Memoria de la práctica EVCharging Network

Nombre: Elena Rocamora Bernabéu
DNI: 29576775V
Asignatura: Sistemas Distribuidos
Año: 2025/26

ÍNDICE

Descripción del proyecto

"EV*Charging Network*" consiste en el diseño e implementación de un sistema distribuido que simula la gestión de una red de puntos de recarga (Charging

Points - CP) de vehículos eléctricos. El fin de esta práctica es crear una solución que imite el funcionamiento de una infraestructura real, permitiendo la interacción las tres componentes del proyecto:

- **La Central de Control (`EV_Central`):** El "cerebro" del sistema, que monitoriza todos los CPs y gestiona las peticiones de los usuarios.
- **Los Puntos de Recarga (Engine, Monitor: `EV_CP_E`, `EV_CP_M`):** Las estaciones físicas que suministran la energía.
- **Los Conductores (`EV_Driver`):** Los usuarios finales que solicitan las recargas.

La solución implementa una arquitectura moderna de microservicios que combina múltiples paradigmas de comunicación para asegurar la resiliencia y escalabilidad:

1. **Sockets:** Se utilizan para la comunicación de estado de bajo nivel, como el registro inicial y la monitorización de "salud" entre los monitores de los CPs y la central.
2. **Streaming de Eventos (Kafka):** Se utiliza como un bus de eventos desacoplado para manejar todo el flujo de negocio: peticiones de recarga, comandos de autorización, envío de telemetría en tiempo real y entrega de tickets de facturación.
3. **Despliegue (Docker):** Todo el sistema está orquestado mediante Docker, permitiendo un despliegue modular que simula un entorno de red distribuido real.

Objetivos

El objetivo principal de esta práctica es aplicar los conceptos teóricos de la asignatura para construir un sistema distribuido robusto, comparando diferentes tecnologías de comunicación.

Los objetivos específicos han sido:

- Implementar conexiones **cliente-servidor (Sockets TCP)** para el intercambio directo y de baja latencia de mensajes de estado.
- Utilizar **Kafka** como un bus de eventos para crear una comunicación desacoplada y resiliente, capaz de gestionar los flujos de negocio (peticiones, telemetría) en tiempo real.

- Crear un sistema donde el **fallo de un componente** (como un CP) **no detenga el funcionamiento** del resto del sistema.
- Implementar un **servidor híbrido** en la `EV_Central` que combine `asyncio` para gestionar eficientemente conexiones de Sockets de E/S, y **threading** para manejar las tareas bloqueantes de los consumidores de Kafka.
- Implementar **mecanismos de detección y recuperación**, como la reconexión de consumidores de Kafka y la gestión de desconexión de Sockets.
- Utilizar **Docker Compose** para definir, construir y lanzar una aplicación multi-contenedor, gestionando redes, volúmenes y dependencias de arranque.

Arquitectura y Tecnologías clave

Se han combinado varias tecnologías, cada una cumpliendo un propósito específico dentro de la arquitectura.

Sockets

Los Sockets TCP son la base de la comunicación de estado en tiempo real. Se utilizan para el canal de comunicación "siempre activo" entre cada `EV_CP_M` (Monitor) y la `EV_Central`.

- Propósito: Autenticación y Registro (`REGISTER`) y monitorización de salud (`FAULT/HEALTHY`).
- Implementación: La `EV_Central` utiliza `asyncio.start_server` para crear un servidor de Sockets asíncrono y eficiente, capaz de manejar varias conexiones simultáneas sin necesidad de un hilo por cliente.

Streaming de Eventos (Kafka)

Kafka se utiliza como el bus de eventos principal para toda la lógica de negocio, permitiendo que los componentes estén totalmente desacoplados.

- Propósito: Gestionar flujos de datos asíncronos.
- Topics Implementados:
    - `requests`: Usado por `EV_Driver` para enviar una petición de recarga.
    - `commands`: Usado por `EV_Central` para enviar una orden (`AUTHORIZE`) a un `EV_CP_E` específico.

- `telemetry`: Usado por `EV_CP_E` para enviar telemetría en tiempo real (`CHARGING`) y el mensaje de finalización (`COMPLETED`).
- `tickets`: Usado por `EV_Central` para enviar el "recibo" final (sea `COMPLETED` o `REJECTED`) al `EV_Driver`.

