Documento de Diseño

Aplicación DomoLinx

Versión:

1.0 elaborada por el grupo Linces.

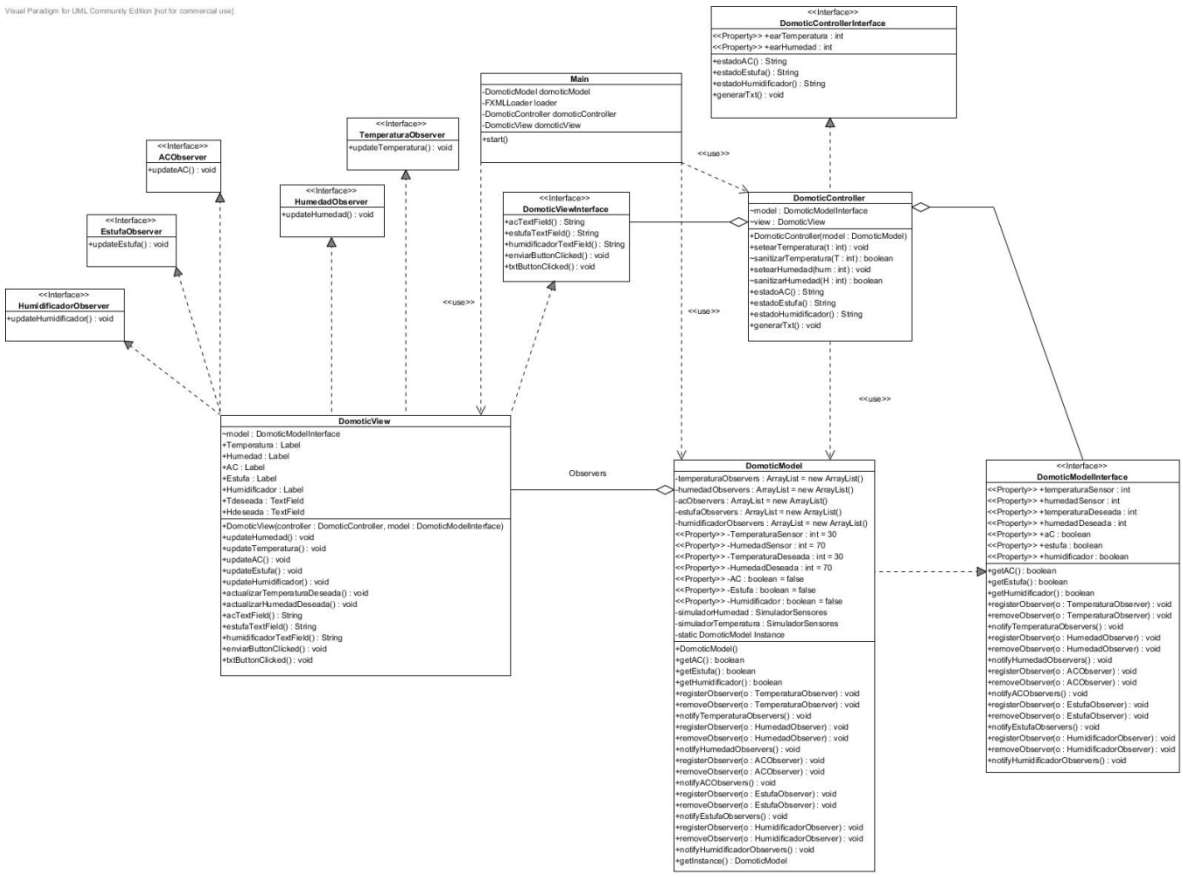
Fecha:

26/06/2017.

En el siguiente documento se busca detallar el diseño de la arquitectura que será utilizado en el desarrollo de la aplicación DomoLinx, los patrones de diseño empleados y la comunicación de las clases en los distintos casos de uso.

