Documento de Arquitectura

Aplicación DomoLinx

En el presente documento se expone:

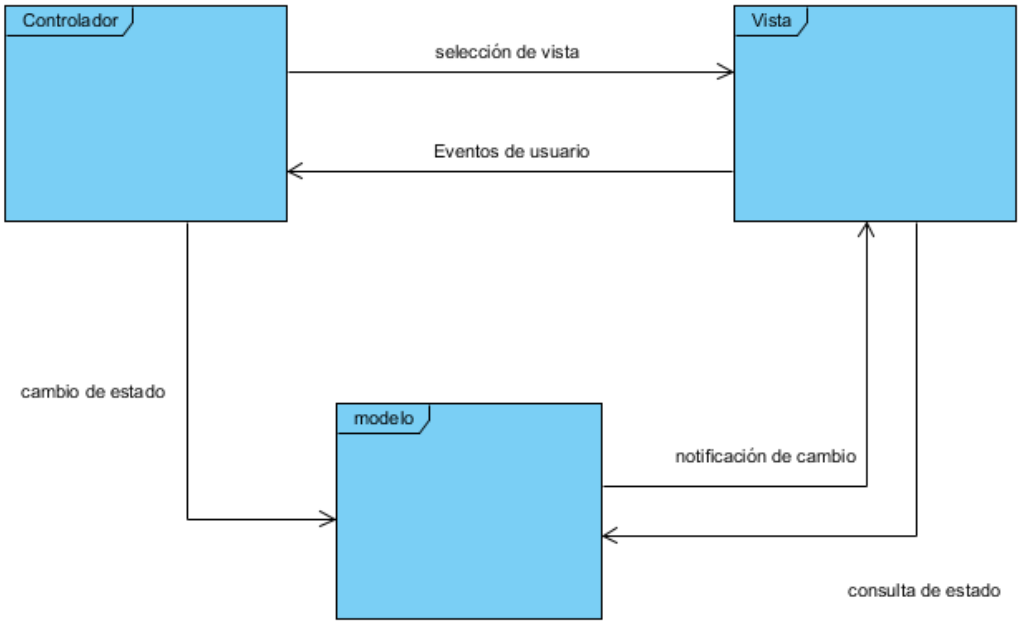
- Elementos y los principales componentes del sistema.
- Módulos del sistema.
- Mecanismos arquitecturales necesarios para el sistema.
- Tecnologías elegidas para hacer frente a cada capa y el mecanismo y la motivación detrás de estas opciones.
- Descripción de los principales patrones de diseño utilizados y por qué la elección.

La aplicación DomoLinx implementa el patrón arquitectónico MVC (Model View Controller), debido a que permite el desacoplo de código de los principales componentes, y mejora la mantenibilidad. Permite la reutilización de código, permite a los miembros del equipo trabajar en forma paralela en cada uno de estos módulos integrados.

Requerimientos No funcionales cubiertos por el patrón de arquitectura:

- Mantenibildiad: La arquitectura MVC se sirve de patrones de diseño como Strategy, el cual permite seleccionar entre varios componentes escritos implementado una interfaz predefinida, lo que los hace intercambiables y permite agregar nuevos módulos cuando se necesiten.
- El patrón observer que permite notificar a todos los objetos registrados al modelo. En este caso la vista y los datos que esta despliega cambian inmediatamente después que los mismos han cambiado en el sujeto observado (modelo).
- Los datos que alberga y manipula el modelo deben ser tratados en una única instancia de la clase domoticModel, por tal motivo se implementó el patrón Singleton para tal fin.

Diagrama de Arquitectura MVC



Su fundamento es la separación del código en tres módulos diferentes:

El Modelo: Este módulo es el que trabaja con los datos, contiene los mecanismos para acceder a la información y para actualizar su estado.

La Vista: Es el módulo que contiene el código que implementa la interfaz de usuario. Mediante el patrón de diseño Observer, las diferentes vistas de la aplicación son notificadas cada vez que los datos del modelo sufren un cambio.

El Controlador: Este módulo, contiene el código necesario para responder a las acciones que se solicitan en la aplicación. Es una capa que media entre la/s vista/s y el/los modelo/s. Al usar el patrón Strategy, un desarrollador podría cambiar el Modelo (ya que este implementa una modelInterface) y seguir utilizando el mismo controlador.

Diagrama de componentes

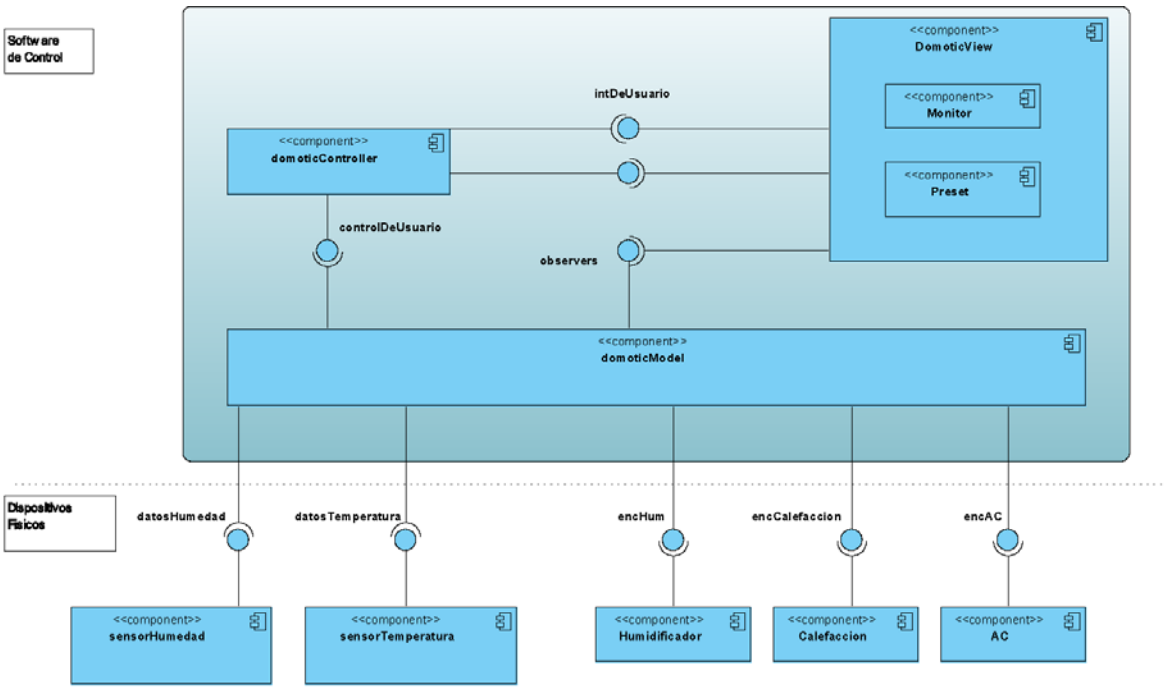


Diagrama de Despliegue