Base de Datos

Para esta implementación, se ha optado por un fichero de texto plano (`cp_database.txt`) como base de datos.

- Propósito: Persistir la información básica de los CPs conocidos por la central (ID, ubicación, precio).
- Implementación: La `EV_Central` utiliza la función `load_database()` al arrancar para cargar esta información en la estructura de datos en memoria (`charging_points`), que actúa como la base de datos "en caliente" del sistema.

Docker y Docker Compose

Docker es la tecnología que permite empaquetar, distribuir y ejecutar cada componente en un contenedor aislado y reproducible.

- Dockerfile: Cada componente (Central, CPs, Drivers) tiene su propio Dockerfile que define su imagen, dependencias y entorno.
- `docker-compose.yml:` Es el “orquestador” que define todo el sistema. Define los servicios, gestiona la red virtual (`ev_net`) que los conecta, y define las dependencias de arranque (`depends_on`) para asegurar que Kafka esté listo antes de que los demás servicios arranquen (`init-kafka`).

Análisis del código

El sistema se divide en cuatro componentes software principales , tal y como se define en la arquitectura de la práctica.

1. EV_Central

Es el módulo principal que implementa la lógica de negocio y la monitorización de toda la red. Este módulo utiliza una arquitectura híbrida de concurrencia para ser lo más eficiente posible:

- Servidor de Sockets:
  - La función `main_async` inicia un servidor TCP asíncrono con `asyncio.start_server`. Esto permite manejar varias conexiones de monitores (`EV_CP_M`) en un solo hilo, siendo mucho más eficiente que el modelo de "un hilo por cliente".
  - `handle_client` se encarga de procesar los mensajes de cada monitor: `REGISTER, FAULT, HEALTHY`.

- Consumidores de Kafka:
  - Los dos consumidores (`kafka_requests_consumer` y `kafka_telemetry_consumer`) se lanzan en hilos de sistema (`threading.Thread`) separados.
  - `kafka_requests_consumer`: Escucha el topic requests. Al recibir una petición, comprueba el estado del CP en el diccionario `charging_points`. Si el CP está "Activado", envía un `AUTHORIZE` al topic `commands`; si no, envía un `REJECTED` al topic tickets.
  - `kafka_telemetry_consumer`: Escucha el topic `telemetry`. Si el status es `CHARGING`, actualiza el consumo e importe en el diccionario `charging_points` en tiempo real. Si es `COMPLETED`, envía el ticket final al topic `tickets` y resetea el estado del CP a "Activado".

- Panel de Control:
  - `display_panel` se lanza como una tarea (`asyncio.create_task`) que se refresca cada 2 segundos, leyendo de forma segura (con

db_lock) el estado del diccionario charging_points y mostrándolo por consola .

- Coordinación y Apagado:
    - Un threading.Lock (db_lock) se utiliza para sincronizar el acceso al diccionario charging_points entre los hilos de Kafka.
    - El sistema captura las señales SIGINT/SIGTERM para un apagado limpio, usando un threading.Event para detener los hilos de Kafka y un asyncio.Event para detener el servidor asyncio.

2. EV_CP_M

Este módulo es un cliente de sockets dual que actúa como el "guardián" de un EV_CP_E (Engine).

- Socket a EV_Central: Al arrancar, se conecta a la Central y se registra enviando un mensaje REGISTER#{cp_id}#{location}.
- Socket a EV_CP_E: Entra en un bucle while true que se ejecuta cada segundo. En cada iteración, se conecta a su EV_CP_E local y le envía un mensaje HEALTH_CHECK.
- Lógica de Estado:
    - Si el EV_CP_E responde OK, el monitor envía HEALTHY#{cp_id} a la central.
    - Si el EV_CP_E no responde (error de socket) o responde KO, el monitor envía FAULT#{cp_id} a la central. Es el responsable de que el panel muestre "Averiado" o "Activado".

3. EV_CP_E

Simula el hardware del punto de recarga y es un componente que actúa como servidor y cliente a la vez.

