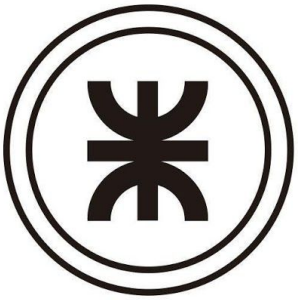


UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

DISEÑO DE SISTEMAS

Trabajo Práctico Anual
“Sistema de Gestión Energética”

Grupo: 1
Integrantes:

- Jonathan Strelczuk 116.565-3
- Mauricio Rocha 158.090-5
- Guido Dicomo 121.305-2
- Flavia De Rosa 158.739-0

Fecha de entrega: 22/05/2018
Profesor: Martin Aguero
Ayudante a cargo: Martin Aguero

Repositorio: https://github.com/jstrelczuk/dds-tp-2018-grupo-01.git

Branch: Master
Commit ID: 8fe6819

Diseño de Sistemas - SGE - Grupo 1 - Entrega 1

Sumario

Registro de cambios

Fecha	Modificaciones
16/05/2018	Se incorpora el patrón de Diseño Estructural **Decorator**, para modelar los dispositivos de tipo Estándar / Inteligentes
20/05/2018	Se modifica el patrón de diseño Decorator por **Adapter**, por considerarlo mejor para el requerimiento.
12/06/2018	Se suprime el Patrón Adapter, del modelado de la conversión del DEstandar a Destandar-Inteligente
20/06/2018	Se agrega el patrón **Observer** para el modelado de las reglas y el **Singleton**, para la creación de un Manager de Reglas y un Manager de Dispositivos
21/06/2018	Se implementa el patrón **Proxy**, para especificar, de qué manera el Sensor accede a los métodos de cada DI, que necesite.

Tabla de decisión grupal, sobre el diseño

FECHA	DECISIÓN	VENTAJA	DESVENTAJA	ALTERNATIVA
16/05/2018	Utilizamos Decorator para modelar los dispositivos Estandar / Inteligentes	Permite agregar funcionalidad extra a los dispositivos estándar, por medio de un adaptador, y de esta manera adquirir las funcionalidades de un dispositivo Inteligente.	No cumple con el requerimiento. Agrega funcionalidad, pero la idea es crear una interfaz compatible a través de un adaptador	Se cambia por el patron Adapter.
17/05/2018	Utilizamos Command para la incorporación de Actuadores y Sensores.	Permite controlar eficientemente magnitudes medidas, donde a través de una serie de reglas preestablecidas, determinarán las acciones asociadas a cumplirse por el Actuador con los dispositivos.		
20/05/2018	Cambiamos al patrón Adapter, por considerar que aplica mejor al requerimiento.	Permite que dos interfaces sean compatibles a través de un adaptador. Para el requerimiento, el dispositivo Estándar + Adaptador, funcionará como un DI.		
21/05/2018	Se agrega un enum EstadoDispositivo.	Permite cambiar a los distintos modos de funcionamiento, que tiene el dispositivo. (Encendido, Apagado y AhorroEnergia).		
12/06/2018	Se suprime patron Adapter		Resulta obsoleto para la implementación. (Cuando se implementó en el Dispositivo Adapter, siempre se usaban los métodos de la Clase Adaptee).	Se implementa el metodo AgregarAdaptador(), en la clase Estandar, que se encarga de realizar la conversión del requerimiento, que retornando una instancia de Dinteligente.
12/06/2018	Se suprime patrón Command		Añade complejidad al modelo.	Se crea una nueva entidad Driver, que contendrá la lista de actuadores, y la lista de sensores,

