SISTEMAS DISTRIBUIDOS

Universitat d'Alacant
Universidad de Alicante

Práctica 1. EVCharging Network. Sockets, Streaming de Eventos, Colas y modularidad.

Alumnos:
Iván Soler Sánchez 74530257E
Guillermo Candela Gabaldón 74019773T

Curso 2025-2026
Grupo 06 - 6

Objetivos

El objetivo principal de esta práctica es extender y afianzar nuestros conocimientos sobre los sockets y otras arquitecturas orientadas al streaming de eventos en tiempo real basados en sockets pero de términos de construcción y desarrollo mucho más elevados, como puede ser Kafka.

Componentes software

Dada la naturaleza de la práctica, hemos decidido realizarla en el lenguaje de programación Python, puesto que lo hemos considerado lo más simple , lo más adecuado para el tiempo de desarrollo disponible y la flexibilidad de la práctica y que ambos teníamos cierto conocimiento previo del lenguaje de programación.

A la hora de guardar los distintos datos del software como pueden ser los diferentes CPs hemos decidido usar MongoDB dada su flexibilidad, puesto que el modelo de la base de datos no estaba fijo desde el inicio e iba a sufrir modificaciones.

Como se nos dice en la práctica hemos implementado 3 bloques principales de para nuestro software: core, cp y drivers. Aparte de estos 3 bloques tenemos varios archivos que utilizan varios bloques para llevar a cabo un código mucho más claro y sencillo, comenzaremos tratando esos archivos.

Archivos generales

Para un desarrollo más fácil hemos decidido utilizar un entorno virtual en Python para poder desplegar e instalar todas las dependencias de las librerías sin ningún problema. Tenemos estos archivos comunes:

- **Utils.py**: para dar color a los diferentes tipos de salidas por terminal y representar el formato adecuado para la práctica.
- **topics.py**: para dar un formato más simple a los topics a la hora de programar, además de tenerlos enumerados y explicados.
- **socketCommunication.py**: archivo con las funciones y variables para la codificación de la comunicación por sockets.
- **kafka_utils.py**: archivo con las funciones necesarias para la comunicación con kafka.

Apartados

Core

Como bien indica su nombre, es el core de la aplicación. En este apartado tenemos la Central y la base de datos. Central gestiona la comunicación tanto usando varias colas de Kafka como con sockets, se comunica con los Charging Points, verifica su estado en la base de datos, etc.
También es importante recalcar el uso de **Docker** para el despliegue de este bloque. En la raíz del proyecto contamos con un archivo ***compose.yml*** para levantar una imagen de Kafka y Mongo, ambos previamente configurados.
En estes apartado también hemos creado varios archivos para estandarizar y facilitar el despliegue de la aplicación:

- **create_topics.sh/create_topics.bat**: archivo para crear los tópicos de kafka, tanto para Linux como para Windows.
- **db_init.py**: archivo para inicializar la base de datos, con CPs iniciales, etc.
- **db.py:** archivo con las funciones necesarias para acceder y manejar correctamente la base de datos.

Cp

En esta carpeta guardamos todo lo relacionado con los Charging Points: Monitor y Engine. Los monitores sirven para monitorizar el funcionamiento del Charging Point, enviando un mensaje de comprobación del estado de salud a Engine constantemente.
Los Engines implementan toda la lógica de funcionamiento del charging point. Es decir, se comunican con Central, permiten que se conecten los drivers, etc.

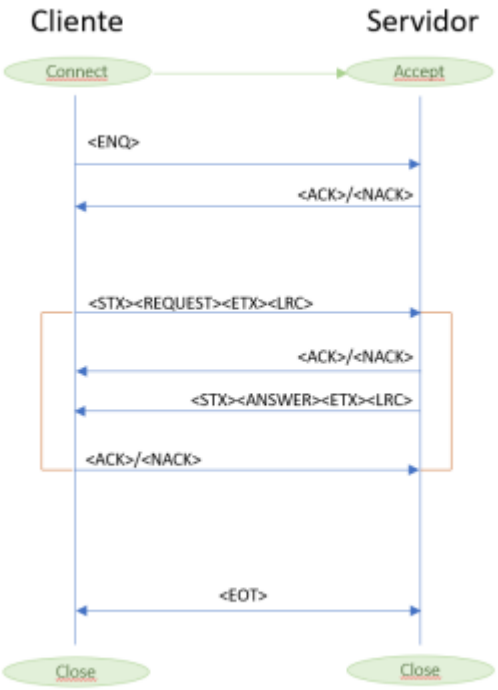
Drivers

En este apartado guardamos lo relacionado con Drivers. Se trata de los propios conductores que se conectan a los Charging Points. Estos conductores solicitarán suministro a través de un archivo el cual contendrá las distintas acciones que el driver realizará durante la ejecución del programa, y se les concederá o no dependiendo del estado del Charging Point.

