

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

## 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 autónomos 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 ciudad 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.

\*\*\* EASY CAB Release 1 \*\*\*

Taxis			Clientes		
Id.	Destino	Estado	Id.	Destino	Estado
1	d	OK. Servicio d	a	B	OK. Taxi 2
2	a	OK. Servicio a	b	E	OK. Taxi 3
3	b	OK. Servicio b	c	F	OK. Taxi 5
5	c	OK. Servicio c	d	D	OK. Taxi 1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	3b	2a																		
2		5c	1d								B									
3																				
4																				
5			A																	
6																				
7											d									
8																	C			
9																				
10																				
11					F															
12																		a		
13																				
14										c										
15																				
16																				
17																		D		
18			b																	
19											E									
20																				

Estado general del sistema: OK

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 siguiente 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 solicitará 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 aplicación 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 representará 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 contingencia (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.

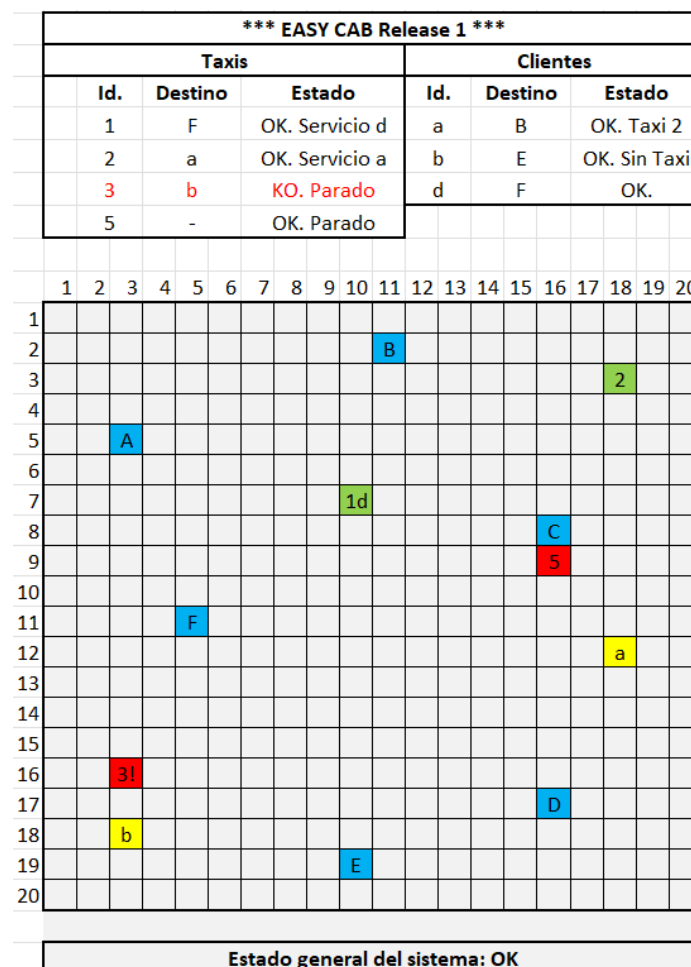


Figura 2: Servicios en curso.

- 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.



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

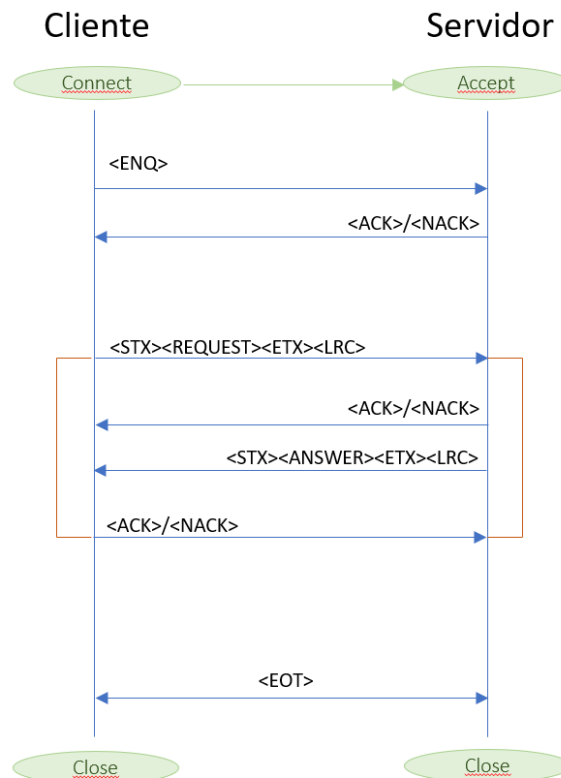
Aplicación que disponen los usuarios del sistema para solicitar un servicio. El nombre de la aplicación será **obligatoriamente "EC\_Customer"**. Para su ejecución recibirá por la línea de parámetros los siguientes argumentos:

