UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

DISEÑO DE SISTEMAS

Trabajo Práctico Anual
“Sistema de Gestión Energética”

Grupo: 2

Integrantes:

- Alejo Scotti - alejoscotti@gmail.com - 1528142
- Juan Pablo Ferreira - juanpabloferreira88@gmail.com - 1275902
- Ivan Metta
- Tomas Villa

Fecha de entrega: 10/07/2018

Profesor: Martín Agüero

Ayudante a cargo: Alejandro Ezequiel Leoz - Nicolas Contreras

Repositorio:https://github.com/tomivilla/DDS-Grupo2

Branch: master

Commit ID:

Registro de cambios

Fecha	Modificaciones
12/05/2018	Patron State para el estado de los dispositivos inteligentes
12/05/2018	Patron decorardor para adaptadores
13/05/2018	Patron Bridge para los Actuadores
16-05-2018	Diagrama de Clases preliminar con los patrones seleccionados.
11-06-2018	Diagrama de Clases - Se cambió el patro Decorador por un Adapter para los adatadores de dispositivos estandar
09-07-2018	- Se agrega la clase de simplex proporcionada por la catedra - Se agrega la clase de DispositivoFactory para seleccionar los dipositivos que se necesitan instanciar, con su caracteristica de consuo y tipo de dispositivo sengun la entrega 2 . - Se agrega la clase de PeriodoFactory para porder hacer las pruebas - Se agreg la prueba maximización test corespondiente a los datos de prueba que se solicitaron en la entrega 2.
10-07-2018	Se agrega logica de scheduler y test para la acarga de zonas y transformadores.
10-07-2018	Diagrama de clases entrega 2

Tabla de Requerimientos no funcionales TP - Entrega 2

Fueron identificados los siguientes requerimientos no funcionales. Se tuvieron en cuenta aquellos que afectaran a la arquitectura del sistema.

Permitir la instalación de transformadores y zonas.

Diagrama de Clases TP - Entrega 2

Diagrama de Clases Entrega 2.zi

Tabla de decisiones de diseño:

Fecha	Decisión	Ventaja	Desventaja	Alternativa
12/05/2018	La idea sería usar un **Estate** para los **Estados** de los **Dispositivos Inteligentes**.	Permitir que un objeto altere su comportamiento cuando su estado interno cambia. Permite modelar las transiciones entre estados.		
12/05/2018	Para los **Adaptadores** de los **Dispositivos Estadar** usaríamos un **Decorador**.	Agregar dinámicamente responsabilidades (funcionalidad) extra a un objeto. Es una forma flexible que sirve de alternativa a subclassing para extender funcionalidad. Mas flexibilidad que la	Un decorador y su componente no son identicos.	Estrategias

		herencia estática.		
13/05/2018	Para el **Actuador** se usaría el patrón **Bridge** via la inteface **Implementador** para no depender de implementación que tiene cada electrodomestico según su fabricante como indica el enunciado.	Desacoplar una abstracción de su implementación, de modo que ambas puedan variar de forma independiente.		Composite
13/05/2018	Se agregó el patrón **Observer**, para avisar cada vez que un **Sensor** realiza una medición de la magnitud que corresponda	Definir dependencias one-to-many entre objetos, de forma tal que cuando un objeto cambia su estado todos los objetos dependientes son notificados y actualizados inmediatamente		
16/05/2018	Una **Regla** tiene un listado de Condiciones y Acciónes, ademas conoce a un dispositivo, sobre el cual comprobaría el cumplimiento de las Condiciones, en caso de cumplirse todas las condiciones, ejecutaría las acciones. A su vez el cliente definiría las reglas a aplicar sobre sus dispositivos.			
11-06-2018	Se cambió el patror Decorador por el patron Adapter para agregar el Adaptador de dispositivos Estandar.	Permite que dos clases incompatibles puedan funcionar en conjunto.		Decorador
09-07-2018	Patron Factory, para crear dispositivos	Esta a cargo de la creación de instancias de dispositivos, segun la tabla indicada en la entrga 2.		