***Tabla de decisiones de diseño***

| **Fecha** | **Decisión** | **Ventaja** | **Desventaja** | **Alternativa** |
| --- | --- | --- | --- | --- |
| 25/04/2018  (Entrega 0) | Modelar una clase Categoría, con sus atributos nombre, consumo, cargoFijo, cargoVariable. Se va a instanciar cada categoría. | Al crear una clase Categoría, podemos instanciar la cantidad de categorías que queramos, pasándole como parámetros sus atributos. | Hay que instanciar la cantidad de categorías que haya, pasarle como parámetros tanto los cargos fijos como los variables y esto nos dejaría setear cualquier valor. | * Modelar una clase por cada categoría. |
| 25/04/2018  (Entrega 0) | Implementar Usuario como una clase abstracta ya que en futuras modificaciones el sistema será más mantenible al querer agregar otro ROL. | Al implementar Usuario como clase abstracta, las subclases Administrador y Cliente heredan atributos en común y ayuda a la mantenibilidad. | Una vez que se inicialice la clase, no puede ser modificada. | * Utilizar una única clase Usuario con un atributo identificador |
| 06/06/2018  (Entrega 1) | Implementar el patrón Observer para la comunicación de las mediciones entre los sensores y las reglas. | Se abstrae el problema de la comunicación de mediciones, el sensor solo sabe que hay una lista de observadores y en una sola llamada los notifica. |  |  |
| 06/06/2018  (Entrega 1) | Implementar una lista de dispositivos en cada sensor. | Permite definir un grupo de dispositivos para que sean controlados por un sensor. |  | * Que cada dispositivo tenga un sensor instalado. |
| 06/06/2018  (Entrega 1) | Se decidió implementar una interfaz para el tipo de dispositivo. | Se puede cambiar fácilmente en tiempo de ejecución el tipo de dispositivo (inteligente y estándar) y por consiguiente su comportamiento |  |  |
| 06/06/2018  (Entrega 1) | Se decidió generar un repositorio con una lista de logs de los cambios de estado del dispositivo con su fecha respectiva | Gracias a esto podemos calcular el consumo del dispositivo en un intervalo de fechas dadas | el costoso manejo de archivos json y contemplar muchísimas excepciones para los casos especiales en los que las fechas no estén dentro del intervalo en el que se desea buscar el consumo | * Incorporar una base de datos con el log de los dispositivos, eso nos ahorrara todo el manejo de json y podremos directamente traer de la base con un query los logs que queramos según el periodo solicitado y el dispositivo en cuestión |
| 06/06/2018  (Entrega 1) | Usar switch case en estado | Como el requerimiento solicitado es muy sencillo, nos ahorra crear 4 clases (estado, ahorro, encendido, apagado) | Si se desea en un futuro agregar más estados habría que tocar el código en vez de agregar más clases si hubiésemos aplicado el patrón state | La utilización del patrón state para que se cambie el comportamiento del objeto según el estado que posea |