En el diseño de la aplicación DomoLinx se utilizará una arquitectura del tipo MVC, implementando patrones como el Singleton y el Observer. Referirse al documento de arquitectura para mayor información.

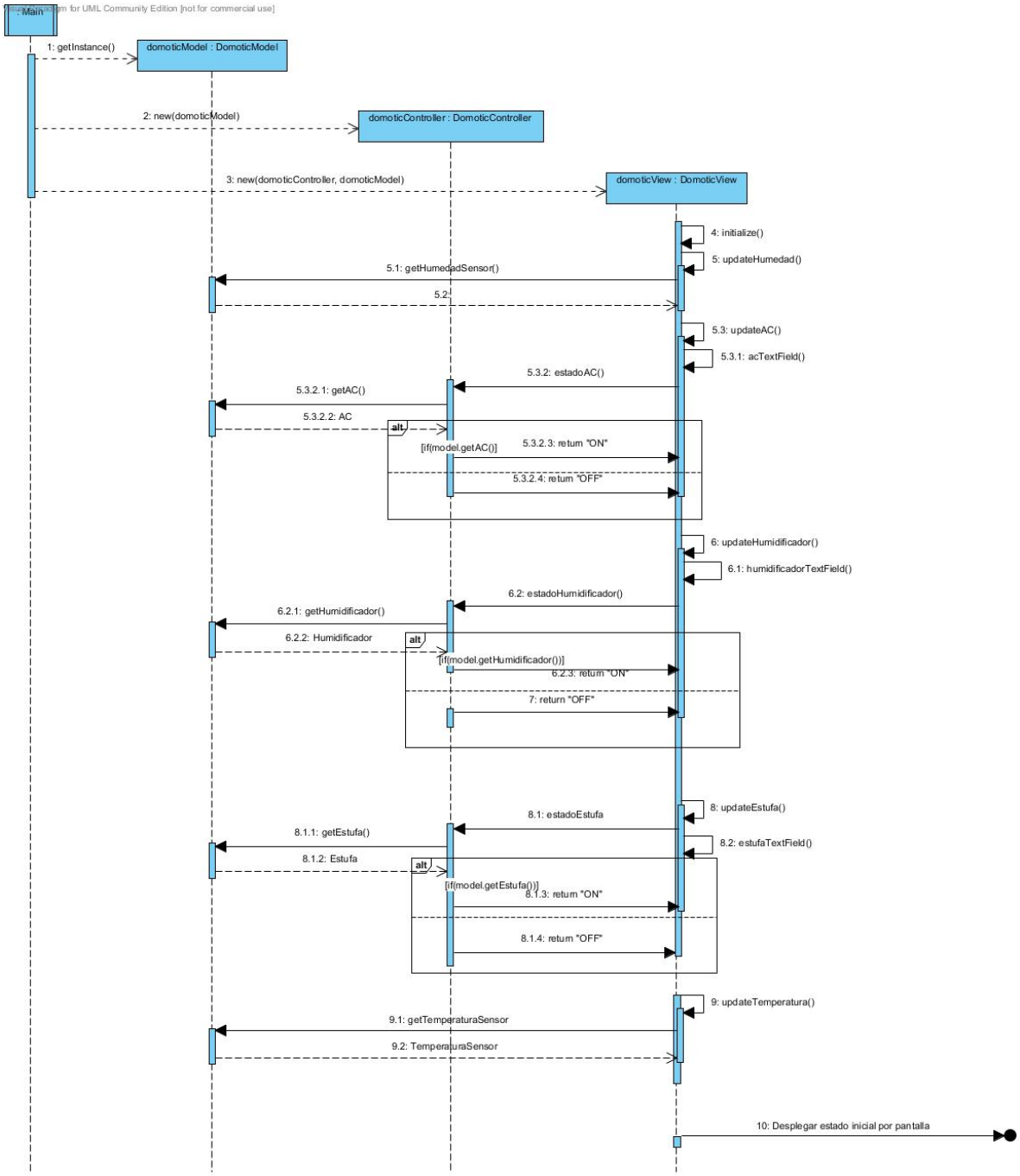
A continuación se presenta un diagrama de clases del sistema.