- 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:
  - o 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:

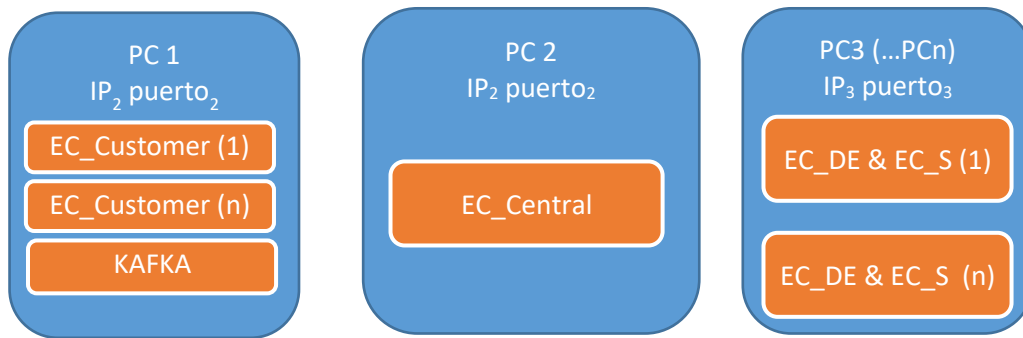


Figura 4. Escenario físico mínimo para el despliegue de la práctica.

## 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.**

<b>SD</b>	<b>Sistemas Distribuidos</b>
<b>24/25</b>	<b>Práctica no guiada: Seguridad y API Rest</b>
	<b>EasyCab</b>

## Preámbulo

El objetivo de esta práctica es que los estudiantes implementen, por un lado, dentro de los principios de arquitectura SOA (Service Oriented Architecture), la tecnología de comunicación de basada en servicios REST y por otro algunos de los principios de seguridad vistos en teoría.

Para ello, se consumirán una serie API Rest ya establecidos por un tercero y, por otro, se creará y expondrá otro API Rest desde un back que sea consumido por un front.

Adicionalmente, se implementarán tres aspectos relacionados con la seguridad: cifrado de canal, autenticación segura y principios de auditoría.

Se partirá de la misma práctica ya generada en la primera parte de la asignatura con la misma funcionalidad expuesta.

# Especificación

## Descripción funcional

La funcionalidad que deberá implementarse será idéntica a la implementada en la release 1 con las siguientes modificaciones:

- 1- EC\_Central deberá comunicarse con un módulo externo EC\_CTC (City Traffic Control) que proporcionarán la viabilidad o no de circular por la ciudad donde se presta el servicio.
- 2- Se añadirá un nuevo módulo llamado EC\_CTC (City Traffic Control) el cual determinará el estado del tráfico. Para ello se comunicará vía API con sistemas externos (ChatGPT y Openweather) como se detalla en los siguientes apartados.
- 3- Se añadirá un nuevo módulo llamado EC\_Registry que permitirá registrar un nuevo Taxi o darlo de baja. Este módulo se comunicará con los Taxis vía API\_Rest.
- 4- Se creará un Front (mediante una simple página web) la cual mostrará el estado del mapa a cualquiera que la invoque. Es, por tanto, una web pública.
- 5- Se implementarán los siguientes mecanismos de seguridad:
  - Autenticación segura entre los Taxis y el Registry: cifrado del canal y protección segura de credenciales.
  - Auditoría de eventos en la Central.
  - Cifrado de los datos entre la Central y los Taxis.