- Servidor de sockets: Inicia un servidor de sockets en su main para escuchar las conexiones de su EV_CP_M. Responde OK o KO a los HEALTH_CHECK.
- Simulación de avería: Lanza un hilo (input_thread) que permite al operador simular una avería (tecla 'a') o una reparación (tecla 'r'), cambiando la variable is_faulty que se usa para responder al monitor.
- Consumidor de Kafka: Lanza un hilo (kafka_commands_consumer) que escucha el topic commands. Al recibir un mensaje AUTHORIZE, inicia la simulación de recarga (charging_simulation_thread).

- Productor de Kafka: Durante la simulación (`charging_simulation_thread`), envía telemetría (status: `'CHARGING'`) cada segundo al topic `telemetry`. Al finalizar, envía un mensaje status: `'COMPLETED'`.

4. EV_Driver

Simula la aplicación del usuario final y es el iniciador del flujo de negocio.

- Lector de Ficheros: Lee un fichero de peticiones para simular un conductor que quiere realizar varias recargas.
- Productor de Kafka: Por cada `cpId` en el fichero, publica un mensaje de petición en el topic `requests`.
- Consumidor de Kafka: Tras enviar una petición, entra en un bucle de espera y se suscribe al topic `tickets`.
- Lógica de Tickets: Para evitar "tickets fantasma" de ejecuciones anteriores, el consumidor usa `'auto.offset.reset': 'earliest'` pero filtra activamente los mensajes, aceptando solo el ticket cuyo `cpId` coincide con el que acaba de solicitar.
- Flujo: Al recibir el ticket (sea `COMPLETED` o `REJECTED`), espera 4 segundos (`time.sleep(4)`) y pasa a la siguiente petición del fichero.

Guía de despliegue

Prerrequisitos:

- Docker Desktop instalado y en ejecución.
- El código fuente completo del proyecto.
- Una terminal.

1. Limpieza

Antes de cada ejecución, se recomienda limpiar los contenedores y volúmenes de Kafka para asegurar un inicio desde cero, sin mensajes "fantasma" en las colas.

```
# Desde la carpeta 'Deploy'
docker-compose down -v
```

2. Terminal 0: Kafka

```
# Desde la carpeta 'Deploy'
docker-compose up -d broker init-kafka
```

3. Terminal 1: EV Central

```
# Desde la carpeta 'Central'
python EV_Central.py 6001 127.0.0.1:9092
```

4. Terminal 2: Engine

```
# Desde la carpeta 'ChargingPoints'
python EV_CP_E.py 7000 127.0.0.1:9092 CP001
```

5. Terminal 3: Monitor

```
# Desde la carpeta 'ChargingPoints'
python EV_CP_M.py 127.0.0.1 6001 CP001 127.0.0.1 7000
```

6. Terminal 4: Driver

```
# Desde la carpeta 'Driver'
python EV_Driver.py 127.0.0.1:9092 D001 request.txt
```

Capturas demostrativas (del punto 2-6)

```
C:\Users\Elena\Desktop\1º cuatri\SD\Practicas\practica2\Deploy>docker-compose up -d broker init-kafka
unable to get image 'apache/kafka:latest': error during connect: Get "http://%2F%2F.%2Fpipe%2FdockerDesktopLinuxEngine/v
1.51/images/apache/kafka:latest/json": open //./pipe/dockerDesktopLinuxEngine: El sistema no puede encontrar el archivo
especificado.

C:\Users\Elena\Desktop\1º cuatri\SD\Practicas\practica2\Deploy>docker-compose up -d broker init-kafka
[+] Running 2/2
 ⠿ Container broker      Healthy                                                                    36.2s
 ⠿ Container init-kafka  Started                                                                     0.5s
```

