Universidad de Buenos Aires

Facultad de Ingeniería

75.45 - TALLER DE PROGRAMACIÓN II

EDITOR MODELO ENTIDAD-RELACIÓN

INFORME TÉCNICO

Profesora: Patricia

Fecha de Entrega:

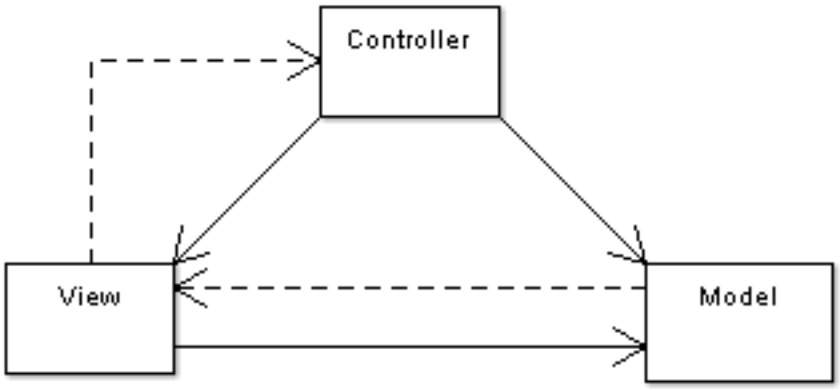
Alumnos:

Durán, Ximena	89771	xime.duran@hotmail.com
Muñoz Facorro, Juan Martín	84672	juan.facorro@gmail.com
Ordiales, Hernán	79106	hordiales@gmail.com

CONTENIDO

ARQUITECTURA GENERAL

El esquema general de la aplicación responde al patrón de diseño MVC (Modelo-Vista-Controlador).



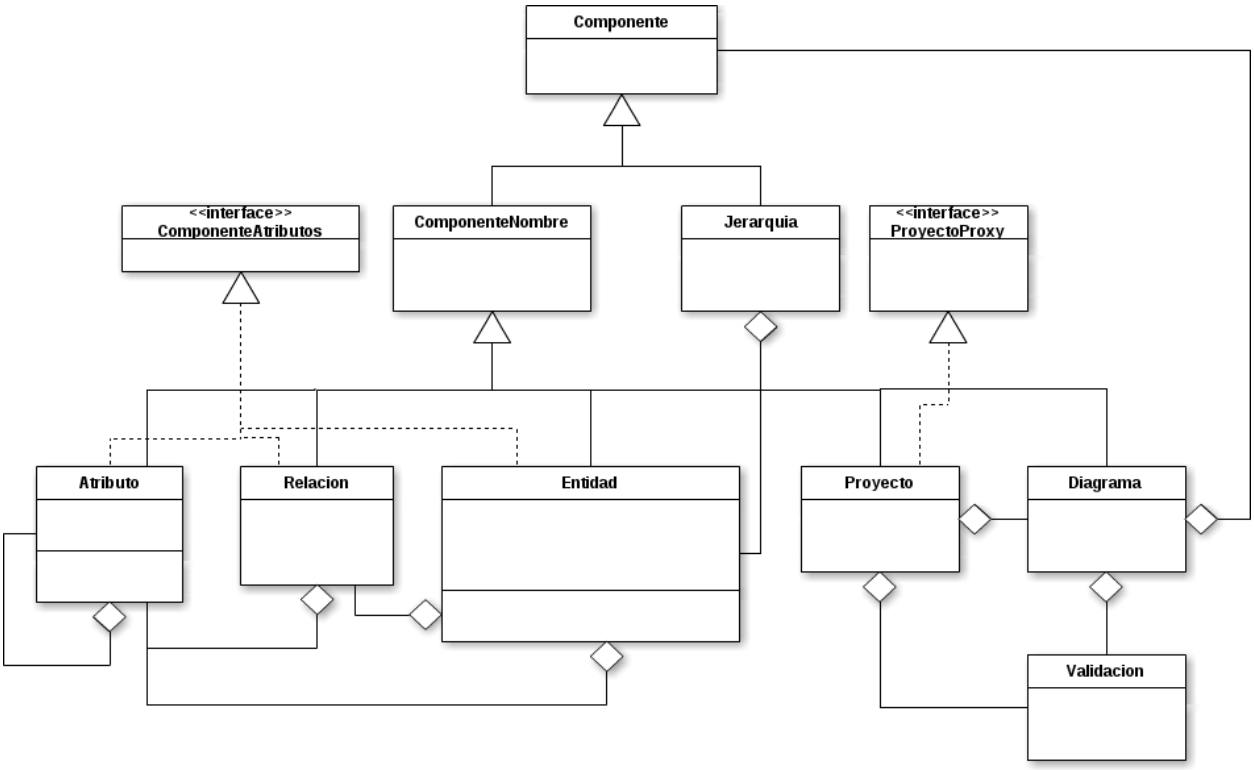
El mismo se eligió desde el comienzo, y casi sin evaluar alternativas, debido a que es el que se acostumbra a utilizar en este tipo de aplicaciones gráficas. El mismo separa de manera efectiva todo lo que es datos de la aplicación (modelo), interfaz de usuario (vista) y control (lógica de negocio).

