Fecha	Decisión	Ventaja	Desventaja	Alternativa
3/4	Por votación de la mayoría, el sistema se desarrollará en java	lenguaje orientado a objetos, soporte de la cátedra		
3/4	Al compartir los mismos atributos, se eligió que cliente y administrador hereden de una clase "usuario"	menos código repetido en las clases		
3/4	Las categorias son 9 objetos que heredan de una clase "categoria", y no 9 clases "categorias "	Si aparecen nuevas categorias no es necesario crear una clase por cada categoria nueva		
7/4	El tipo del atributo "documento" es de tipo String y el atributo "numeroDocumento" es de tipo Integer y no una clase "Documento"	El documento es mas sencillo de manipular		
8/4	La clase Cliente, junto con la Clase Administrador, heredan el atributo "id"	Identificación univoca de los clientes residenciales mediante un identificador		
8/4	La clase Cliente no posee otro atributo, mas alla de los que ya se especificaban en el enunciado y en el punto anterior.	Los atributos mínimos y necesarios		
10/4	Los repositorios tienen la habilidad de poder cambiar el DataAcessObject que van a utilizar	Permite polimorfismo de "DAO's"		
15/5	Que "DispositivoStandard" sea un estado	Permite flexibilidad para convertir un dispositivo de standard a inteligente		Se podría hacer un patrón adaptador con el módulo inteligente, pero en este caso no se justifica pues no agrega comportamiento distinto al dispositivo inteligente.
15/5	Aplicar patrón de estado para los estados del dispositivo	Flexibiliza los cambios de estados		
15/5	Aplicar patrón de comando para las reglas con los actuadores	Permite hacer reglas configurables, agregando y sacando actuadores, que actuarían de comandos		
16/5	Elegimos el patrón observador para las reglas y el sensor			
16/5	Decidimos que exista un único sensor con varias mediciones como atributos (humedad, intensidad lumínica, temperatura, movimiento)	Facilita la comunicación con las reglas, unifica parámetros y simplifica la implementación	No es extensible a N tipos de mediciones distintas	Crear una interfaz o clase abstracta Sensor que permita N tipos de mediciones distintas
18/5	Uso del patrón State para manejar los estados de un Dispositivo: Un DispositivoStandard (estado de Dispositivo), puede cambiar de estado (AhorroDeEnergia, Prendido, Apagado) convirtiéndose así en DispositivoInteligente, el recíproco no vale, un DispositivoInteligente no se hace Standard	Se puede enviar un mismo mensaje a todos los dispositivos (Inteligentes y Standard), los mismos saben cómo responder a dichos mensajes y realizan coherentemente las acciones	El dominio no especifica qué pasa si el "adaptador" utilizado deja de utilizarse, y el dispositivo volviera a ser Standard en caso de que fuera posible: no se contempla este caso.	Utilizar un adapter para convertir a los dispositivos standard en dispositivos inteligentes, ésta opción fue descartada dado que es una aproximación innecesaria al "mundo físico", cuando una abstracción de éste tipo se maneja de manera más sencilla con el patrón State, abstrayéndonos de la manera en que el usuario lleva a cabo la acción.
18/5	[Del punto anterior] Un DispositivoInteligente no puede convertirse en Standard, se atrapa dicho error, y se lo maneja			
18/5	Cambio en la decisión del 16/5: decidimos usar una clase concreta Sensor, al momento de instanciar, distintos tipos de sensores, lo haremos con esta clase.			