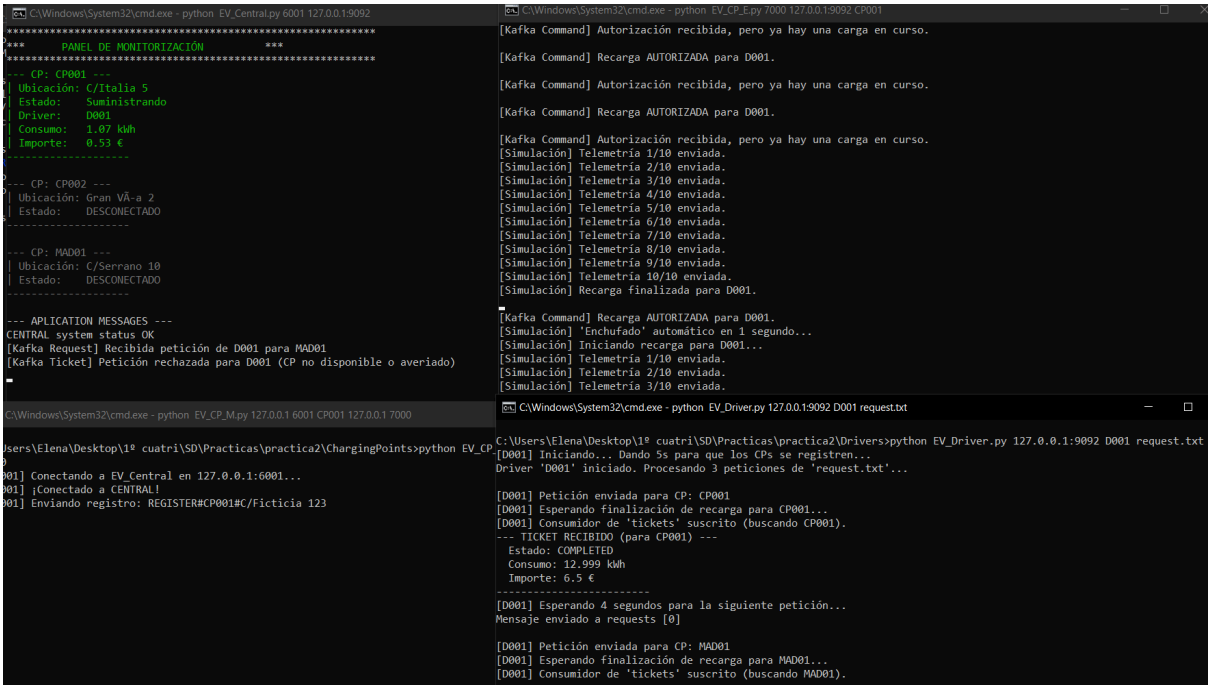
```
************************************************************
***      PANEL DE MONITORIZACIÓN          ***
************************************************************
--- CP: CP001 ---
| Ubicación: C/Italia 5
| Estado:    DESCONECTADO
--------------------

--- CP: CP002 ---
| Ubicación: Gran VÃ-a 2
| Estado:    DESCONECTADO
--------------------

--- CP: MAD01 ---
| Ubicación: C/Serrano 10
| Estado:    DESCONECTADO
--------------------

--- APLICATION MESSAGES ---
CENTRAL system status OK
```

