SD	**Sistemas Distribuidos**
24/25	Práctica no guiada: Sockets, Streaming de Eventos, Colas y modularidad.
	EasyCab

Preámbulo

El objetivo de esta práctica es que los estudiantes extiendan y afiancen sus conocimientos sobre el uso de la tecnología de comunicación básica sockets, estudiada durante las sesiones de teoría.

Los sockets son la base de las comunicaciones entre computadores y suponen el punto de enlace básico entre nuestra aplicación e Internet, utilizando muy pocos recursos de red. Como contrapartida, las aplicaciones que utilizan sockets como mecanismo de comunicación deben interpretar los mensajes que intercambian entre sí, es decir, deben establecer un protocolo o acuerdo de comunicación entre ellos. Esta necesidad implica un fuerte acoplamiento entre las aplicaciones distribuidas que utilizan sockets como mecanismo de comunicación, ya que todas las partes deben conocer de antemano la estructura de los mensajes que serán intercambiados y codificarlo en su programación.

Por otro lado, otras tecnologías y arquitecturas actuales orientadas al streaming de eventos en tiempo real, basadas en sockets pero de más alto nivel en términos de construcción y desarrollo, permiten la implementación de soluciones menos acopladas y más resilientes que estos.

Esta práctica establece el marco inicial de un desarrollo que se ampliará durante el cuatrimestre y que permitirá al estudiante crear una aplicación similar a las que se desarrollan hoy en día en los entornos empresariales, poniendo el foco en el uso e integración de distintos paradigmas de comunicación susceptibles de ser utilizados.

Especificación

El objetivo de la práctica a desarrollar es un sistema distribuido que implemente una simulación de una solución para la gestión de un servicio de taxis de conducción autónoma.

Descripción funcional

Los estudiantes deberán implementar una solución de control online en tiempo real denominada "**EasyCab**".

Alcance

La solución tendrá como objetivo realizar un sistema que simula la gestión de una flota de taxis con conducción autónoma.

Imagen 1: Flota de taxis autónomos en China

Nuestra simulación consistirá en, partiendo de un mapa de una ciudad ficticia, dispondremos de una serie de vehículos autonomos que recibirán instrucciones desde una central para ir de un punto A de esa ciudad al punto B de acuerdo a las peticiones que se van recibiendo de los clientes que quieran usar el servicio de taxi.

Los vehículos, recorrerán la ciudad, enviando constantemente su posición a la central de control la cual, ante cualquier contingencia, podrá tomar decisiones para hacer que el vehículo se pare, cambie su destino o vuelva a la base.

Una vez recorrida su ruta los vehículos esperarán nuevas instrucciones de la central (prestar un nuevo servicio a otro cliente, volver a la central, …)

En el estado inicial, los vehículos partirán de la localización donde se ubica la central de control.

Para la realización de todo el sistema, nos basaremos en una serie de componentes distribuidos que estarán interconectados y que conformarán la solución de manera resiliente, escalable y segura.

Mapa

La cuidad será representada por una simple matriz 2D de 20x20 posiciones. Cada posición del mapa (elemento de la matriz) puede contener una de las siguientes posibilidades:

1- Un taxi: Representado por su identificador (número) y un color (verde si está moviéndose hacia su posición final, rojo si está parado).
2- Localizaciones: Destinos para los que se solicita el servicio. Representado por un carácter en mayúsculas con fondo AZUL.
3- Nada: Representado por una celda sin contenido.
4- Cliente: Representado por un carácter en minúsculas con un fondo Amarillo.

Todos las aplicaciones que representan los taxis así como el motor central que los gobierna deberán estar recibiendo y visualizando en todo momento el mapa completo en su estado actual.

El mapa representa una geometría esférica de manera que las posiciones más al este (límite derecha del mapa) están conectadas con las situadas al oeste y viceversa, de la misma forma que las situadas en el límite norte (zona superior del mapa) conectan con el lado sur y viceversa.