## Diseño técnico

Partiendo del diseño técnico ya implementado en la primera parte de la práctica, se modificará y ampliará el mismo para contemplar la siguiente arquitectura:

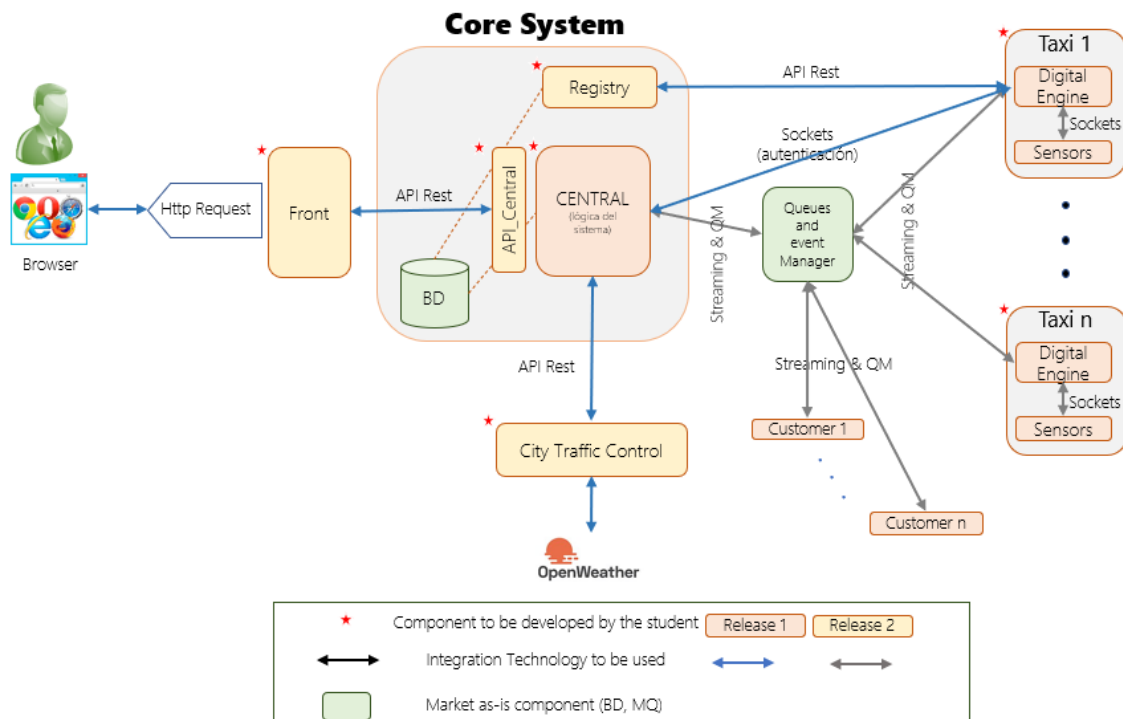


Figura 1. Esquema conceptual del Sistema software, interconexiones entre los componentes y tipos de interfaces.

Al igual que en la primera parte de la práctica, 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.

## Core System

### Central:

Será idéntico al ya realizado en la primera parte con las siguientes modificaciones:

- Consumo de API\_rest de un gestor de tráfico de la ciudad:** EC\_Central se conectará a un nuevo módulo llamado EC\_CTC (City Traffic Control) el cual le informará si el tráfico en la ciudad representada por el mapa es viable o no. Para ello, EC\_Central, **cada 10 segundos** consumirá un API que ofrecerá EC\_CTC el cual devolverá OK si la situación en la ciudad permite el tráfico o KO en caso de que la circulación no sea viable. Más adelante se indica el funcionamiento de EC\_CTC.
- Implementación de la autenticación entre EC\_Central y los Taxis:** En el momento de la autenticación, la central devolverá al taxi un token el cual deberá ser usado

por este en todos los mensajes que envíe a la Central. La central deberá comprobar que dicho token está en vigor antes de responder al taxi. Los tokens no tendrán caducidad temporal pero **sí expirarán cuando el taxi reciba una orden de la central de volver a la base (posición 1,1) y desconectarse**. Los tokens **NO podrán ser almacenados por el taxi ni tampoco de forma permanente en la BD**. Si se realiza almacenamiento en la BD será temporal y deberá quedar totalmente limpia de tokens una vez hayan expirado. Tal y como se refleja en el diagrama de arquitectura conceptual, la autenticación seguirá siendo por sockets pero el alumno que lo desee podrá implementar un API Rest entre Taxi y Central a tal propósito lo que, en la práctica profesional sería más correcto. En este caso Central expondría un API de Autenticación que el Taxi consumiría.