(Figura 1. Diagrama de clases del sistema)

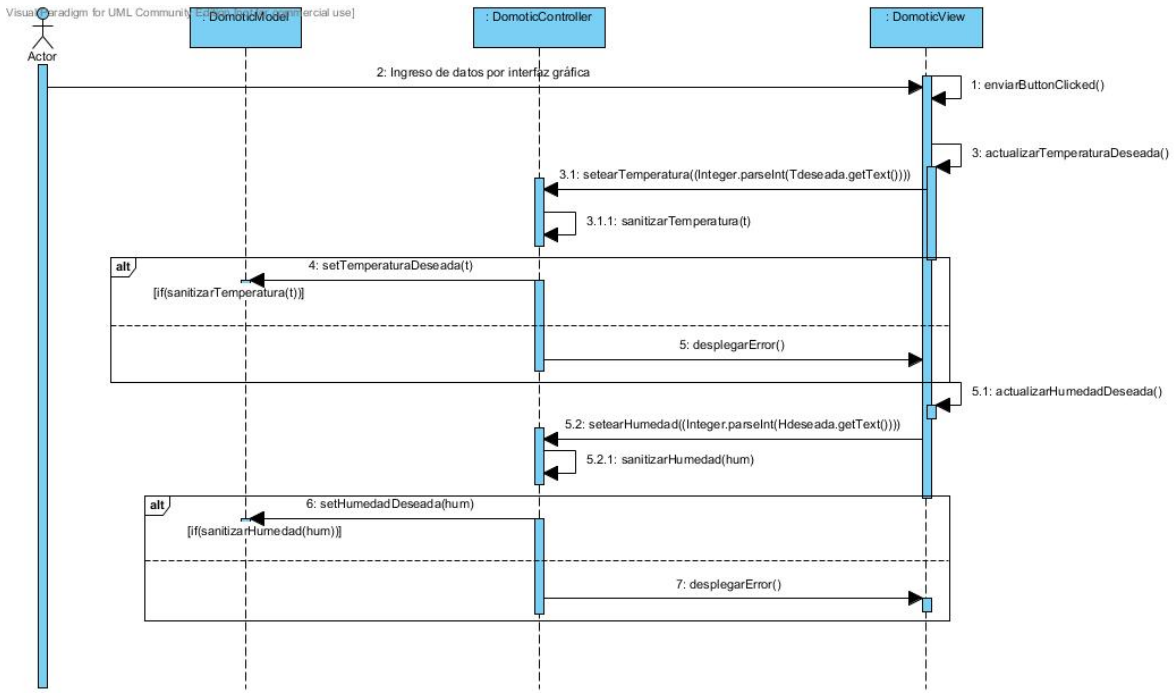
En la figura 2 se presenta el estado inicial del sistema como un diagrama de clases. En el diagrama se puede ver que una clase Main creará un modelo que será Singleton. La razón para utilizar Singleton es que se quiere forzar a que no haya un duplicado de modelo de datos en el sistema, lo cual podría generar complicaciones a la hora del manejo correcto de la información. Utilizando el Singleton, salvamos la posibilidad de que existan dos o más modelos en una ejecución del programa. Al existir solamente una instancia de la clase Model, se asegura la unicidad de los datos. Luego se crea una clase Controller, a la cual se le asigna el Model para que éste lo controle. La vista tendrá campos conteniendo tanto al modelo como al controlador. La fundamentación es que la vista se suscribirá como un observador de un modelo que implementa la interfaz observable y la vista actuará como receptor de los inputs del usuario y se los pasará al controlador para que éste realice operaciones. Más adelante en este mismo documento se detallará la utilización del patrón Observer.

En la figura 2 se detalla el estado inicial que se desplegará una vez que el usuario inicie la aplicación. La vista se inicializará con los valores iniciales almacenados en el modelo, utilizando el controlador como intermediario.