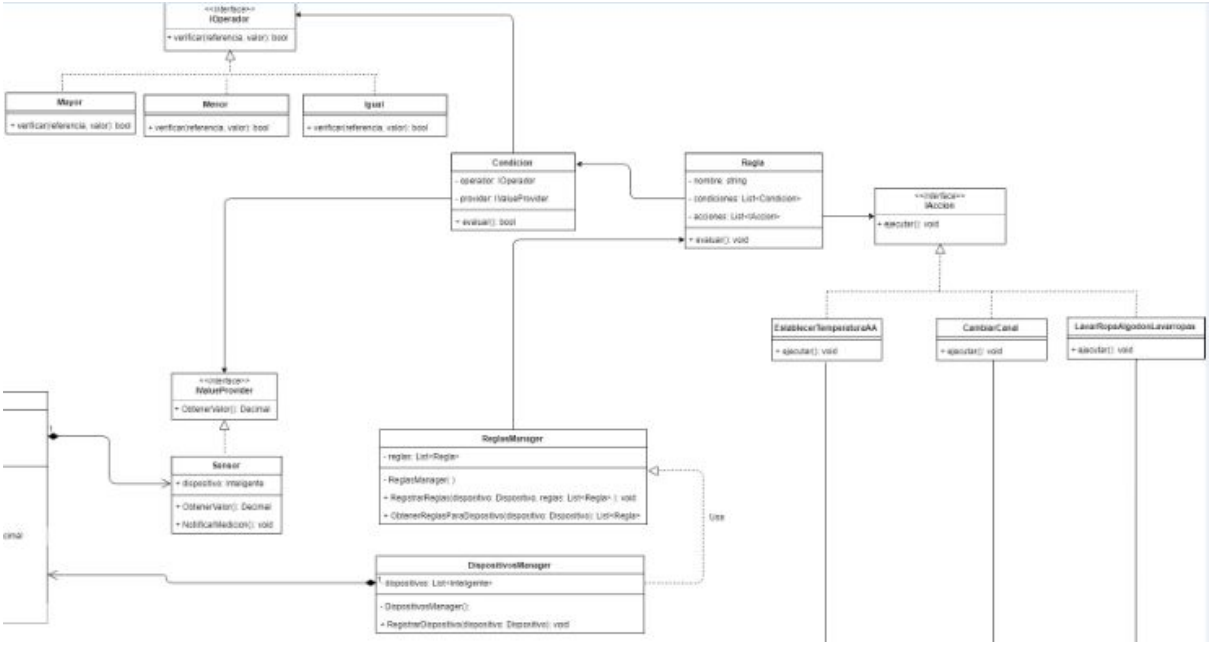
				correspondientes. Además de la implementación de los métodos que corresponden a cada tipo de DI.
20/06/2018	Se Agrega patrón Observer	Tomando la idea propuesta por el ayudante de la materia: Evita el acoplamiento. Notifica y actualiza a todos los dispositivos dependientes		
20/06/2018	Se agrega Patrón Singleton	Tomando la idea propuesta por el ayudante de la materia: Se crea un Manager de Reglas y un Manager de Dispositivos, lo cual permite tener un acceso global a dispositivos y a reglas.		
21/06/2018	Se agrega patrón Proxy	Permite controlar cómo accede un sensor, al método del DI correspondiente.		
23/06/2018	Se agrega una Entidad Activación	Permite registrar el momento donde el dispositivo es apagado, guardando intervalos de tiempo y la fecha.		

