Sistemas Distribuidos 2023-2024

Práctica 3 - Art With Drones

Santiago Nogales Chiarenza 74011341D
Alejandro Molines Molina 48800938K

INTRODUCCIÓN

Esta práctica es una actualización de la práctica anterior, ampliando y fortaleciendo nuestro sistema distribuido llamado "Art With Drones" (AwD). En esta iteración, nos centraremos en la implementación de servicios basados en arquitectura SOA (Service Oriented Architecture), haciendo uso de tecnologías de comunicación RESTful.

El objetivo primordial de esta práctica es la introducción de conceptos avanzados de seguridad en nuestra arquitectura, abordando temas cruciales como el cifrado de canal, autenticación segura y auditoría de eventos. Además, se incorporarán cambios sustanciales en la comunicación entre componentes, como el consumo de API Rest externas y la exposición de un API Rest propio desde el back hacia el front.

A lo largo de esta práctica, los estudiantes serán desafiados a consumir API Rest de un proveedor externo para obtener datos climáticos, a crear un front-end accesible públicamente, y a implementar medidas de seguridad robustas en la comunicación entre los distintos módulos del sistema.

Continuaremos trabajando sobre la base de la primera parte de la práctica, mejorando y extendiendo nuestra arquitectura. Se espera que los estudiantes desarrollen un conocimiento más profundo de la interconexión de sistemas distribuidos, así como una comprensión sólida de los principios de seguridad en el contexto de las API Rest.

TECNOLOGÍAS UTILIZADAS

Exploramos una variedad de tecnologías que nos permitieron crear un sistema distribuido robusto y escalable. A continuación, se detallan las tecnologías clave que implementamos y utilizamos para llevar a cabo este proyecto:

- Flask: Empleamos Flask para la creación de un API Rest seguro desde el componente Engine. Esta tecnología nos proporcionó un marco de desarrollo web ligero y flexible, permitiendo exponer funcionalidades de manera eficiente.

- SSL (Secure Socket Layer): Implementamos SSL para cifrar el canal de comunicación entre los componentes distribuidos. Esta capa de seguridad asegura la confidencialidad y protección de los datos durante su transferencia.

- JavaScript y HTML: Desarrollamos el front-end mediante una página web simple utilizando JavaScript y HTML. Esta interfaz proporciona una visualización intuitiva en tiempo real del mapa y el estado de los drones, mejorando la interacción con el sistema.

COMPONENTES

ADDrone

Se incorporó una nueva opción, "5. Take Over Drone", que permite al usuario seleccionar un dron existente para controlarlo. Se añadió el método take_over_drone correspondiente.

Este método permite al usuario seleccionar un dron existente de una lista y asumir su control.

Se agregó un nuevo método para listar los drones disponibles en la base de datos.

En lugar de iniciar el menú (self.show_menu()) en el constructor, ahora se inicia el hilo (self.start()). Además, se agregó el atributo self.in_show_mode con un valor inicial de False.

En lugar de iniciar el menú, se llama al método start_consuming_messages para iniciar el consumo de mensajes del dron.

Se añadió un mensaje de impresión indicando que el dron está esperando mensajes de Kafka.

Se agregaron configuraciones SSL para las conexiones Kafka y el socket SSL en el método join_show.

Se implementaron bloques try-except en varios lugares para manejar excepciones y proporcionar mensajes de error informativos.

Se introdujo la generación de claves RSA para la encriptación de los datos, lo que mejora considerablemente la seguridad en la comunicación y se implementaron métodos para cifrar y descifrar mensajes, utilizando estas claves para asegurar la transmisión de datos entre el dron y otros componentes del sistema.

Se mejoró la forma en que se consumen y procesan los mensajes de Kafka, incluyendo el manejo de mensajes cifrados y se añadió la lógica para decodificar mensajes en Base64 y descifrarlos antes de su procesamiento.

Se agregó funcionalidad para almacenar las claves generadas y los mensajes de Kafka en MongoDB, lo que facilita la gestión y el seguimiento de estas informaciones.

Se introdujeron cambios en los métodos de registro del dron, incluyendo opciones para registrar a través de una API o directamente, además se añadió la capacidad de generar y almacenar claves criptográficas en MongoDB durante el registro y se implementó un sistema para obtener tokens JWT (JSON Web Tokens) que probablemente se utilizan para la autenticación.