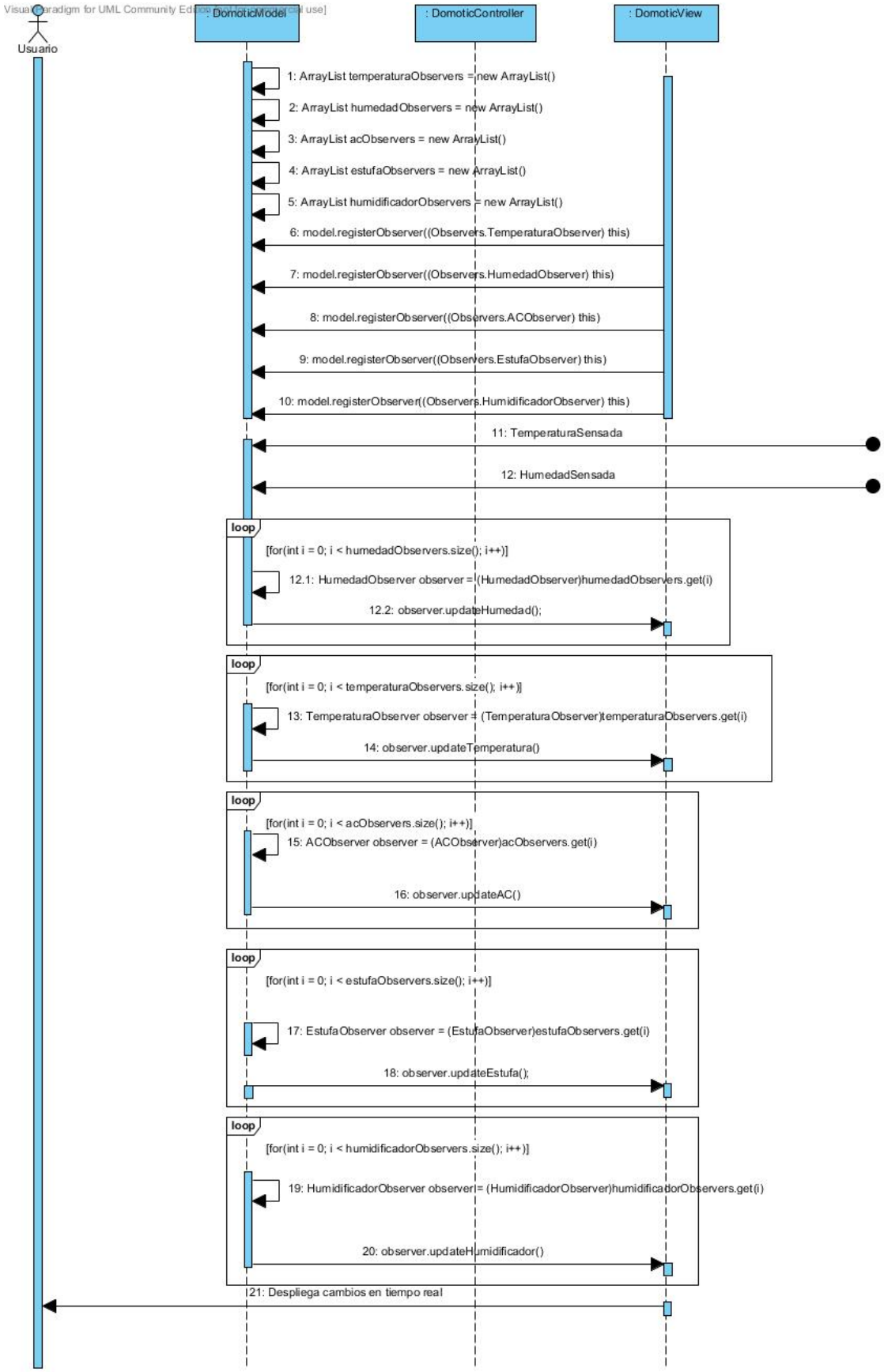
(Figura 2. Inicialización del programa)

En la figura 3 se expone el caso en el que un usuario ingresa datos por medio de la interfaz gráfica en el modo preset. Los datos son actualizados en el modelo si es que éstos pasan una prueba de sanitizado de la expresión, para evitar que el modelo se actualice con valores inválidos.



