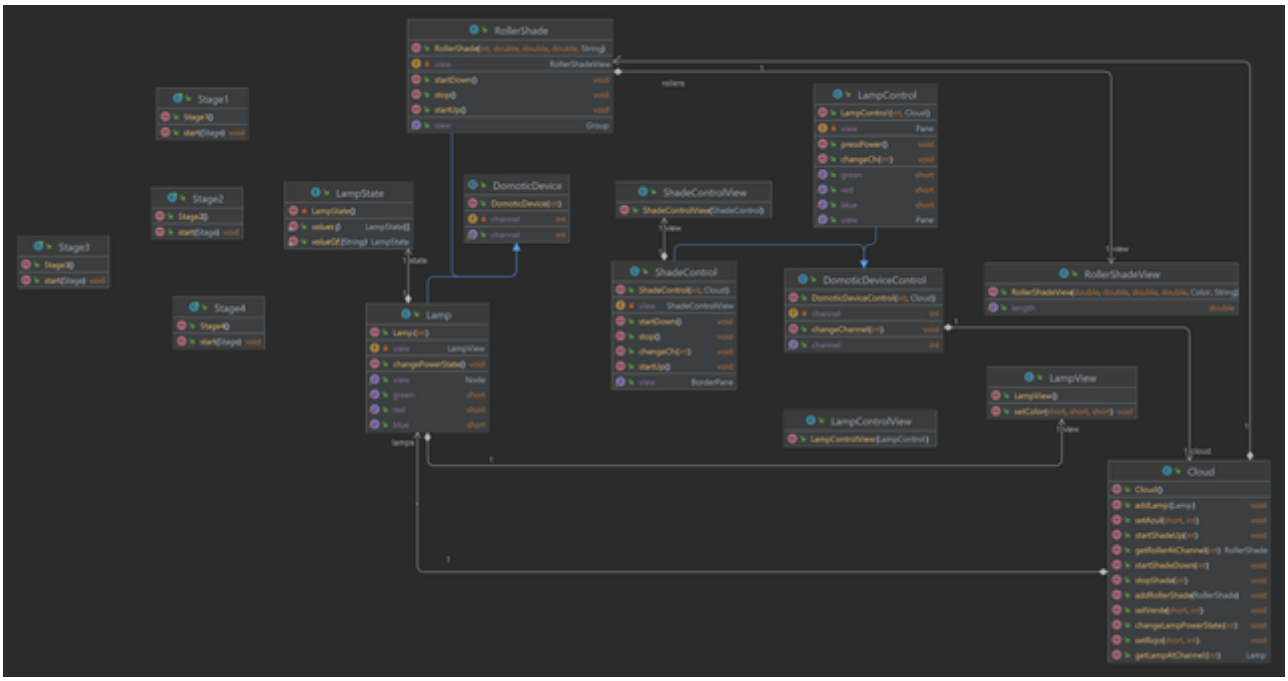
Documentación Tarea 2 - Elo329

Documentación Tarea 2 - Elo329

Diagrama UML



Solución

El objetivo de la **tarea 2** era extender la **tarea 1** mediante la integración de un **modelo gráfico** que represente los **dispositivos domoticos** trabajados en la tarea 1 (Cortinas y Lamparas RGB). Particularmente el `stage 4` nos solicita llevar a cabo la representación de **2 Cortinas** y de **2 lamparas** que deben funcionar en **canales individuales**, además de sus respectivos **controles**. Mediante la utilización de **javaFX** llegamos a la solución representada en el diagrama UML anteriormente añadido.

Interaccion de clases:

- En la clase `stage 4.java` que es la que contiene el metodo `start()` hacemos el llamado a los constructores de `Lamp.java` , de `RollerShade.java` , sus respectivos controladores y de `Cloud.java` . Las instancias de `Lamp.java` y `RollerShade.java` son añadidas a la instancia de `Cloud.java` .
- `stage 4` obtiene las visualizaciones de las cortinas, lamparas y controles mediante el metodo `getView()`.
- `RollerShade.java` crea una instancia de `RollerShadeView.java` en su constructor, para luego ser accesada mediante `getView.java`.

- `ShadeControl.java` crea una instancia de `ShadeControlView.java` en su constructor, para luego ser accesada mediante `getView.java`.
- `Lamp.java` crea una instancia de `LampView.java` en su constructor, para luego ser accesada mediante `getView.java`.
- `LampControl` crea una instancia de `LampControlView.java` en su constructor, para luego ser accesada mediante `getView.java`.
- Al momento de interactuar en la GUI, las instancias de `LampControlView.java` y `ShadeControlView.java` activan los metodos correspondientes a las instancias de `ShadeControl.java` y `LampControl.java`. Quienes, a su vez, llaman a los metodos correspondientes de la instancia de `Cloud.java`. Luego, la instancia de `Cloud.java` llama a los metodos de las instancias de `RollerShade.java` y de `Lamp.java` que efectuaran el cambio, y posteriormente lo haran repercutir en las instancias de las clases `RollerShadeView.java` y `LampView.java` segun corresponda.

Dificultades

- Lograr un correcto posicionamiento de los controles de la cortina fue un verdadero dolor de cabeza. Sin embargo, vimos que al utilizar un GridPane era mucho mas sencillo lograr lo que queriamos sin la necesidad de batallar con los metodos de BorderPane en el que venia por defecto.
- Efectuar el cambio de canal de los controles en un modelo dinámico (En referencia a que sea escalable automaticamente el numero de cortinas o lamparas) resultó ser una tarea demasiado complicada para el equipo. Por lo tanto, decidimos optar por un modelo estático, con 2 cortinas y 2 lamparas.
- Lograr el cambio de canal desde la clase `ShadeControlView.java` y `LampControlView.java` solo teniendo acceso a los controles nos parecio una tarea imposible en un principio. Sin embargo, nos dimos cuenta que creando un metodo `changeChannel()` en la clase `DomoticDeviceControl.java` que es de donde heredaban ambos controles, se nos hacias mucho mas facil la tarea.