REQUERIMIENTOS / OBJETIVOS A CUMPLIR

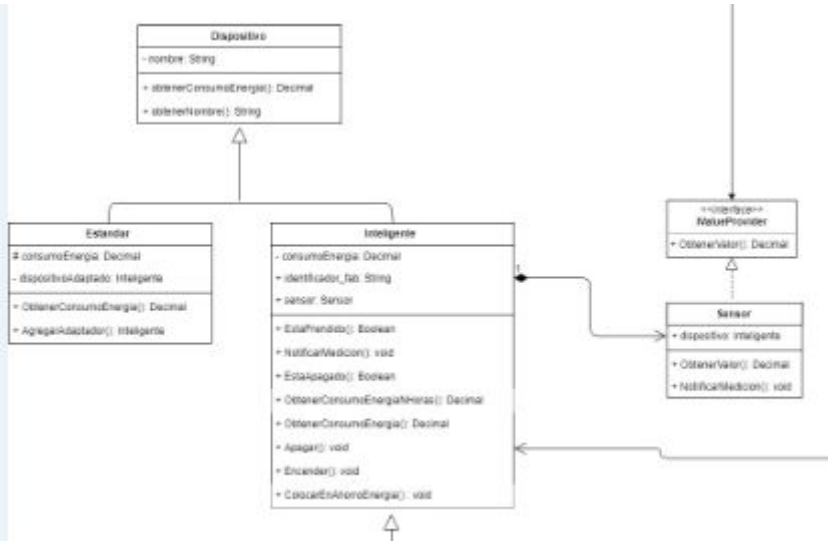
1. Modelar la conversión DEstandar -> Destandar-Inteligente
2. Calcular el puntaje total de los clientes por cada dispositivo registrado y por cada DEstandar convertido.
3. Modelar Actuadores: los que son los encargados de enviar diferentes acciones a cada DI las cuales varían de acuerdo al DI.
4. Modelar Sensores: miden magnitudes proporcionadas por los DI.
5. Reglas: las cuales contienen un conjunto de condiciones y estarán asociadas al tipo de DI.

SOLUCIÓN PLANTEADA

1. Se plantea el método AgregarAdaptador, en la clase Estandar que retorna una nueva instancia del DI.
2. Se plantea el método registrarDispositivo() y convertirADI(), los cuales suman 15 y 10 puntos respectivamente.
3. Se plantea la aparición del Driver, el cual a partir de una interfaz, contendrá la lista de las Interfaces de cada DI. Modelamos la acción que se aplica a cada DI, la cual, está asociada a reglas, sujetas a un conjunto de condiciones que contienen operadores de comparación.



4. El Sensor se modela, dado que es el que toma la magnitud del DI, y es específico a cada DI.



5. Las Reglas se modelaron, teniendo en cuenta la idea propuesta por el ayudante de la materia. Comenzando por un caso de Regla Base, y yendo al modelado que se necesitaba para nuestro sistema, donde el tipo de sensor, obtiene un valor de magnitud específico de un DI, y en consecuencia, aplica un conjunto de condiciones asociadas a reglas más específicas, y termina en la acción a llevar a cabo, según el DI, . Usando 2 patrones de diseño (Observer y Singleton), Modelando el manager de Reglas y el Manager de Dispositivos.