- 3- **Implementación de cifrado en el canal entre EC\_Central y los Taxis:** Se establecerá un sistema de cifrado del canal que los Taxis depositan en los Topics de las colas para evitar ataques del tipo MITM (Man In The Middle). Se realizará un sencillo sistema de cifrado con intercambio de claves y certificados (simétrico o asimétrico) elegible por el estudiante con su justificación adecuada. *Nota: Kafka dispone de 3 componentes que permiten implementar mecanismos de seguridad a este propósito. Dichos componentes son: Cifrado de datos SSL/TLS, Autenticación SSL o SASL y Autorización mediante ACL. No se exige al estudiante la incorporación en la práctica de estos componentes si bien aquellos que lo deseen pueden hacerlo siendo valorado positivamente.*



- 4- Implementará un **registro de auditoría de todos los eventos que sucedan**, indicando, de forma estructurada, los datos de dicho evento como se indica a continuación:
- Fecha y hora del evento.
  - Quién y desde dónde se produce el evento: IP de la máquina que genera el evento (ej. IP del taxi, IP del Usuario)
  - Qué acción se realiza: Autenticación o intentos de autenticación fallidos o no, Incidencias durante el servicio, cambio de la situación del tráfico, usuarios o taxis bloqueados, errores, etc.
  - Parámetros o descripción del evento.

\*Se deberá incluir un API REST que exponga la información de la auditoría de seguridad. Se debe incluir un Front Web que acceda a dicha API a través de un navegador y la visualice de manera amigable. Se valorará la actualización en la parte del cliente (navegador Web) en tiempo real.

Nota informativa: Los sistemas profesionales deben incorporar estos conceptos de auditoría mediante herramientas específicas que integran un SIEM (Security Information and Event Management).

### API\_Central:

Este componente expondrá un API\_Rest con los métodos oportunos (GET, PUT, DELETE,...) que permitirá desde cualquier componente externo consultar el estado de los Taxis, Usuarios y el mapa en curso. Al igual que en la anterior versión de la práctica, el interfaz front que consuma este API deberá contener todos los elementos necesarios para visualizar claramente el estado de situación de todos los elementos del sistema (taxis, usuarios, mensajes de error, estado del CTC, etc).

### Registry (EC\_Registry):

EC\_Registry será el módulo mediante el cual los taxis pueden darse de alta o baja en el sistema. Solo dispondrá de las opciones de alta y baja de un taxi.



Los taxis, previo a su autenticación y posterior prestación de servicios, deberán realizar una petición de alta para registrar su ID en el sistema o darlo de baja. Sin este proceso los taxis no podrán autenticarse ni prestar servicios. Ante un intento de autenticación sin previo registro recibirán una denegación de esta.

El mecanismo de integración con los taxis se realizará mediante un API\_Rest que implementará los métodos oportunos (GET, PUT, DELETE, ...) para el registro de los Taxis.

La comunicación entre EC\_Registry y EC\_DE (Taxis) deberá realizarse mediante el establecimiento de un canal seguro (HTTPS, SSL, RSA, ...).

La identificación de los Taxis podrá realizarse en base a un certificado que contenga las credenciales de este.

### **Base de Datos:**

Podrá ser accesible tanto EC\_Central como EC\_Registry para compartir y actualizar la información de los taxis.

## **Digital Engine (EC\_DE)**

Adicionalmente a las funcionalidades ya disponibles, este módulo implementará una nueva opción (disponible en un menú) para poder conectarse al EC\_Registry y registrarse en el sistema. Este proceso se implementará mediante el consumo de un API.

La conexión y autenticación deberá realizarse de forma segura evitando la exposición en claro de los datos de identificación del taxi.

## **City Traffic Control (EC\_CTC)**

Como se ha indicado en el apartado de descripción funcional de este documento, se añadirá un nuevo módulo al sistema llamado EC\_CTC (City Traffic Control) el cual determinará la viabilidad de circulación en la ciudad donde se presta el servicio devolviendo un OK en caso que haya total normalidad o un KO en caso que la circulación no sea viable y los Taxis deban volver a la base.

La viabilidad de la circulación vendrá determinada por la temperatura de la ciudad.

Para ello, el módulo EC\_CTC dispondrá, a efectos de simulación, mediante una opción de menú, la posibilidad de indicar una ciudad donde se presta el servicio.