Justificación de diseño

Sockets

A la hora de definir el protocolo de comunicación de los sockets hemos decidido seguir el que se nos recomendaba en la práctica ya que se nos explicaba cómo debía ser la comunicación y nos facilitaba el desarrollo de un protocolo:



Esta comunicación empieza con el cliente mandando un <ENQ>, el servidor responderá con un <ACK> si el mensaje ha sido recibido correctamente o con un <NACK> si ha ocurrido algún error. Si el cliente recibe <ACK> comienza a mandar <REQUEST> acompañado junto con <STX>, <ETX> y con <LRC>, este último se usa para confirmar que el mensaje ha sido enviado correctamente. El servidor responderá con <ACK>/<NACK> y enviará <ANSWER> contestando a <REQUEST>, este mensaje se enviará de la misma manera que se envió <REQUEST>. Por último, se concluye la comunicación con <EOT>.

Kafka

Para la gestión de colas a través de Kafka hemos decidido implementar las siguientes colas:

```
EV_SUPPLY_REQUEST = "evSupplyRequest"       # Driver -> Central (solicitud de recarga)

EV_SUPPLY_AUTH_DRI = "evSupplyAuthDri"      # Central -> Driver (autorización)
EV_SUPPLY_STARTED = "evSupplyStarted"       # Central -> Driver (Comienza el supply driver)
EV_SUPPLY_TICKET = "evSupplyTicket"         # Central -> Driver (ticket final con precio)

EV_SUPPLY_CONNECTED = "evSupplyConnected" # Engine -> Central (Comienza el supply engine)
EV_SUPPLY_AUTH = "evSupplyAuth" # Central -> Engine  (iniciar o rechazar)
EV_SUPPLY_END = "evSupplyEnd"    # Engine -> Central (fin del supply, trigger para que central mande el ticket)
EV_SUPPLY_HEARTBEAT = "evSupplyHeartbeat" # Engine -> Central (monitorizar estado)
EV_CENTRAL_HEARTBEAT = "evCentralHeartbeat" # Driver -> Central (latido para monitorizar estado)
EV_DRIVER_SUPPLY_HEARTBEAT = "evDriverSupplyHeartbeat" # Driver -> Central (monitorizar estado)
EV_DRIVER_SUPPLY_ERROR = "evDriverSupplyError" # Driver -> Central (error durante el suministro)
```

A continuación explicaremos la función de cada cola y el porqué de su implementación.

EV_SUPPLY_REQUEST -> Esta cola se usa para cuando un driver solicita una recarga en un Charging Point. Una vez se recibe se confirma que dicho CP existe y está operativo, en caso de estarlo se añade el driver a la base de datos, comunica Driver con Central.

EV_SUPPLY_AUTH-DRI -> Si la solicitud es correcta se Central informa al Driver de ello.

EV_SUPPLY_AUTH -> Si la solicitud del driver es correcta se usa esta cola para que Central envíe la solicitud al Engine.

EV_SUPPLY_CONNECTED -> Mediante esta cola Engine mandará a Central que la solicitud del Driver ha sido aceptada.

EV_SUPPLY_STARTED -> Central manda a Driver que el supply ha comenzado siempre y cuando el CP no esté en uso, en caso de estar en uso también se mandará a Driver.

EV_SUPPLY_END -> Engine comunica a Central que se ha acabado el suministro.

EV_SUPPLY_TICKET -> Una vez se haya acabado el supply Central manda a Engine el ticket de su recargar.

EV_SUPPLY_HEARTBEAT -> Engine manda mensajes a Central para monitorear el estado de Central durante el supply.

EV_CENTRAL_HEARTBEAT -> Driver manda mensajes a Central para comprobar que esta esté operativa.

EV_DRIVER_SUPPLY_HEARTBEAT -> Driver manda mensajes a Central para comprobar que esta esté operativo durante el supply.

EV_DRIVER_SUPPLY_ERROR -> Driver informa de un error durante el supply a Central.

Guía de despliegue

Para ejecutar la aplicación en 3 ordenadores diferentes, ejecutaremos 1 apartado por ordenador (Core, Cp y Drivers).

