Contenido

[Entrega 3 2](#_Toc529265936)

[Comentarios y justificaciones de la Entrega 3 2](#_Toc529265937)

[Casos de prueba (mínimos) 2](#_Toc529265938)

[Justificaciones 4](#_Toc529265939)

[Tecnologías elegidas para la implementación 4](#_Toc529265940)

[Otros entregables (Documentación): 4](#_Toc529265941)

[Instrucciones para correr tests unitarios contra MYSQL 5](#_Toc529265942)

# Entrega 3

## Comentarios y justificaciones de la Entrega 3

|  |  |
| --- | --- |
| **IMPLEMENTACIÓN** | |
| **Requerimiento** | **Solución propuesta (decisión de diseño)** |
| El sistema deberá persistir tanto a los clientes como los administradores, teniendo en cuenta todos los atributos mínimos de la Entrega 0 para cada entidad. | Si. Ver, por ejemplo, el test:  *casosDePruebaMinimos.PruebaTransformadores.java*  donde se persiste un administrador con todos sus atributos. En el test *casosDePruebaMinimos.PruebaUsuario.java* se muestra como se persisten usuarios de tipo cliente. |
| Los administradores son los únicos usuarios que pueden crear, modificar o eliminar los dispositivos que soportará el sistema (los soportados actualmente son los que figuran en la tabla de dispositivos de la Entrega 2). | Resuelto en la Entrega 2 con el patrón Prototype. Se añadió el requerimiento de que no se puedan instanciar dispositivos a través de sus constructores: solo el administrador puede crear dispositivos. Al intentar instanciar un dispositivo inteligente o estándar mediante *new DispositivoInteligente()* o *new DispositivoEstandar()* se comprobará que no es posible. |
| El sistema deberá persistir los dispositivos que utilizan los clientes junto con el consumo generado por cada uno. | Si. Ver, por ejemplo, el test:  casosDePruebaMinimos.PruebaDispositivos.java |
| El sistema debe tener trazabilidad (persistir) los estados por los que paso un dispositivo. | Si. Ver el test:  *casosDePruebaMinimos.PruebaDispositivos.java*  dónde se muestra cuando un dispositivo estuvo en estado apagado y cuando en estado encendido. |
| Se deberá tener registro de los dispositivos estándares que fueron convertidos a inteligentes. | testsHibernate.ConversionDispositivoPersistenciaTest |
| El sistema tiene que persistir cada uno de los sensores, junto con sus últimas N mediciones, así como también las reglas asociadas a cada sensor y los actuadores que responden frente a cada aserción por regla. |  |
| Geoposicionamiento: Se tiene que persistir en la BD el listado de transformadores activos (que nos envía el ENRE mensualmente) y la posición geográfica de cada uno. | Si. Ver el test:  *casosDePruebaMinimos.PruebaTransformadores.java*  donde se persisten los transformadores activos (que nos envía el ENRE en formato JSON) como también las zonas a la que pertenecen. |
| Reportes | |
| Consumo por hogar/período |  |
| Consumo promedio por tipo de dispositivo (inteligente o estándar) por período. |  |
| Consumo por transformador por período |  |
| Nota: Según lo mencionado por los docentes en clase, el período es un tiempo determinado que queda a elección de los alumnos (un mes, un semestre, un año, etc…) | |

