Programación Concurrente y Distribuida

Curso 25-26

Debido a la creciente complejidad de los sistemas eléctricos modernos, basados en redes inteligentes (Smart Grids), resulta necesario desarrollar software capaz de gestionar de forma eficiente y segura múltiples eventos de consumo y suministro. En estos escenarios aparecen problemas típicos de programación concurrente y distribuida: accesos simultáneos a recursos compartidos, condiciones de sincronización, acceso a recursos remotos, etc.

En este proyecto implementaremos nuestro propio sistema simplificado de red inteligente (MySmartGrid) aplicando conceptos de Programación Concurrente y Distribuida. Obviamente no es un sistema real, pues excede el ámbito de la asignatura, pero sí modela situaciones que surgen en sistemas de monitorización y gestión energética.

MySmartGrid se va a construir paso a paso, introduciendo en cada momento algún concepto nuevo de los que vamos aprendiendo en clase. A cada paso lo denominaremos “Versión”.

En el aula virtual tienes el proyecto base que usarás para construir tu solución. En esta versión (V0) el sistema es secuencial. Se describe a continuación de forma genérica el comportamiento del sistema y luego de forma más detallada se describen las clases que lo componen.

Proyecto base (V0)

El proyecto **MySmartGrid** tiene como objetivo modelar de forma simplificada la gestión de los **consumos energéticos** que se producen en una ciudad, representada como una **red energética** (`RedEnergetica`). Esta red está organizada en distintas **zonas energéticas** (`ZonaEnergetica`), que permiten dividir el sistema y gestionar de manera independiente los consumos asociados a cada zona. Cada zona dispone de una **batería** (`Bateria`), que representa la energía disponible para atender los consumos, y de una **cuenta energética** (`CuentaEnergetica`), donde se registra el total de consumos realizados en una zona resultante de los consumos tramitados. Además, todas las zonas cuentan con un **centro de control** (`CentroControl`), encargado de simular la realización del trabajo de suministro. Cada zona tiene su centro de control.

Los **consumos energéticos** (`Consumo`) pueden ser utilizados para diferentes usos (`CanalConsumo`), como pueden ser industrial, domótico o vehículo eléctrico. Cada consumo está compuesto por una o varias **demandas** (`Demanda`), que representan peticiones concretas de

energía con determinadas características (por ejemplo, cantidad solicitada o prioridad). Estas dos clases constituyen la unidad básica de información del sistema y **su estructura debe mantenerse inalterada a lo largo del proyecto**, ya que los consumos se leen y escriben desde fichero para su posterior procesamiento. En V0, las demandas se usan únicamente para calcular el total de kWh del consumo (suma de demandas); la prioridad no afecta al procesado en esta versión.

El programa principal (`MySmartGrid`) se encarga de inicializar la red energética a partir de los parámetros definidos en la clase *Config*, leer los consumos desde un fichero y enviarlos a la **zona energética correspondiente** para su tramitación. Cada zona valida el consumo recibido y se lo envía al centro de control para su procesado. El centro de control simula la realización del trabajo (en esta versión sólo es un mensaje y una espera), suministra energía desde la batería y registra el resultado en la cuenta energética.

Para facilitar las pruebas, el proyecto incluye la clase `MainGeneraConsumos`, que permite generar ficheros de consumos de forma automática, y la clase `TicketEnergetico`, que proporciona identificadores únicos para los consumos procesados.

Esta versión base del sistema es completamente **secuencial** y está pensada como punto de partida para, en versiones posteriores, introducir progresivamente los conceptos de concurrencia y sincronización estudiados en la asignatura.

Estructura del proyecto

El proyecto está organizado en varios paquetes con diversas clases. A continuación se describen estas clases y su papel dentro del sistema.

Paquete pcd.util

- Contiene la clase `Ventana` para poder imprimir mensajes en diferentes ventanas y con diferente color.

Paquete energy

- Clase **Consumo**: almacena los datos de un consumo energético. Sus atributos son el identificador del consumo, la dirección, la zona destino, el canal de consumo y la lista de demandas asociadas. Proporciona métodos “getter” y un método para calcular el total de kWh del consumo. Incluye además un método para generar consumos de forma pseudo-aleatoria (útil para pruebas).
- Clase **Demanda**: representa una demanda dentro de un consumo (por ejemplo, un tipo de carga o servicio). Sus atributos son el identificador del tipo, los kWh solicitados y una prioridad. Proporciona métodos “getter” y un método para generar demandas de forma pseudo-aleatoria (útil para pruebas).

La estructura de estas dos clases (Consumo y Demanda) debe respetarse dado que su serialización para lectura y escritura en fichero requiere el mantenimiento de dicha estructura de datos.

La Figura 1 muestra la relación entre las diferentes clases que conforman el Proyecto.

