Contenido

[Entrega 3 2](#_Toc528593535)

[Comentarios y justificaciones de la Entrega 3 2](#_Toc528593536)

[Casos de prueba (mínimos) 2](#_Toc528593537)

[Tecnologías elegidas para la implementación 4](#_Toc528593538)

[Otros entregables (Documentación): 4](#_Toc528593539)

# 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 |
| 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. |
| El sistema deberá persistir los dispositivos que utilizan los clientes junto con el consumo generado por cada uno. | Si |
| El sistema debe tener trazabilidad (persistir) los estados por los que paso un dispositivo. | Si (ver caso de prueba mínima número 2) |
| 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 |
| 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** |
| 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 |
| 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. |  |
| 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).

## 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)