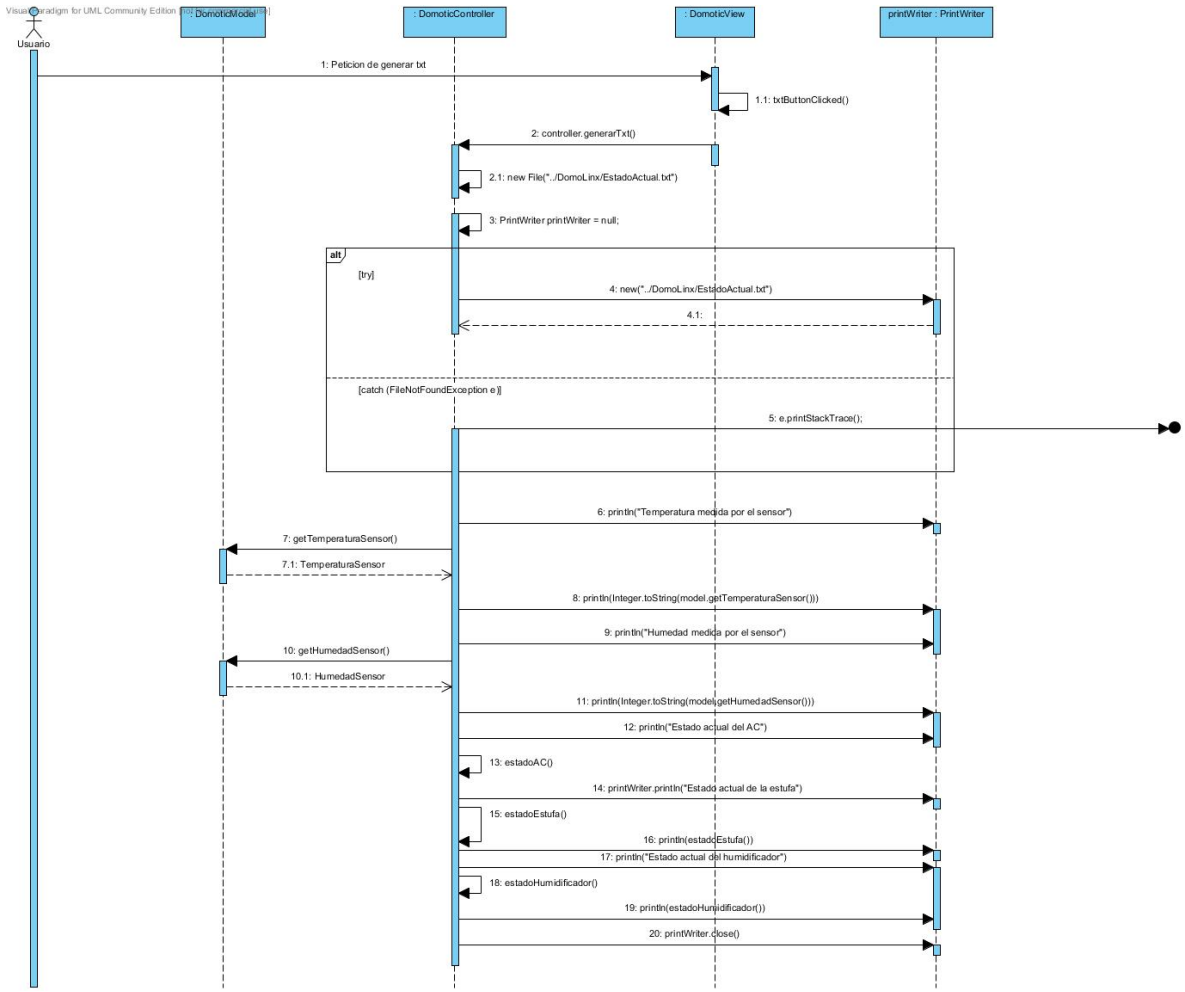
(Figura 3. Ingreso de datos en el modo preset)