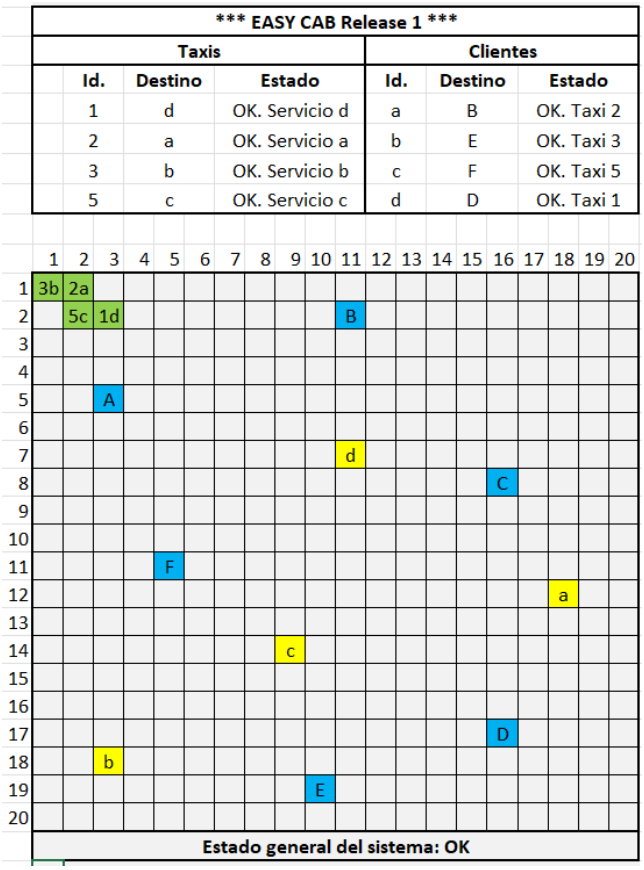


Figura 1: Ejemplo de mapa con localizaciones, clientes y taxis en situación inicial.

Taxis

Los taxis estarán representados por una aplicación distribuida autónoma.

Disponen de los siguientes atributos:

- ID: Identificador del mismo. Numérico [0..99]-Se usará para representarlo en el mapa.
- Estado: Representado por un color. En movimiento = Verde, en posición final o parado= Rojo.
- Posición: Coordenadas del mapa en las que se encuentra: { int [1..20], int[1..20] }

Cada taxi se moverá por el espacio retransmitiendo su posición cada vez que cambie de coordenada. Como si se tratará de un espacio real, solo se puede mover a su siguente coordenada adyacente en cualquier dirección.

El taxi dispondrá de una serie de sensores (lidars, sensores de proximidad, gps,…) que alertarán en todo momento al cerebro central (Digital Engine) del vehículo de cualquier anomalía en el recorrido (semáforo en rojo, persona cruzando, vehículo cercano,…)

Customers

Los clientes del servicio dispondrán de una aplicación que permitirá solicitar un servicio de taxi para ir de un sitio a otro. Así, la aplicación enviará a la central la petición de un servicio de un ciudadano para ir al punto X.

El punto de recogida será donde se encuentre el propio cliente.

Mecánica de la solución

El sistema comienza mediante la ejecución del servidor central (en adelante “CENTRAL”) que permanecerá en ejecución sin límite de tiempo.

El sistema realizará entonces la siguiente secuencia:

1- CENTRAL leerá un fichero con la configuración del mapa de la ciudad:
   El formato de fichero será:
   <ID_LOCALIZACION><COORDENADA_X ><COORDENADA_Y >
   <ID_LOCALIZACION><COORDENADA_X ><COORDENADA_Y >
   …
2- CENTRAL dispondrá de una BD o fichero con los taxis disponibles.
3- Los clientes, desde su aplicación, solicitarán un primer servicio de taxi. El servicio se solicatará tan solo indicando a que destino se quiere ir. Con el objetivo de automatizar las pruebas del sistema, los servicios estarán contenidos en un fichero con el siguiente formato:
   <ID_LOCALIZACION>
   <ID_LOCALIZACION>
   <ID_LOCALIZACION>
   …