## Casos de prueba (mínimos)

|  |  |  |
| --- | --- | --- |
| **CASOS DE PRUEBA (MÍNIMOS)** | | |
| **Caso de prueba** | | **Nombre del test** |
| **Nota:** Para los siguientes test la property *hibernate.hbm2ddl.auto* del archivo persistence.xml debe tener el valor *create-drop* (por defecto la subimos con esa configuración al repositorio). | | |
| 1 | Crear un nuevo usuario. Persistirlo. Recuperarlo, modificar la geolocalización y grabarlo. Recuperarlo y verificar que el cambio se haya realizado (consultando su geolocalización de vuelta). | *casosDePruebaMinimos.PruebaUsuario.java* |
| 2 | Recuperar un dispositivo. Mostrar por consola todos los intervalos que estuvo encendido durante el último mes. Modificar un atributo (por ejemplo, su nombre) y grabarlo. Recuperarlo y verificar que el cambio se haya efectivizado. | *casosDePruebaMinimos.PruebaDispositivos.java* |
| 3 | Crear una nueva regla. Asociarla a un dispositivo. Agregar condiciones y acciones. Persistirla. Recuperarla y ejecutarla. Modificar alguna condición y persistirla. Recuperarla y revisar que la condición modificada posea la última modificación. | Nota: Por como modelamos a las reglas no podríamos hacerlo, ya que no tenemos condiciones y acciones como atributos de una regla. |
| 4 | Persistir los transformadores que envía el ENRE. Recuperar todos los transformadores persistidos. Registrar la cantidad. Agregar un nuevo transformador al JSON de transformadores (que nos envía el ENRE) a mano. Ejecutar el método de lectura y persistencia. La cantidad actual de transformadores debe ser la anterior + 1. | *casosDePruebaMinimos.PruebaTransformadores.java*  **Nota:** Tal como se describe en el enunciado, para verificar el funcionamiento del test se necesita operatoria manual. Los pasos son los siguientes:   1. Correr el test mencionado y ver que la cantidad de transformadores actuales es 4 (el JSON de transformadores tiene 4 en total). Si se está corriendo el test contra una base de datos en disco (como MySQL) recomendamos hacer la consulta *SELECT \* FROM transformadores\_activos* para ver que efectivamente se están persistiendo los 4 transformadores. 2. Dirigirse al directorio *SGE\src\test\resources\data\json* y abrir el archivo *Transformadores.json*. Al final de la lista de transformadores, agregar un quinto transformador (ya sea copiando uno existente y asegurándose de cambiar su id por 5 o superior, o copiando el transformador que se encuentra en el txt *Transformador N°5.txt*). Guardar los cambios. 3. Volver a correr el test y verificar que ahora se muestra que hay 5 transformadores activos. De vuelta, si tenemos abierto MySQL Workbench, podemos volver a ejecutar la consulta *SELECT \* FROM transformadores\_activos* para visualizar que efectivamente se añadió el quinto transformador con todos sus campos. |
| **Nota:** Para los siguientes test nosotros fijamos el período a 30 días. | | |
| 5.1 | Dado un hogar y un período, mostrar por consola (interfaz de comandos) el consumo total. |  |
| 5.2 | Dado un dispositivo y un período, mostrar por consola su consumo promedio. | *casosDePruebaMinimos.ConsumoPromedioDispositivo.java* |
| 5.3 | Dado un transformador y un período, mostrar su consumo promedio. |  |
| 5.4 | Recuperar un dispositivo asociado a un hogar de ese transformador e incrementar un 1000% el consumo para ese período. Persistir el dispositivo. Volver a mostrar el consumo de ese transformador. |  |

## Justificaciones

* En algunos tests (como el test casosDePruebaMinimos.PruebaDispositivos.java, referido al caso de prueba mínimo 2) utilizamos la anotación @BeforeClass de Junit en vez de @Before. Lo hacemos porque todo método marcado con @Before corre antes de cada test dentro de la clase, mientras que un método marcado con @BeforeClass corre únicamente al principio. Como estamos accediendo a una base de datos, creando tablas, insertando, etc… hacer todo esto por cada test hace que el tiempo de los mismos se incremente, más aún si los corremos contra una base en disco como MYSQL (y no contra una en memoria). Y, además, en esos casos en particular, no es necesario que el método de inicialización corra antes de cada test (basta con que corra una única vez al principio).
* Estrategias de mapeo de herencia

## Tecnologías elegidas para la implementación

* Motor de Base de Datos: MySQL Server
* Cliente gráfico de Base de Datos: MySQL Workbench
* ORM: JPA + Hibernate
* Base de datos en memoria para las pruebas: [HSQLDB](http://hsqldb.org/)

## Otros entregables (Documentación):

* DER (subido al repositorio en Github)
* Instructivo para correr los tests contra una base de datos en MYSQL (también se encuentra subido al repositorio en Github y se adjunta en este documento)

## Instrucciones para correr tests unitarios contra MYSQL

1. Abrir el archivo persistence.xml ubicado en *SGE\src\main\resources\META-INF\*
2. Comentar las etiquetas property de la base de datos HSQLDialect (base de datos en memoria) y descomentar las referidas a MYSQL (se encuentran inmediatamente abajo). Puede hacerse rápidamente desde un IDE. Por ejemplo, en Eclipse con las opciones Add Block Comment/Remove Block Comment.
3. Conectarse a una instancia del motor de base de datos MYSQL con el usuario root y contraseña 1880. Se recomienda utilizar el cliente gráfico MYSQL Workbench para ello. Asegurarse que el servidor esté corriendo.
4. Crear una base de datos llamada *sgedb*. Esto puede hacerse ejecutando la instrucción *CREATE DATABASE sgedb.*
5. Luego de correr los tests se debe ejecutar la instrucción *USE sgedb* para poder hacer consultas a las tablas.

Nota: Si se desea utilizar otro usuario y contraseña se pueden alterar las siguientes properties en el archivo persistance.xml ya mencionado:

<property name="hibernate.connection.username" value="usuario" />

<property name="hibernate.connection.password" value="contraseña" />