EC\_CTC se conectará al servidor de clima externo OPENWEATHER para preguntarle el tiempo que hace en la ciudad.

Si la temperatura de la ciudad está por debajo de los 0º C, EC\_CTC devolverá un mensaje de KO al módulo Central. En cualquier otro caso enviará un OK.

Nota: A efectos de la corrección, **la ciudad podrá ser cambiada a voluntad del profesor en cualquier momento** mediante la opción de menú mencionada, sin necesidad de reiniciar ninguna parte del sistema.

## Front

Este módulo consistirá en una simple página web que, haciendo peticiones al API\_Central muestre el mapa en curso y el estado de los Taxis y los Usuarios.

Como se ha comentado anteriormente, este interfaz front deberá contener todos los elementos necesarios para visualizar claramente el estado de situación de todos los elementos del sistema: taxis (incluyendo su estado de registro, activación y token), usuarios, mensajes de error de cualquier parte del sistema, mensajes de estado del CTC, alertas ante cualquier fallo de cualquier módulo del sistema, etc).

***Nota: Aunque al igual que en el resto de componentes el estudiante podrá elegir la tecnología de su preferencia se recomienda, por su sencillez, el uso de Node js con arreglo a las indicaciones que se han ofrecido en la práctica guiada entregada.***

## Sensores y Usuarios

No cambian su funcionalidad respecto de la entrega anterior.

## Servidor de clima (OPEN WEATHER)

Mediante la funcionalidad aportada por la plataforma OPEN WEATHER (<https://openweathermap.org/>) es posible obtener múltiples datos relacionados con el clima de cualquier parte del mundo.

A través de su API el EC\_Central podrá acceder a dichos datos.

Para ello bastará con que el estudiante se registre en la versión FREE, solicite el API key (Get API Key) y realizar una petición para cada ciudad que quiera consultar.

### Current weather and forecasts collection

Free	Startup 40 USD / month	Developer 180 USD / month	Professional 470 USD / month	Enterprise 2,000 USD / month
<a href="#">Get API key</a>	<a href="#">Subscribe</a>	<a href="#">Subscribe</a>	<a href="#">Subscribe</a>	<a href="#">Subscribe</a>
60 calls/minute 1,000,000 calls/month	600 calls/minute 10,000,000 calls/month	3,000 calls/minute 100,000,000 calls/month	30,000 calls/minute 1,000,000,000 calls/month	200,000 calls/minute 5,000,000,000 calls/month
Current Weather	Current Weather	Current Weather	Current Weather	Current Weather
Minute Forecast 1 hour*	Minute Forecast 1 hour**	Minute Forecast 1 hour	Minute Forecast 1 hour	Minute forecast 1 hour
Hourly Forecast 2 days*	Hourly Forecast 2 days**	Hourly Forecast 4 days	Hourly Forecast 4 days	Hourly Forecast 4 days
Daily Forecast 7 days*	Daily Forecast 16 days	Daily Forecast 16 days	Daily Forecast 16 days	Daily Forecast 16 days
National Weather Alerts*	National Weather Alerts**	National Weather Alerts	National Weather Alerts	National Weather Alerts
Historical weather 5 days*	Historical weather 5 days**	Historical weather 5 days	Historical weather 5 days	Historical weather 5 days
Climatic Forecast 30 days	Climatic Forecast 30 days	Climatic Forecast 30 days	Climatic Forecast 30 days	Climatic Forecast 30 days
Bulk Download	Bulk Download	Bulk Download	Bulk Download	Bulk Download
Basic weather maps	Basic weather maps	Advanced weather maps	Advanced weather maps	Advanced weather maps
Historical maps	Historical maps	Historical maps	Historical maps	Historical maps
Global Precipitation Map	Global Precipitation Map	Global Precipitation Map	Global Precipitation Map	Global Precipitation Map
Road Risk API	Road Risk API	Road Risk API	Road Risk API	Road Risk API
Air Pollution API	Air Pollution API	Air Pollution API	Air Pollution API	Air Pollution API
Geocoding API	Geocoding API	Geocoding API	Geocoding API	Geocoding API
Weather widgets	Weather widgets	Weather widgets	Weather widgets	Weather widgets
Uptime 95%	Uptime 95%	Uptime 99.5%	Uptime 99.5%	Uptime 99.9%

Modelos de suscripción de Open Weather

De todas las posibilidades que ofrece el API es suficiente con que se acceda a las funcionalidades que se encuentran en los métodos de “Current weather data” como se aprecia en la imagen siguiente. Toda la información al respecto se encuentra disponible en la página web del proveedor ( <https://openweathermap.org/current> ):

## Current weather data

---

Access current weather data for any location on Earth including over 200,000 cities! We collect and process weather data from different sources such as global and local weather models, satellites, radars and a vast network of weather stations. Data is available in JSON, XML, or HTML format.