El primer paso será levantar el Core. Para ello seguimos los siguientes pasos:

- En la raíz del proyecto, tendremos el archivo compose.yml, en este archivo tendremos que poner la IP de nuestra máquina:

```
# Lo que anuncia a los clientes externos (pon aquí tu IP LAN del host)
KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://localhost:9092
```

  Tras esto tenemos que activar docker en el ordenador y ejecutar el comando "***docker compose up"*** para levantar las imágenes de kafka y mongo.
- Tras acabar de levantarlas, crearemos el *virtual environment en la raíz del proyecto* de esta manera: ***python3 -m venv venv.*** I
- Una vez creado hay que proceder a activarlo, la forma de activarlo va a depender de si estamos en Windows o en Linux. En Linux: **"source /venv/bin/activate"**. En Windows: **"venv\Scripts\activate"**
- Tras activarlos instalamos los requerimientos: ***pip install -r requirements.txt***
- Cambiamos la IP de Kafka en el archivo de create_topics por la nuestra:

```
  --bootstrap-server localhost:9092 \
```

- y ejecutamos: ***./create_topics.sh*** o ***./create_topics.bat***
- Inicializamos la base de datos: ***python3 evcharging/core/db_init.py***
- Ejecutamos Central desde la raíz del proyecto: ***python3 -m evcharging.core.EV_CENTRAL <puerto> <kafka_ip:port>***

Tras esto ya habríamos levantado la central, y procederemos a levantar los charging points, en un terminal haremos lo siguiente para levantar el Engine:

- Creamos el *virtual environment* de esta manera: ***python3 -m venv venv***
- Lo activamos con uno de los dos comandos explicaods anteriormente: **“source /venv/bin/activate”** o **“venv\Scripts\activate”**
- Instalamos los requerimientos: ***pip install -r requirements.txt***
- Ejecutamos el engine desde la raíz del proyecto: ***python3 -m evcharging.cp.EV_CP_E <kafka_ip> <kafka_port> <socket_port>***

Desde otra terminal volvemos a crear y activar el virtual environment como hemos hecho en el paso anterior y lanzamos el siguiente comando para lanzar el Monitor:

- Ejecutamos el monitor desde la raíz del proyecto: ***python3 -m evcharging.cp.EV_CP_M <ipCentral> <portCentral> <ipEngine> <portEngine> <CP_ID>***

Por último, ejecutamos los Drivers:

- Creamos el *virtual environment* de esta manera: ***python3 -m venv venv.***
- Lo activamos como habíamos hecho antes en los CPs.
- Instalamos los requerimientos: ***pip install -r requirements.txt***
- Ejecutamos el driver desde la raíz del proyecto: ***python3 -m evcharging.drivers.EV_Driver <kafkaIp:KafkaPort> <clientId> <fileName>***

Funcionamiento

Para mostrar el funcionamiento vamos a seguir los pasos que hemos detallada en la **"Guía de despliegue"**:
Una vez modificado el archivo compose.yml poniendo nuestra IP lanzamos el contenedor:

```
PS D:\SD-Pract1> docker compose up
[+] Running 21/21
 ✔mongo Pulled                                                    156.7s
   ✔ce8dccf1f4c1 Pull complete                                    153.7s
   ✔74522aa03d00 Pull complete                                      0.9s
   ✔5d6f7cf8e15d Pull complete                                     37.3s
   ✔f90aba9940a7 Pull complete                                     37.4s
   ✔34baad995762 Pull complete                                      1.5s
   ✔a4ada45c862e Pull complete                                      1.5s
   ✔4b3ffd8ccb52 Pull complete                                     37.0s
   ✔5d3652024acd Pull complete                                      1.6s
 ✔broker Pulled                                                   92.6s
   ✔28bd55152645 Pull complete                                      0.3s
   ✔f18232174bc9 Pull complete                                      5.7s
   ✔2fa1f65d07a3 Pull complete                                      1.3s
   ✔35e38a4b206f Pull complete                                      1.3s
   ✔729fc64ae8c1 Pull complete                                     49.7s
   ✔b8eeb529f1af Pull complete                                     89.2s
   ✔a723193c2f26 Pull complete                                     89.4s
   ✔73851e29d6a7 Pull complete                                      1.4s
   ✔643bf8a7c247 Pull complete                                      1.4s
   ✔b42f712acf6d Pull complete                                     88.9s
   ✔c3f73af09931 Pull complete                                     24.2s
[+] Running 3/3
 ✔Network sd-pract1_default  Created                               0.1s
 ✔Container mongo            Created                               3.7s
 ✔Container broker           Created                               3.7s
Attaching to broker, mongo
```

A continuación vamos a crear y activar el virtual environment:

```
PS D:\SD-Pract1> python -m venv venv
PS D:\SD-Pract1> .\venv\Scripts\activate
(venv) PS D:\SD-Pract1>
```

Instalamos las dependencias con el comando "pip3 install -r .\requirements.txt"

Ahora tenemos que crear las colas de kafka con el script "create_topics.sh/bat", en mi caso será el .bat ya que estoy en Windows.