En la figura 4 se presenta con detalle cómo será utilizado el patrón observer. La vista implementará interfaces de observador para el caso de la temperatura sensada, la humedad sensada y los estados de la estufa, el AC y el humidificador. Cuando ocurren cambios en el modelo de cualquiera de estas variables, el modelo, que implementa la interfaz subject (observable), recorrerá los arraylists de los observadores correspondientes suscriptos e irá actualizando los estados de los mismos por medio del método notifyObserver. De esta manera, el sistema se actualiza en tiempo real en presencia de cualquier cambio de las variables observadas.



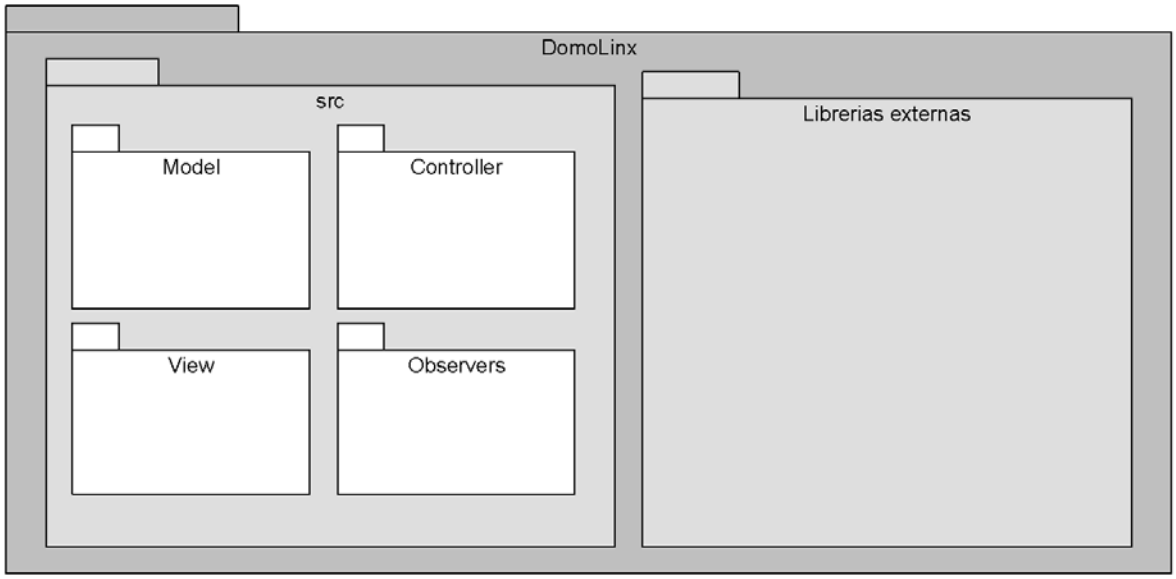
(Figura 4. Interacción entre los observers y las variables observables en presencia de un cambio)

En la figura 5 se presenta un diagrama de secuencias que detalla la interacción de las clases en el caso en el que un usuario desea generar un archivo con extensión “.txt” con los datos actuales del sistema. En este caso el controlador, se encarga de generar un archivo nuevo y llenarlo con los datos actuales del sistema. Este archivo se almacenará en la carpeta /DomoLinx.



(Figura 5. Generación de archivo .txt con el estado actual del sistema)

El esquema de directorios del proyecto estará organizado como se muestra en la figura 6.



(Figura 6. Esquema de directorios del proyecto)