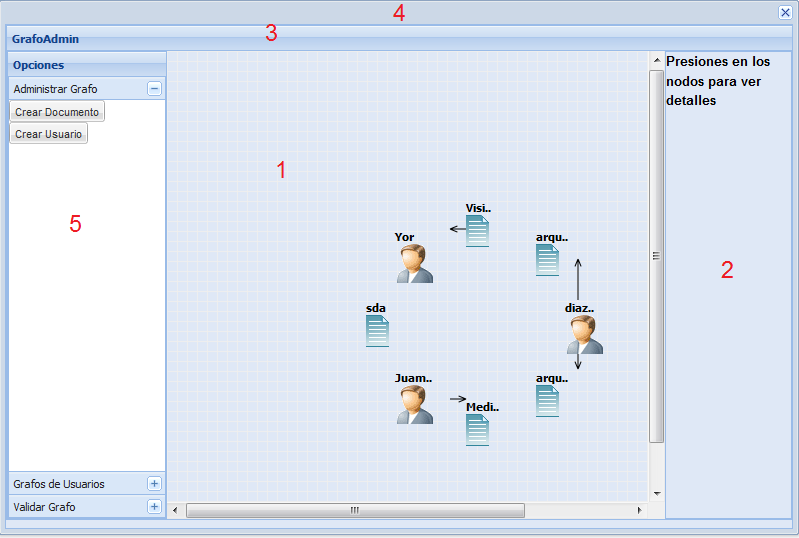
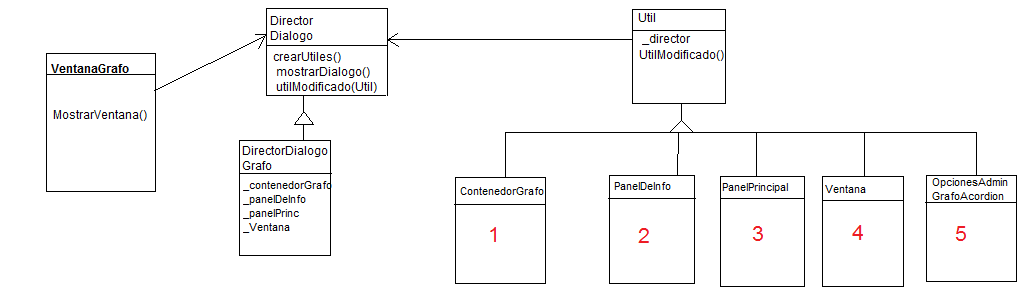


## Patrón Mediator

Ventana del grafo.

La clase DirectorDeDialogoGrafo se encarga de inicializar todos los componentes de la ventana, además de ubicarlos de manera adecuada.

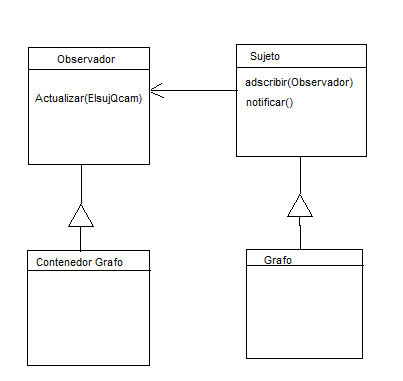
Cada componente es un Util, que mediante el método útilModificado(), le avisa al director que debe realizar cambios en la vista. Los métodos útilModificado son llamados por los Handlers del sistema, que se encargan de tomar los eventos en pantalla-> luego al ocurrir el evento adecuado-> llaman a UtilModificado, pasándole al director la responsabilidad de mostrar en pantalla los cambios.



## Observer Observable

La clase contenedorGrafo es la encargada de dibujar en pantalla el grafo.

Cuando ocurre un cambio en el grafo este llama a notificar() , y el contenedor grafo redibuja el grafo en pantalla.



## Controlador

Este componente contiene todos los handler de los componentes del sistema que deben responder a los eventos correspondientes.

“Acá van los handlers”

## Librería Links

<http://code.google.com/p/gwt-links/>

Permite interconectar widget en pantalla por medio de arcos.

Realizamos una extensión de la librería con el fin de controlar los diálogos de contexto de los arcos. ( al presionar el botón derecho sobre ellos).