Tras esto, inicializamos la base de datos lanzando el script db_init.py localizado en el módulo core.

Con todo esto preparado lanzamos EV_Central:

```
(venv) PS D:\SD-Pract1> python -m evcharging.core.EV_Central 7000 127.0.0.1:9092
[CENTRAL] En escucha permanente en 7000

=== Charging Points en Sistema ===
ID: cp-001 | State: DISCONNECTED | Price: 1.53 eur | Location: UA-Lab
ID: cp-002 | State: DISCONNECTED | Price: 2.0 eur | Location: Luceros
ID: cp-003 | State: DISCONNECTED | Price: 0.28 eur | Location: SanJuan
ID: cp-004 | State: DISCONNECTED | Price: 2.11 eur | Location: TheOne
ID: cp-005 | State: DISCONNECTED | Price: 1.13 eur | Location: SantaPola
ID: cp-006 | State: DISCONNECTED | Price: 1.98 eur | Location: Elche
[CENTRAL] Esperando mensajes ...
```

Lanzamos EV_E:

```
(venv) PS D:\SD-Pract1> python -m evcharging.cp.EV_CP_E 127.0.0.1 9092 7100
[ENGINE] Esperando monitor en puerto 7100 ...
```

Lanzamos EV_M:

```
(venv) PS D:\SD-Pract1> python -m evcharging.cp.EV_CP_M 127.0.0.1 7000 127.0.0.1 7100 cp-001
[MONITOR] Conectado a CENTRAL (127.0.0.1:7000)
[MONITOR] ✅ CP validado por CENTRAL
[MONITOR] Conectado a ENGINE (127.0.0.1:7100)
[MONITOR] ✅ CP registrado en ENGINE
[MONITOR] Conexión con ENGINE: OK
```

Y lanzamos driver con este archivo para solicitar supplies:

driver.txt

```
1  cp-001
2  cp-011
3  cp-002
```

```
(venv) PS D:\SD-Pract1> python -m evcharging.drivers.EV_Driver 127.0.0.1:9092 ivan .\driver.txt
[DRIVER ivan] Esperando mensajes Kafka ...
[DRIVER ivan] CENTRAL está activa
[DRIVER ivan] Solicito recarga en cp-001
[DRIVER ivan] Autorizado para recargar en cp-001 (CP disponible)
```

Se autoriza el supply desde engine:

```
(env) PS D:\SD-Pract1> python -m evcharging.cp.EV_CP_E 127.0.0.1 9092 7100
[ENGINE] Esperando monitor en puerto 7100 ...
[ENGINE] CP registrado: cp-001
[ENGINE cp-001] Escuchando mensajes de CENTRAL ...
[ENGINE cp-001] Registrado, esperando suministro/heartbeat
[ENGINE] Autorización aprobada para driver ivan
[ENGINE] Iniciando suministro para driver ivan
Ingrese 'FIN' para finalizar el suministro cuando desee.
```

Mensajes de central:

```
[CENTRAL] Driver ivan solicita recarga en CP cp-001
[CENTRAL] Recarga autorizada para Driver ivan en CP cp-001
[CENTRAL] Notificación enviada a Driver ivan de confirmación
[CENTRAL] Driver ivan consumiendo en CP cp-001: Power: 0.71 kW, Energy: 0.001 kWh
[CENTRAL] Driver ivan consumiendo en CP cp-001: Power: 1.08 kW, Energy: 0.001 kWh
[CENTRAL] Driver ivan solicita recarga en CP cp-001
[CENTRAL] Recarga denegada para Driver ivan en CP cp-001: CP no disponible
[CENTRAL] Driver ivan consumiendo en CP cp-001: Power: 1.28 kW, Energy: 0.003 kWh
[CENTRAL] Driver ivan consumiendo en CP cp-001: Power: 1.59 kW, Energy: 0.004 kWh
[CENTRAL] Driver ivan consumiendo en CP cp-001: Power: 2.25 kW, Energy: 0.006 kWh
[CENTRAL] Driver ivan consumiendo en CP cp-001: Power: 1.45 kW, Energy: 0.007 kWh
[CENTRAL] Driver ivan consumiendo en CP cp-001: Power: 1.99 kW, Energy: 0.009 kWh
[CENTRAL] Driver ivan consumiendo en CP cp-001: Power: 1.5 kW, Energy: 0.01 kWh
[CENTRAL] Driver ivan consumiendo en CP cp-001: Power: 1.06 kW, Energy: 0.011 kWh
```