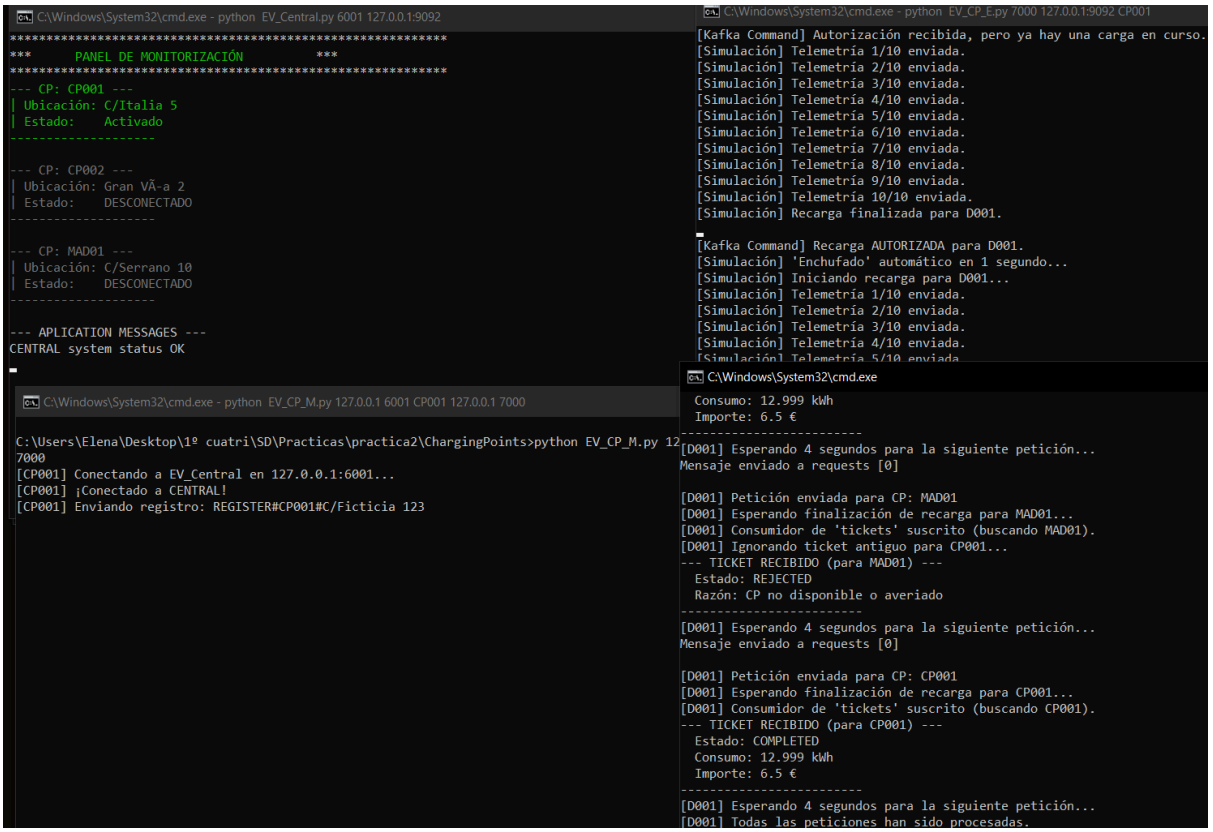
```
C:\Windows\System32\cmd.exe - python  EV_Central.py 6001 127.0.0.1:9092
************************************************************
***      PANEL DE MONITORIZACIÓN          ***
************************************************************
--- CP: CP001 ---
| Ubicación: C/Italia 5
| Estado:    Suministrando
| Driver:    D001
| Consumo:   3.70 kWh
| Importe:   1.85 €
--------------------

--- CP: CP002 ---
| Ubicación: Gran VÃ-a 2
| Estado:    DESCONECTADO
--------------------

--- CP: MAD01 ---
| Ubicación: C/Serrano 10
| Estado:    DESCONECTADO
--------------------

--- APLICATION MESSAGES ---
CENTRAL system status OK
```

```
C:\Windows\System32\cmd.exe - python  EV_CP_E.py 7000 127.0.0.1:9092 CP001
[Simulación] 'Enchufado' automático en 1 segundo...
[Simulación] Iniciando recarga para D001...

[Kafka Command] Recarga AUTORIZADA para D001.

[Kafka Command] Autorización recibida, pero ya hay una carga en curso.

[Kafka Command] Recarga AUTORIZADA para D001.

[Kafka Command] Autorización recibida, pero ya hay una carga en curso.

[Kafka Command] Recarga AUTORIZADA para D001.

[Kafka Command] Autorización recibida, pero ya hay una carga en curso.

[Kafka Command] Recarga AUTORIZADA para D001.

[Kafka Command] Autorización recibida, pero ya hay una carga en curso.

[Kafka Command] Recarga AUTORIZADA para D001.

[Kafka Command] Autorización recibida, pero ya hay una carga en curso.
[Simulación] Telemetría 1/10 enviada.
```

```
C:\Windows\System32\cmd.exe - python  EV_Central.py 6001 127.0.0.1:9092
************************************************************
***      PANEL DE MONITORIZACIÓN          ***
************************************************************
--- CP: CP001 ---
| Ubicación: C/Italia 5
| Estado:    Activado
--------------------

--- CP: CP002 ---
| Ubicación: Gran VÃ-a 2
| Estado:    DESCONECTADO
--------------------

--- CP: MAD01 ---
| Ubicación: C/Serrano 10
| Estado:    DESCONECTADO
--------------------

--- APLICATION MESSAGES ---
CENTRAL system status OK
```

```
C:\Windows\System32\cmd.exe - python  EV_CP_E.py 7000 127.0.0.1:9092 CP001
[Kafka Command] Recarga AUTORIZADA para D001.

[Kafka Command] Autorización recibida, pero ya hay una carga en curso.

[Kafka Command] Recarga AUTORIZADA para D001.

[Kafka Command] Autorización recibida, pero ya hay una carga en curso.

[Kafka Command] Recarga AUTORIZADA para D001.

[Kafka Command] Autorización recibida, pero ya hay una carga en curso.

[Kafka Command] Recarga AUTORIZADA para D001.

[Kafka Command] Autorización recibida, pero ya hay una carga en curso.
[Simulación] Telemetría 1/10 enviada.
[Simulación] Telemetría 2/10 enviada.
[Simulación] Telemetría 3/10 enviada.
[Simulación] Telemetría 4/10 enviada.
[Simulación] Telemetría 5/10 enviada.
[Simulación] Telemetría 6/10 enviada.
[Simulación] Telemetría 7/10 enviada.
[Simulación] Telemetría 8/10 enviada.
[Simulación] Telemetría 9/10 enviada.
[Simulación] Telemetría 10/10 enviada.
[Simulación] Recarga finalizada para D001.
```