El código también se organiza en diferentes paquetes de clases agrupados convenientemente, entre los que se destacan:

- **merditor.control**: Con todo lo referido a la lógica del negocio.
- **merditor.interfaz.swt.***: Vista gráfica en general y dividida en subpaquetes.
    - **Diálogos:** de configuración.
    - **Editores:** de entidades del modelo.
    - **Figuras:** de entidades del modelo.
    - **Listeners:** callbacks de los widgets.
- **merditor.modelo.***: Modelo de datos.
    - **Validacion:** implementación de métricas.
- **merditor.xml**: Manejo de serialización en archivos del tipo XML.

MODELO DE DATOS

Luego de identificar las principales entidades del sistema, se diseño un modelo de clases fuertemente influenciado por el patrón de diseño Composite. En varios aspectos era necesario poder tener diferentes tipos de componentes en un mismo diagrama o proyecto y que todos ellos respondan a una interfaz común para asi poder ser tratados de la misma forma. También se implementó una suerte de patrón proxy para controlar el acceso a la instancia de proyecto.



INTERFAZ

El diseño de la interfaz de usuario responde en su totalidad a los requerimientos del enunciado. Con un canvas principal para el diagrama actual, un árbol de proyecto a la izquierda y un menú de opciones. Más dialogos de edición y/o configuración de las diferentes entidades del sistema.

HERRAMIENTAS UTILIZADAS

Para el desarrollo del proyecto se utlizó el lenguaje de programación JAVA (versión 7) y se programó indistintamente bajo el sistema operativo Linux y Windows. Se optó por el mismo ya que era el que mejor se adaptaba en cuanto a los requerimientos de la aplicación y experiencia en su manejo de todos los integrantes del grupo.

También se utilizaron algunas librerias, complementarias a la estándard del lenguaje, que facilitaron el desarrollo de la parte gráfica. Sus características principales y justificación de su elección se especifican en la siguiente sección.

Como entorno de desarrollo se eligió el Eclipse (Indigo) por razones similares a la elección del lenguaje. Con el complemento de diferentes plugins, manejo de repositorio (svn), xml, etc, pero es para destacar el uso del plugin WindowBuilder para diseñar algunos diálogos de las interfaces en SWT.

LIBRERIAS

Debido a las características de la aplicación requerida, se focalizó bastante en la elección de la librería gráfica a utilizar.

Se evaluaron, por medio de la lectura de documentación y ejecución de ejemplos, las siguientes alternativas: Swing+JGraph, Qt (Jambi) y SWT+Draw2d. Finalmente se terminó optando por la última opción, principalmente, debido a que resultó la más simple de utilizar sin que eso este asociado una menor cantidad de recursos técnicos disponibles. Entre los factores a destacar para tal decisión se encuentran que SWT era conocida en parte por todos los integrantes del grupo, y que la librería Draw2d provee extensas facilidades para desarrollar aplicaciones como la requerida, es decir, editores de diagramas basados en un canvas. Facilitando por ejemplo la representación de figuras, conexiones, impresión, manejo de ventanas, etc.

Una vez elegido el camino de SWT+Draw2d, también se evaluó la posibilidad de utilizar el framework GEF. El cual facilita aún más el desarrollo de este tipo de aplicaciones y más complejas también. El uso del mismo se terminó descartando ya que su utilización era no trivial e implicaba aprender y adaptarse a desarrollar de una determinada forma, es decir, respetando su marco de trabajo. Se terminó considerando que no se justificaba para solo desarrollar una aplicación de estas características. Aunque también influyó en la decisión de prescindir de su uso definitivamente el hecho de que tras correr una serie de ejemplos y profundizar en su documentación, se encontró que no era posible desarrollar aplicaciones standalone con la misma, sino solo plugins para la plataforma Eclipse. Por lo que este camino requería desarrollar primero un plugin para Eclipse y luego exportarlo en forma de RCP (Rich Client Platform) junto a sus dependencias, pero consideramos que no se adaptaba al tipo de organización del trabajo que creiamos conveniente para cumplir con los requerimientos.

Para parsear los archivos XML se utilizó la librería del tipo DOM (Document Object Model) del paquete org.w3c que genera un árbol de objetos con sus dependencias en memoria.

Como complemento de SWT y Draw2d, también se utilizó la librería JFace para facilitar algunas cuestiones de interfaz como diálogos y manejo de widgets complejos.

VALIDACIÓN

Cómo metricas de validación se eligieron las siguientes:

- Validación de acoplamiento: Verificando si alguna relación tiene alguna otra entidad en otro diagrama o si alguna entidad pertenece a una relación en otro diagrama.
- Claridad de atributos: Controla que los componentes no superen una determinada cantidad máxima de atributos.
- Equilibrio de atributos: Advierte sobre componentes con una cantidad de atributos lejana al promedio.
- Equilibrio de componentes: Advierte sobre diagramas con una cantidad de componentes lejana al promedio.
- Equilibrio de relaciones: Advierte cuando una entidad pertence a una cantidad de relaciones lejana al promedio.
- Validar superposición: Controla la cantidad de diagramas en los que esta presente un mismo componente.