4- CENTRAL procederá a consultar si tiene algún taxi libre y, de ser así, enviará un mensaje al usuario notificando la aceptación de servicio (OK) y otro al taxi para que vaya a recoger al cliente que solicite el servicio y lo transporte al destino deseado. En caso de no poder asignar el servicio enviará una denegación del mismo al cliente (KO). Dichos mensajes **se deben mostrar claramente en pantalla, tanto en la aplicaión del cliente como de CENTRAL.**

5- Los taxis, según reciben instrucciones, se irán moviendo progresivamente a su dirección de destino **notificando cada movimiento que realizan**. Este estado "RUN" del taxi se represantará en pantalla mediante la expresión en color VERDE de ese taxi.

6- Durante el trayecto el cerebro digital (Digital Engine) del taxi **irá recibiendo información de los sensores** que indicarán tanto un estado correcto (ok) como cualquier contigencia (obstáculo al frente, semáforo en rojo, peatón cruzando, …) que producirá la parada inmediata del vehículo y alertará a CENTRAL de dicha situación. Los sensores **estarán enviando un mensaje cada segundo** al Digital Engine. En caso de que la contingencia se resuelva, los sensores enviarán un nuevo mensaje de OK y el taxi reanudará su viaje. Toda la información tiene que ser claramente visible en pantalla.

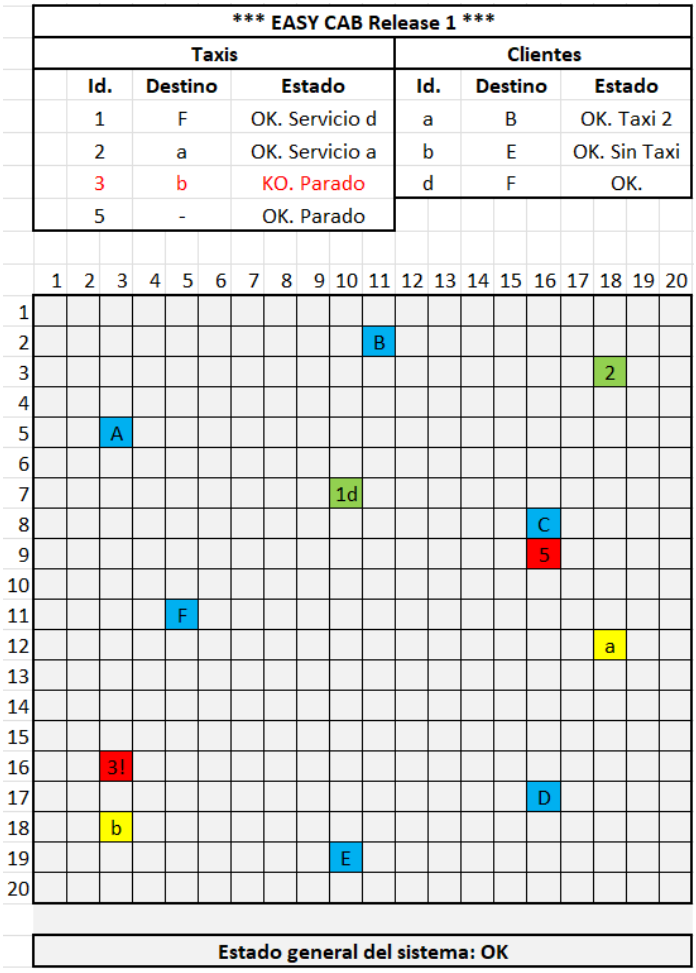


Figura 2: Servicios en curso.

7- CENTRAL recibirá todos los movimientos de cada taxi y volverá a emitir el estado de situación de todo el mapa que todos los taxis recibirán a su vez.

8- Cuando un taxi llegue a su destino deberá permanecer en él cambiando su estado a "END" que se reflejará en pantalla mediante la expresión en color ROJO de ese taxi. **Solo cuando reciba nuevas órdenes para ir a otro destino o para volver a la base deberá iniciar nuevamente su movimiento**.