C:\Windows\System32\cmd.exe - python EV_Central.py 6001 127.0.0.1:9092
*** PANEL DE MONITORIZACIÓN ***
--- CP: CP001 ---
| Ubicación: C/Italia 5
| Estado: Suministrando
| Driver: D001
| Consumo: 1.07 kWh
| Importe: 0.53 €
--- CP: CP002 ---
| Ubicación: Gran VÃ-a 2
| Estado: DESCONECTADO
--- CP: MAD01 ---
| Ubicación: C/Serrano 10
| Estado: DESCONECTADO
--- APLICATION MESSAGES ---
CENTRAL system status OK
[Kafka Request] Recibida petición de D001 para MAD01
[Kafka Ticket] Petición rechazada para D001 (CP no disponible o averiado)
C:\Windows\System32\cmd.exe - python EV_CP_E.py 7000 127.0.0.1:9092 CP001
[Kafka Command] Autorización recibida, pero ya hay una carga en curso.
[Kafka Command] Recarga AUTORIZADA para D001.
[Kafka Command] Autorización recibida, pero ya hay una carga en curso.
[Kafka Command] Recarga AUTORIZADA para D001.
[Kafka Command] Autorización recibida, pero ya hay una carga en curso.
[Simulación] Telemetría 1/10 enviada.
[Simulación] Telemetría 2/10 enviada.
[Simulación] Telemetría 3/10 enviada.
[Simulación] Telemetría 4/10 enviada.
[Simulación] Telemetría 5/10 enviada.
[Simulación] Telemetría 6/10 enviada.
[Simulación] Telemetría 7/10 enviada.
[Simulación] Telemetría 8/10 enviada.
[Simulación] Telemetría 9/10 enviada.
[Simulación] Telemetría 10/10 enviada.
[Simulación] Recarga finalizada para D001.
[Kafka Command] Recarga AUTORIZADA para D001.
[Simulación] 'Enchufado' automático en 1 segundo...
[Simulación] Iniciando recarga para D001...
[Simulación] Telemetría 1/10 enviada.
[Simulación] Telemetría 2/10 enviada.
[Simulación] Telemetría 3/10 enviada.
C:\Windows\System32\cmd.exe - python EV_CP_M.py 127.0.0.1 6001 CP001 127.0.0.1 7000
Users\Elena\Desktop\1º cuatri\SD\Practicas\practica2\ChargingPoints>python EV_CP
001] Conectando a EV_Central en 127.0.0.1:6001...
001] ¡Conectado a CENTRAL!
001] Enviando registro: REGISTER#CP001#C/Ficticia 123
C:\Windows\System32\cmd.exe - python EV_Driver.py 127.0.0.1:9092 D001 request.txt
C:\Users\Elena\Desktop\1º cuatri\SD\Practicas\practica2\Drivers>python EV_Driver.py 127.0.0.1:9092 D001 request.txt
[D001] Iniciando... Dando 5s para que los CPs se registren...
Driver 'D001' iniciado. Procesando 3 peticiones de 'request.txt'...
[D001] Petición enviada para CP: CP001
[D001] Esperando finalización de recarga para CP001...
[D001] Consumidor de 'tickets' suscrito (buscando CP001).
--- TICKET RECIBIDO (para CP001) ---
Estado: COMPLETED
Consumo: 12.999 kWh
Importe: 6.5 €
[D001] Esperando 4 segundos para la siguiente petición...
Mensaje enviado a requests [0]
[D001] Petición enviada para CP: MAD01
[D001] Esperando finalización de recarga para MAD01...
[D001] Consumidor de 'tickets' suscrito (buscando MAD01).