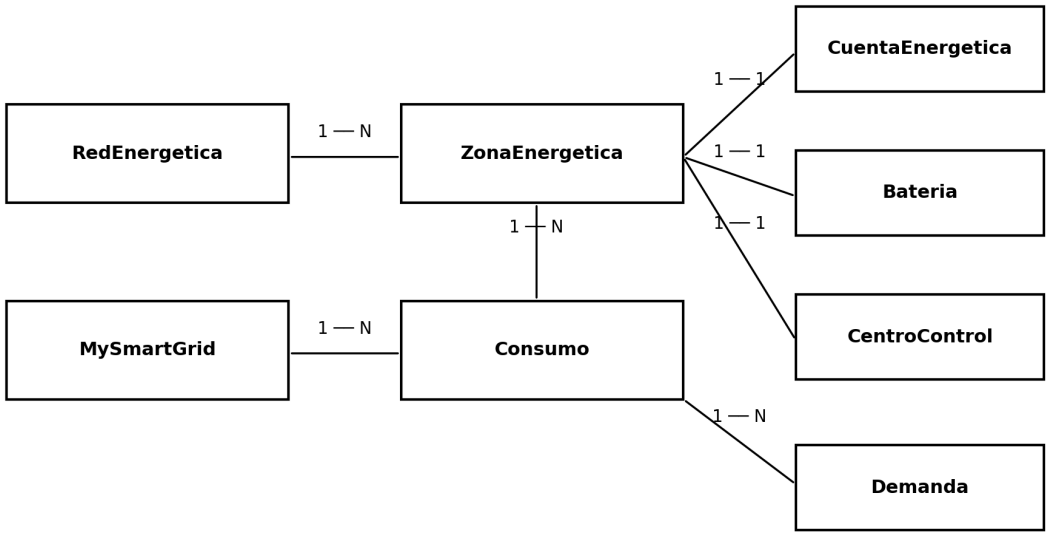

Figura 1. Relación entre las principales clases que conforman el proyecto base

- Clase **ZonaEnergetica**: representa una zona de la red. Mantiene una cuenta energética, una batería de almacenamiento y una referencia al centro de control. Ofrece el método `tramitarConsumo`, que simplemente valida que la zona es correcta y delega el trabajo sobre `CentroControl` a través del método `enviarTrabajo`. También mantiene una `Ventana` donde imprimir los mensajes relacionados con una zona concreta (esta clase no está incluida en la Figura 1).
- Clase **RedEnergetica**: crea y mantiene una lista de zonas. Proporciona métodos de auditoría para obtener el consumo anotado y la energía disponible en baterías en un momento dado, tanto a nivel individual de zona como a nivel global de la red.
- Clase **CuentaEnergetica**: asociada a una **zona energética**, en la que se registra el consumo energético derivado de los consumos tramitados en dicha zona. Mantiene la información acumulada sobre la energía suministrada y proporciona métodos para **actualizar** y **consultar** dicho balance. Esta clase permite llevar un control del consumo energético de cada zona y se utiliza posteriormente para realizar **auditorías** del sistema, tanto a nivel individual de zona como a nivel global de la red.

Paquete storage

- Clase **Batería**: modela una batería de almacenamiento energético. Contiene la capacidad máxima y el nivel actual (kWh). Proporciona métodos para comprobar si puede

suministrar energía, `suministrar` (reduciendo el nivel) y `cargar` (incrementando el nivel sin superar la capacidad). En versiones posteriores se ampliará su papel.

Paquete operators

- Clase **CentroControl**: representa un componente operador del sistema. En la versión base secuencial simula el procesado del consumo requerido mediante una espera y un mensaje. Ofrece el método `enviarTrabajo` que es el encargado de suministrar energía desde la batería y anotar el resultado en la cuenta. En versiones posteriores se ampliará su papel.

Paquete main

- Clase **MySmartGrid**: programa principal. Inicializa la red (`RedEnergetica`), lee los consumos desde fichero y, para cada consumo, localiza la `ZonaEnergetica` correspondiente y ejecuta su tramitación. Al final imprime una auditoría del estado de la red.
- Clase **MainGeneraConsumos**: programa auxiliar que permite generar un fichero de consumos para que `MySmartGrid` lo lea posteriormente. Se apoya en los generadores pseudo-aleatorios de Consumo y Demanda.
- Clase **Config**: contiene atributos estáticos y constantes de configuración (por ejemplo, número de zonas, capacidad de batería, fichero de consumos, semilla de aleatoriedad). En versiones posteriores añadiremos más variables.

Se aportan dos ficheros de ejemplo denominados `consumos2.bin` y `consumos5.bin` para que todos podamos experimentar con los mismos datos. Adicionalmente puedes crearte los tuyos usando `MainGeneraConsumos`. Para generar un fichero de consumos debes establecer en el archivo `Config` los parámetros deseados (número de zonas, nombre del fichero a generar y número de consumos a generar). También puedes cambiar las variables relacionadas con el tamaño por defecto de las ventanas que se usan o las variables relacionadas con la capacidad y nivel inicial de las baterías.

Ejecución del proyecto base

Puedes ejecutar `MySmartGrid` directamente desde el entorno de desarrollo. El programa leerá los consumos desde el fichero indicado en `Config.FICHERO_CONSUMOS`. Asegúrate antes de haber generado un fichero de consumos a través del programa `MainGeneraConsumos` o usar alguno de los proporcionados.

Siempre que ejecutes `MySmartGrid` asegúrate de tener los valores adecuados en el fichero `Config`. Si por ejemplo usas `consumos5.bin`, asegúrate de que `NUM_ZONAS` sea 5 y `FICHERO_CONSUMOS` sea "`consumos5.bin`". Si usas `consumos2.bin`, entonces `NUM_ZONAS` debe ser 2 y `FICHERO_CONSUMOS` debe ser "`consumos2.bin`".

Evolución por versiones

A partir de ahora iremos complicando el sistema poco a poco introduciendo conceptos de la asignatura (creación y coordinación de hilos, exclusión mutua y condiciones de sincronización, programación reactiva, programación distribuida, etc).