DIAGRAMA DE CLASES

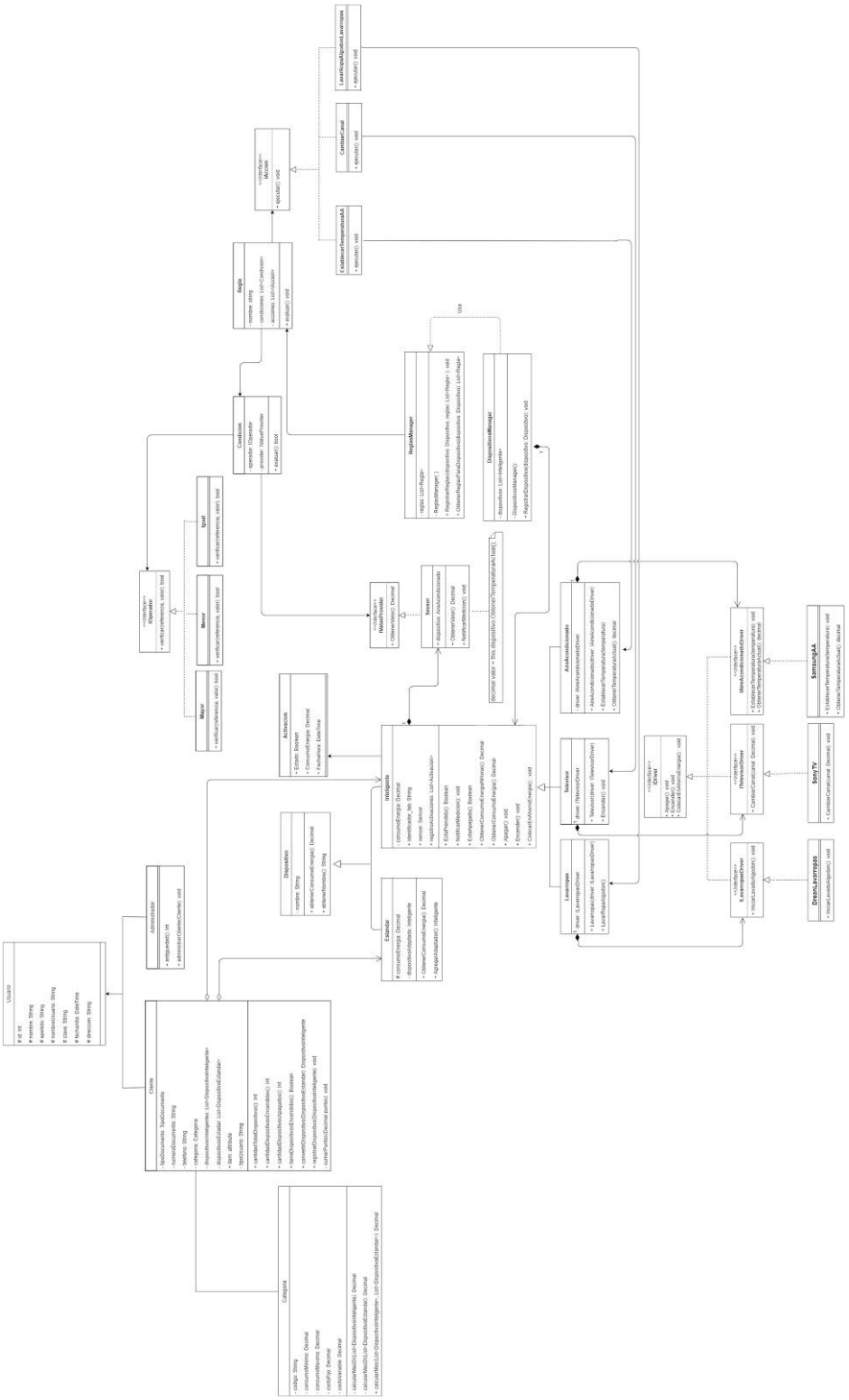
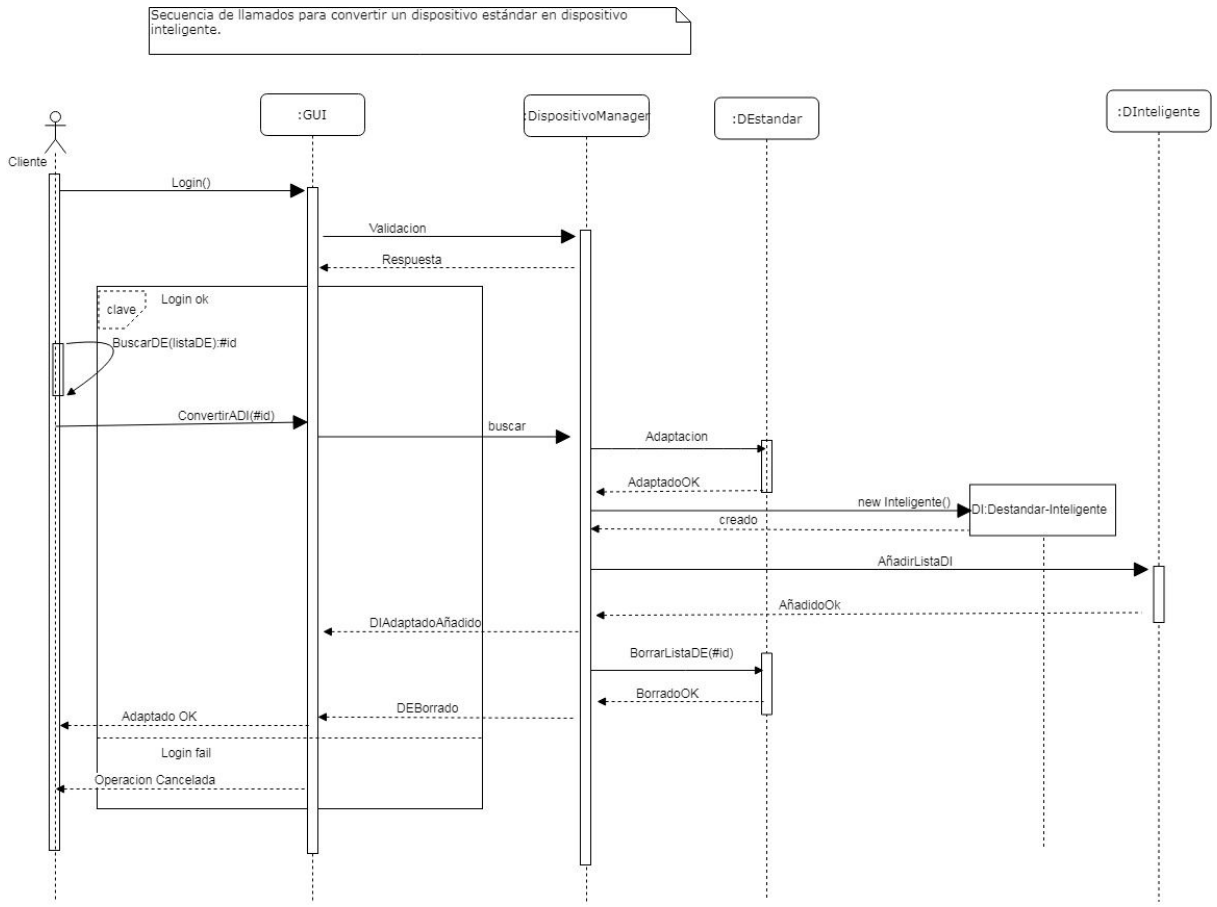