9- Cuando un servicio concluya, se notificará al cliente la finalización del mismo y, si dicho cliente precisa de otro servicio (tiene más registros en su fichero) **esperará 4 segundos y procederá a solicitar un nuevo servicio.**

10- CENTRAL permanecerá siempre en funcionamiento a la espera de nuevas peticiones de servicios. Adicionalmente dispondrá de opciones para, de forma arbitraria, poder decirle a un vehículo cualquiera de estas opciones:
    a. Parar: El taxi parará donde esté y pondrá su estado en ROJO.
    b. Reanudar: El taxi reanudará el servicio y pondrá su estado en VERDE.
    c. Ir a Destino: El taxi cambiará su destino por el que se indica en esta petición.
    d. Volver a la base: Ir al destino [1,1]

Las coordenadas de inicio de todos los taxis en el momento de arrancar serán la [1,1] y se mueven hacia su destino con un movimiento de 1 coordenada adyacente cada vez.

La acción se va desarrollando de forma asíncrona, es decir, sin "turnos", de manera que cada cliente envía servicios en cualquier momento y cada taxi se mueve en cualquier momento.

Los taxis pueden moverse en cualquier dirección: N,S,W,E,NW,NE,SW,SE asumiendo que hay carreteras en cualquier dirección.

En cada movimiento se recalcula la posición del taxi. Si un taxi falla o su aplicación se bloquea por cualquier causa es eliminado visualmente de la acción. Si se recupera debe volver a autenticarse y volver a moverse a su destino según le indique CENTRAL.

Diseño técnico

Se propone al estudiante implementar un sistema distribuido compuesto, al menos, de los siguientes componentes software:

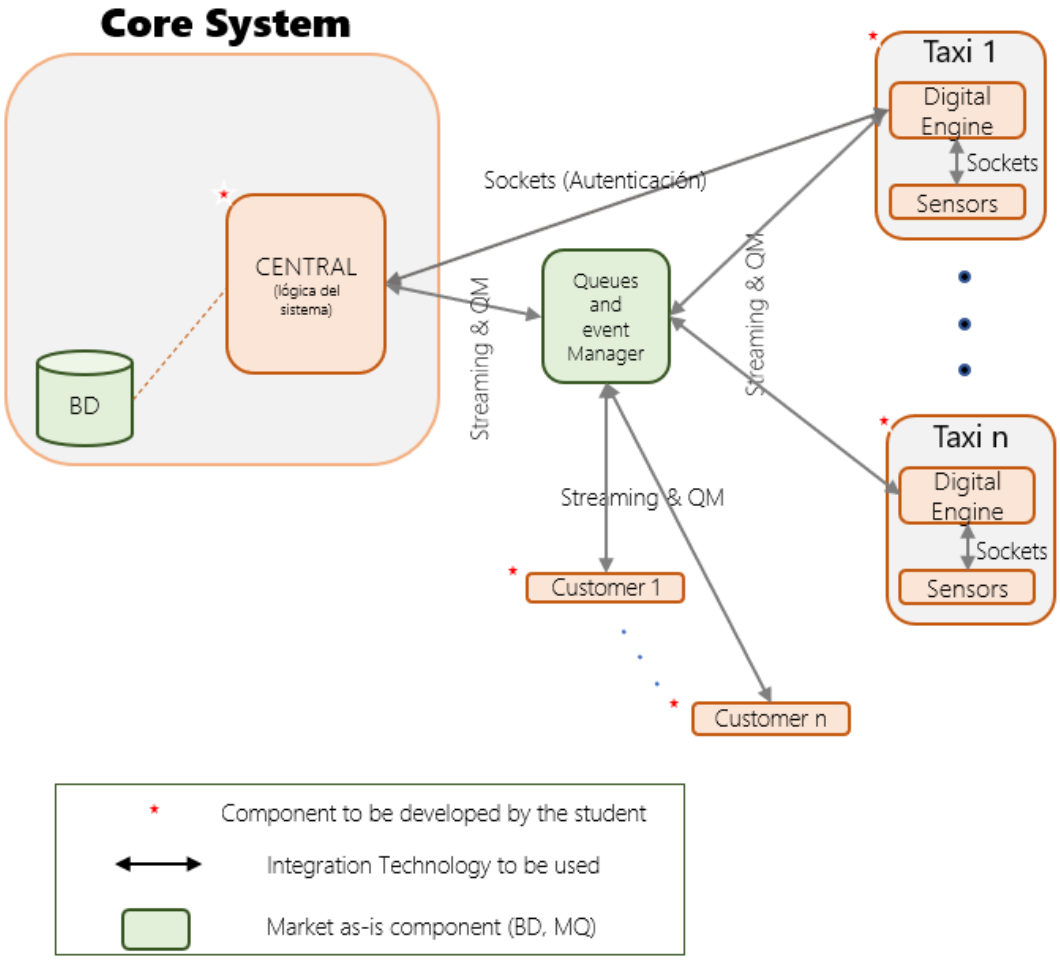


Figura 3. Arquitectura conceptual del Sistema software, interconexiones entre los componentes y tipos de interfaces. Release 1

Los componentes software que el estudiante ha de desarrollar son los siguientes:

- Core System: Módulos centrales de la solución
  - CENTRAL: Módulo que representa la central de control del servicio de taxis e implementa la lógica y gobierno de todo el sistema.

- Taxi: Módulos que simulan el taxi
  - Digital Engine: Cerebro del vehículo que recibe la información de los sensores y se conecta al sistema central.
  - Sensores: Dispositivos que informan al Digital Engine de cualquier incidencia durante su recorrido.

- Customer: Aplicación que usan los clientes para solicitar un servicio de taxi.

Los componentes pueden ser desarrollados en el lenguaje de preferencia del estudiante: Java, C/C++, . NET, Python, etc. asumiendo el estudiante la responsabilidad de su conocimiento y forma de desplegarlo en el laboratorio.

El subsistema de gestión de streaming de eventos será implementado mediante la tecnología KAFKA cuyo funcionamiento se explicará en las sesiones de prácticas.

A continuación, se especifica más detalladamente cada componente.

CORE

Contiene todos los módulos que implementan la estructura principal del sistema.

CENTRAL:

Se trata de la aplicación que implementará la lógica fundamental del sistema. El nombre de la aplicación será **obligatoriamente "EC_Central".** Para su ejecución recibirá por la línea de parámetros los siguientes argumentos:

- Puerto de escucha
- IP y puerto del Broker/Bootstrap-server del gestor de colas.
- IP y puerto de la BBDD (solo en caso de que sea necesario)

Al arrancar la aplicación quedará a la escucha en el puerto establecido e implementará todas las funcionalidades necesarias para ejecutar la mecánica de la solución expresada en el apartado anterior incluidas las indicadas en el punto 10 del apartado "Mecánica de la solución".

Para mayor sencillez de implementación, ante cada movimiento de cada taxi le responderá con todo el mapa de situación.

Los estudiantes podrán decidir la forma de expresar el mapa aunque se recomienda un array de bytes donde cada elemento de la matriz de bytes es una posición en del mapa.

Base de Datos:

Contendrá, los datos de los taxis (Id, estado,…) así cualquier otra información que los estudiantes consideren necesaria para la implementación del sistema.

Los estudiantes podrán decidir el motor de BD a implementar recomendando los profesores los siguientes: SQlite, MySQL, SQLServer o MongoDB. **Igualmente será posible usar simples ficheros para la implementación.**

TAXI

Digital Engine:

Se trata de la aplicación que implementará la lógica principal de todo el sistema. El nombre de la aplicación será **obligatoriamente "EC_DE".** Para su ejecución recibirá por la línea de parámetros los siguientes argumentos:

- IP y puerto del EC_Central.
- IP y puerto del Broker/Bootstrap-server del gestor de colas.
- IP y Puerto EC_S.
- ID del taxi.