## Call current weather data for one location

### By city name

You can call by city name or city name, state code and country code. Please note that searching by states available only for the USA locations.

#### API call

---

```
api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}
```



```
api.openweathermap.org/data/2.5/weather?q={city name},{state code}&appid={API key}
```



```
api.openweathermap.org/data/2.5/weather?q={city name},{state code},{country code}&appid={API key}
```



*API Current Weather Data*

El JSON de respuesta de las peticiones que se muestran en la anterior imagen es el siguiente del cual solo necesitaremos la temperatura.

## JSON

Example of API response

```
{
  "coord": {
    "lon": -122.08,
    "lat": 37.39
  },
  "weather": [
    {
      "id": 800,
      "main": "Clear",
      "description": "clear sky",
      "icon": "01d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 282.55,
    "feels_like": 281.86,
    "temp_min": 280.37,
    "temp_max": 284.26,
    "pressure": 1023,
    "humidity": 100
  },
  "visibility": 16093,
  "wind": {
    "speed": 1.5,
    "deg": 350
  },
  "clouds": {
    "all": 1
  },
  "dt": 1560350645,
  "sys": {
    "type": 1,
    "id": 5122,
    "message": 0.0139,
    "country": "US",
    "sunrise": 1560343627,
    "sunset": 1560396563
  },
  "timezone": -25200,
  "id": 420006353,
  "name": "Mountain View",
  "cod": 200
}
```

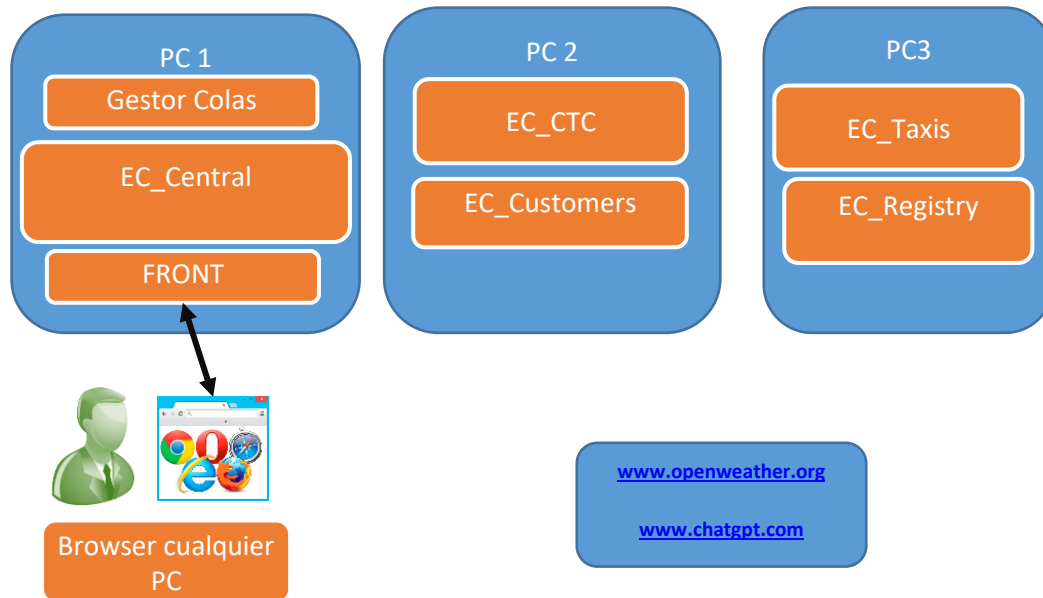
Temperatura en °K

Ciudad

JSON de respuesta del request al API Current Weather Data

## 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 que para su prueba es necesario al menos 3 PCs distintos en los que se desplegarán los componentes solicitados proporcionando el siguiente escenario:



*Escenario físico 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 con todos los módulos 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 16/12/2024.**