DIAGRAMA DE SECUENCIA



Comunicación entre sistema y dispositivos

Pensamos en la tecnología que ofrece IoT (Internet de las cosas), con la finalidad de conectar el maximo de nuestros dispositivos entre ellos y con nosotros (clientes, administradores del sistema).

El objetivo final, sera que el entorno del usuario este conectado de manera transparente, y así, aprovechar al máximo, los recursos de los dispositivos y sus utilidades.

En sus hogares, con todos sus dispositivos conectados a internet a través de SGE, pudiendo medir ciertos parámetros como luz, humedad, temperatura, errores ocurridos, tiempo en horas de consumo, cantidad de dispositivos encendidos y/o apagados, entre otros, de forma automática y sin la intervención del usuario (en el caso de los DI).
Esto permitirá, generar la información del consumo energético de cada cliente por hogar, para que nuestro SGE, pueda tomar decisiones en tiempo real y de esta manera optimizar el uso de la energía.

Herramientas necesarias:
Procesadores y plataformas, se encargan de gestionar la información.

- ARM Cortex M
- Arduino Risc de Atmel
- Intel Quark
- MediaTek
- Samsung o Qualcomm

Sensores: es el hardware que va a interactuar entre nuestra tecnología y el entorno, capturando los datos que nosotros deseemos.

- Arduino: **sensores táctiles, acelerómetros, de inclinación, potenciómetros, de humedad y temperatura, altitud, presión...**casi cualquier cosa que imaginemos se puede medir con estos sensores.

Protocolos de comunicación: muchos de los que conocemos, se encuentran vigentes para IoT. Como por ejemplo:

- Conexiones de red local vía Ethernet o de transmisión inalámbrica a través de conectividad móvil.

Vodafone e IBM : donde las velocidades de conexión permitirán alcanzar a los próximos protocolos de comunicación, como el 5GB.
También hay nuevos protocolos que han sido ideados pensando en IoT y la comunicación de objetos entre ellos y a corta distancia. Por ejemplo:

- NFC o también **Bluetooth 4.0**, LE 'Low Energy'. Está pensado para ser implementado en sistemas con baterías reducidas (pulseras cuantificadoras).