Al arrancar la aplicación, esta se conectará al EC_Central para autenticarse y validar que el vehículo está operativo y preparado para prestar servicios. La aplicación permanecerá a la espera hasta recibir una solicitud de servicio procedente de la central. **El taxi quedará asignado con el ID recibido en la línea de parámetros y cuya validez deberá ser confirmada por EC_Central en el proceso de autenticación.**

Sensors:

Aplicación que simula la funcionalidad de los sensores embarcados en el vehículo y que velan por la seguridad y correcto funcionamiento de este sin incidencias. El nombre de la aplicación será **obligatoriamente "EC_S".** Para su ejecución recibirá por la línea de parámetros los siguientes argumentos:

- IP y puerto del EC_DE

Al arrancar la aplicación esta se conectará al Digital Engine del vehículo que corresponda y empezará a enviarle cada segundo un mensaje de estado, bien para indicar que todo es correcto (OK), bien para indicar que hay una incidencia en el camino y debe pararse (KO). Para simular dichas incidencias, la aplicación deberá permitir que, en tiempo de ejecución, se pulse una tecla para provocar dicha incidencia.

Customers

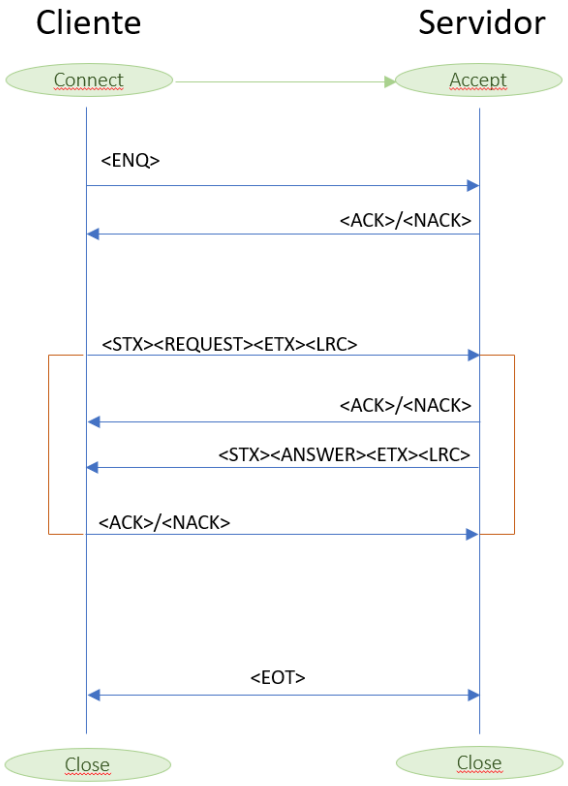
Aplicación que disponen los usuarios del sistema para solicitar un servicio. El nombre de la aplicación será **obligatoriamente "EC_Customer".** Para su ejecución recibirá por la línea de parámetros los siguientes argumentos:

- IP y puerto del Broker/Bootstrap-server del gestor de colas.
- ID del cliente

Esta aplicación leerá de un fichero todos los servicios que va a solicitar. Enviará el primero a EC_Central y, tras su conclusión (sea en éxito o fracaso), esperará 4 segundos y pasará a solicitar el siguiente servicio.

Protocolo de comunicación (solo sockets)

Como se comenta al principio de este documento, la comunicación mediante sockets obliga a los actores involucrados a definir la mensajería y protocolo con precisión. Los estudiantes podrán definir los protocolos de comunicación entre los distintos módulos del sistema a su criterio. **Los profesores recomiendan, aunque no es obligatorio,** usar una mensajería basada en el estándar de empaquetado <STX><DATA><ETX><LRC> que corresponde al siguiente flujo**:**



Donde:

- <REQUEST> y <ANSWER>: Contienen el mensaje transmitido entre ambos puntos con los campos separados por un carácter determinado:
    - Ej.: <REQUEST>: Código Operación#campo1#...#campo n

- <LRC>: Se define como el XOR(MESSAGE) byte a byte y sirve para validar que la transmisión del mensaje se ha realizado satisfactoriamente y ha sido recibida de forma correcta por el destinatario el cual responderá al mismo con un <ACK> o <NACK>

Aclaraciones finales

Antes incongruencias funcionales o aspectos no considerados en los anteriores apartados que impidan la correcta implementación de la práctica, los estudiantes deberán implementar la alternativa que entiendan más conveniente previa consulta con el profesor para consensuar dicha implementación evitando un excesivo nivel de complejidad.