Mensaje de engine:

```
(venv) PS D:\SD-Pract1> python -m evcharging.cp.EV_CP_E 127.0.0.1 9092 7100
[ENGINE] Esperando monitor en puerto 7100 ...
[ENGINE] CP registrado: cp-001
[ENGINE cp-001] Escuchando mensajes de CENTRAL ...
[ENGINE cp-001] Registrado, esperando suministro/heartbeat
[ENGINE] Autorización aprobada para driver ivan
[ENGINE] Iniciando suministro para driver ivan
Ingrese 'FIN' para finalizar el suministro cuando desee.
FIN
[ENGINE] Suministro finalizado para driver ivan
```

Mensaje de driver:

```
[DRIVER ivan] CENTRAL está activa
[DRIVER ivan] Solicito recarga en cp-001
[DRIVER ivan] Autorizado para recargar en cp-001 (CP disponible)
[DRIVER ivan] Carga iniciada en cp-001
[DRIVER ivan Consumiendo en cp-001: Power: 1.46 kW, Energy: 0.001 kWh at 22:30:57
[DRIVER ivan Consumiendo en cp-001: Power: 1.32 kW, Energy: 0.002 kWh at 22:31:01
[DRIVER ivan Consumiendo en cp-001: Power: 2.28 kW, Energy: 0.004 kWh at 22:31:04
[DRIVER ivan Consumiendo en cp-001: Power: 2.44 kW, Energy: 0.006 kWh at 22:31:07
[DRIVER ivan Consumiendo en cp-001: Power: 2.24 kW, Energy: 0.008 kWh at 22:31:10
[DRIVER ivan Consumiendo en cp-001: Power: 0.93 kW, Energy: 0.009 kWh at 22:31:13
[DRIVER ivan Consumiendo en cp-001: Power: 1.9 kW, Energy: 0.01 kWh at 22:31:16
[DRIVER ivan Consumiendo en cp-001: Power: 1.98 kW, Energy: 0.012 kWh at 22:31:19
[DRIVER ivan Consumiendo en cp-001: Power: 1.09 kW, Energy: 0.013 kWh at 22:31:22
[DRIVER ivan Consumiendo en cp-001: Power: 1.75 kW, Energy: 0.014 kWh at 22:31:25
[DRIVER ivan] Ticket recibido de cp-001: 0.020000 EUR
[DRIVER ivan] Ticket recibido para cp-001
[DRIVER ivan] Solicito recarga en cp-011
[DRIVER ivan] Recarga DENEGADA en cp-011: CP no disponible
[DRIVER ivan] No se recibió ticket para cp-011, intento 1
[DRIVER ivan] Recarga DENEGADA en cp-011: CP no disponible
[DRIVER ivan] No se recibió ticket para cp-011, intento 2
[DRIVER ivan] No se pudo recibir ticket para cp-011 tras 2 intentos, pasando al siguiente CP.
[DRIVER ivan] Solicito recarga en cp-002
[DRIVER ivan] Recarga DENEGADA en cp-002: CP no disponible
```

Estos fallan porque el cp-002 no está activo y el cp-011 no existe.

Más mensajes de central:

```
[CENTRAL] Driver ivan solicita recarga en CP cp-011
[CENTRAL] Recarga denegada para Driver ivan en CP cp-011: CP no existe
[CENTRAL] Recarga denegada para Driver ivan en CP cp-011: CP no disponible
[CENTRAL] Driver ivan solicita recarga en CP cp-011
[CENTRAL] Recarga denegada para Driver ivan en CP cp-011: CP no existe
[CENTRAL] Recarga denegada para Driver ivan en CP cp-011: CP no disponible
[CENTRAL] Driver ivan solicita recarga en CP cp-002
[CENTRAL] Recarga denegada para Driver ivan en CP cp-002: CP no disponible
[CENTRAL] Driver ivan solicita recarga en CP cp-002
[CENTRAL] Recarga denegada para Driver ivan en CP cp-002: CP no disponible
```