C:\Windows\System32\cmd.exe - python EV_Central.py 6001 127.0.0.1:9092
*** PANEL DE MONITORIZACIÓN ***
--- CP: CP001 ---
| Ubicación: C/Italia 5
| Estado: Activado
--- CP: CP002 ---
| Ubicación: Gran VÃ-a 2
| Estado: DESCONECTADO
--- CP: MAD01 ---
| Ubicación: C/Serrano 10
| Estado: DESCONECTADO
--- APLICATION MESSAGES ---
CENTRAL system status OK
C:\Windows\System32\cmd.exe - python EV_CP_E.py 7000 127.0.0.1:9092 CP001
[Kafka Command] Autorización recibida, pero ya hay una carga en curso.
[Simulación] Telemetría 1/10 enviada.
[Simulación] Telemetría 2/10 enviada.
[Simulación] Telemetría 3/10 enviada.
[Simulación] Telemetría 4/10 enviada.
[Simulación] Telemetría 5/10 enviada.
[Simulación] Telemetría 6/10 enviada.
[Simulación] Telemetría 7/10 enviada.
[Simulación] Telemetría 8/10 enviada.
[Simulación] Telemetría 9/10 enviada.
[Simulación] Telemetría 10/10 enviada.
[Simulación] Recarga finalizada para D001.
[Kafka Command] Recarga AUTORIZADA para D001.
[Simulación] 'Enchufado' automático en 1 segundo...
[Simulación] Iniciando recarga para D001...
[Simulación] Telemetría 1/10 enviada.
[Simulación] Telemetría 2/10 enviada.
[Simulación] Telemetría 3/10 enviada.
[Simulación] Telemetría 4/10 enviada.
C:\Windows\System32\cmd.exe - python EV_CP_M.py 127.0.0.1 6001 CP001 127.0.0.1 7000
C:\Users\Elena\Desktop\1º cuatri\SD\Practicas\practica2\ChargingPoints>python EV_CP_M.py 127.0.0.1 6001 CP001 127.0.0.1 7000
[CP001] Conectando a EV_Central en 127.0.0.1:6001...
[CP001] ¡Conectado a CENTRAL!
[CP001] Enviando registro: REGISTER#CP001#C/Ficticia 123
C:\Windows\System32\cmd.exe
Consumo: 12.999 kWh
Importe: 6.5 €
[D001] Esperando 4 segundos para la siguiente petición...
Mensaje enviado a requests [0]
[D001] Petición enviada para CP: MAD01
[D001] Esperando finalización de recarga para MAD01...
[D001] Consumidor de 'tickets' suscrito (buscando MAD01).
[D001] Ignorando ticket antiguo para CP001...
--- TICKET RECIBIDO (para MAD01) ---
Estado: REJECTED
Razón: CP no disponible o averiado
[D001] Esperando 4 segundos para la siguiente petición...
Mensaje enviado a requests [0]
[D001] Petición enviada para CP: CP001
[D001] Esperando finalización de recarga para CP001...
[D001] Consumidor de 'tickets' suscrito (buscando CP001).
--- TICKET RECIBIDO (para CP001) ---
Estado: COMPLETED
Consumo: 12.999 kWh
Importe: 6.5 €
[D001] Esperando 4 segundos para la siguiente petición...
[D001] Todas las peticiones han sido procesadas.