Guía mínima de despliegue

Para la correcta evaluación de la práctica es necesario comprobar que la aplicación distribuida solicitada es desplegada en un entorno verdaderamente distribuido. Es por ello por lo que para su prueba es necesario al menos 3 PCs distintos en los que se desplegarán los componentes solicitados. Al menos se ha de desplegar el siguiente escenario. **El uso de la tecnología Docker para el despliegue es aconsejable y se valorará positivamente**, aunque no estrictamente obligatorio:

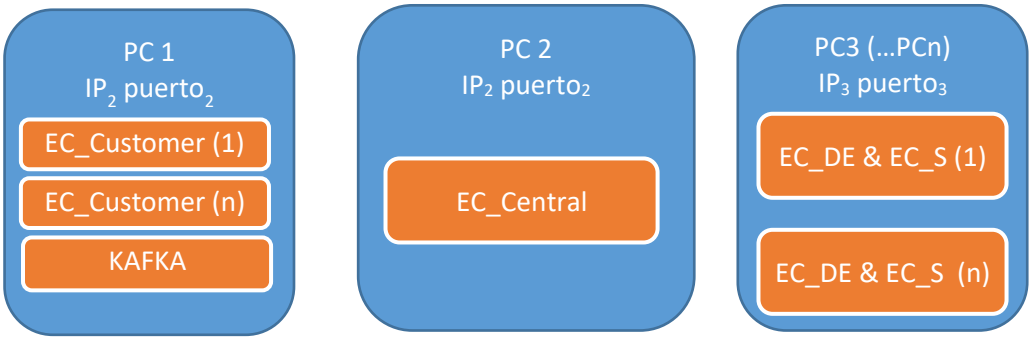


Figura 4. Escenario físico mínimo para el despliegue de la práctica.

Entregables y evaluación

La evaluación de la práctica se realizará en los laboratorios. **Se podrá realizar en grupos de hasta 2 personas sin perjuicio de que, durante el momento de la corrección, el profesor pueda preguntar a cualquier estudiante del grupo por cualquier aspecto de cualquiera de los módulos**. Los estudiantes deben desplegar por ellos mismos la práctica que resuelve el enunciado anterior. Deben desplegar un sistema completo, es decir, una central , los taxis (digital engine y sensores y clientes todos ellos interconectados entre sí. **Este requisito es indispensable para poder realizar la corrección**. Además, deben poderse evaluar positiva o negativamente todos los apartados que aparecerán en la Guía de corrección que se entregará a tal propósito. Cada uno de los apartados puntúa de forma variable, por tanto, cada apartado no implementado o que no pueda comprobarse su correcto funcionamiento no podrá ser tenido en cuenta y por tanto no puntuará. Los estudiantes deberán presentar para la evaluación el documento **"Guía de corrección"** cumplimentado para que el profesor pueda validar los apartados implementados.

Los estudiantes deberán entregar, además, mediante la funcionalidad de evaluación del UACloud antes de la fecha establecida a su profesor de prácticas una **memoria de prácticas**, con el código fuente y compilados generados, así como un documento donde se detalle la siguiente información. El formato es libre, pero debe ser un documento ordenado y debidamente formateado, cuidando la redacción y ortografía.

- Portada con los nombres, apellidos y DNI de los estudiantes, año académico y el título de la práctica.
- Un informe donde se indique el nombre de los componentes software desarrollados y una descripción de cada uno de ellos, explicando y enviando además el código fuente de todos ellos.

- El detalle, paso a paso, de una guía de despliegue de la aplicación, que deberá ser la misma que utilice cuando haga la corrección de la práctica.
- Capturas de pantalla que muestren el funcionamiento de las distintas aplicaciones conectadas.

Cada profesor de prácticas podrá solicitar a los estudiantes cualquier otra evidencia que el profesor considere adecuada para poder formalizar la evaluación.

La fecha de entrega será en la semana del 21/10/2024.