Se agregó la lógica para manejar el estado de conexión del dron y verificar si está "en show" y se introdujo un mecanismo de heartbeat para mantener la conexión activa y detectar desconexiones.

Se mejoraron las conexiones SSL para comunicarse con seguridad con el ADEngine y se implementaron cambios en la forma en que se envían y reciben mensajes del ADEngine, utilizando cifrado para mejorar la seguridad.

Se implementaron funciones para registrar usuarios y obtener tokens JWT, lo que sugiere una capa adicional de autenticación y gestión de usuarios.

ADEngine

En el código modificado, se utiliza SSL/TLS para asegurar la conexión del servidor. Se crea un contexto SSL y se envuelve la conexión del cliente con este contexto en el método accept_connections. Esto proporciona una capa de seguridad adicional para la comunicación entre el servidor y los drones.

En el código modificado, el KafkaProducer se inicializa con configuraciones adicionales para admitir SSL/TLS. Se proporcionan los archivos de certificado CA, certificado del servidor y clave privada del servidor

.

En el código original, hay una importación del módulo Flask (from flask import Flask). Sin embargo, no parece haber ninguna instancia de la aplicación Flask ni se usa en ninguna parte del código proporcionado. Puede ser que esta importación sea un remanente de código no utilizado o se haya omitido parte del código donde se utiliza Flask.

En el código modificado, se manejan las excepciones ssl.SSLError cuando se intenta envolver la conexión del cliente con el contexto SSL en el método accept_connections.

El método check_all_drones_in_position se ha modificado para proporcionar un registro (logging.info) cuando todos los drones han alcanzado sus posiciones finales.

Se ha agregado un nuevo método llamado update_drone_position. Este método verifica si un dron ha llegado a la posición final y, en caso afirmativo, envía una instrucción de finalización al dron.

Se ha agregado un mensaje de impresión al método check_all_drones_connected para indicar que todos los drones están conectados para la figura actual.

En algunos lugares del código modificado, se ha utilizado el módulo logging en lugar de print para mostrar mensajes. Esto puede ser beneficioso para una mejor gestión de registros.

Se agregó la generación de claves RSA para la encriptación de datos, lo que mejora la seguridad en la transmisión de datos entre los drones y el servidor y se implementaron métodos para cargar y utilizar las claves privadas de los drones para descifrar datos recibidos, y para cifrar mensajes antes de enviarlos.

Se añadió funcionalidad para almacenar mensajes cifrados y descifrados en MongoDB, lo que permite un mejor seguimiento de la comunicación y aumenta la seguridad.

Se modificó la forma de manejar las conexiones y los mensajes de los drones, incluyendo el procesamiento de datos cifrados y la respuesta a diferentes tipos de acciones y se introdujo un conjunto para manejar los drones involucrados en el espectáculo (drones_involucrados), lo que puede ayudar en la organización y control de los drones.

Se mejoró la manera de enviar mensajes a Kafka, incluyendo la encriptación de mensajes y se añadió la deserialización y descifrado de mensajes Kafka en el consumidor, mejorando la seguridad y la integridad de los datos.

Se implementó un hilo para enviar mensajes de 'heartbeat', lo que puede ayudar en la monitorización del estado y la disponibilidad del sistema y se añadió un método para gestionar el estado de los drones y verificar si todos han alcanzado sus posiciones.

Se agregaron configuraciones SSL para las conexiones Kafka y el manejo de excepciones en varios puntos del código para una mejor robustez y seguridad.

Se modificó el flujo de inicialización y control, moviendo la lógica de inicio del menú a la inicialización de un hilo y añadiendo el modo de espectáculo (self.in_show_mode).

ADRegistry

Se agregó una nueva función register_drone_via_api que permite registrar un dron a través de una API REST en lugar de usar directamente MongoDB. Esto proporciona una interfaz más flexible para el registro y permite cambiar fácilmente el mecanismo de almacenamiento sin modificar el código principal.

Se agregó la generación del access_token en la función register_drone, para añadir seguridad temporal a la aplicación

La respuesta al cliente ahora solo incluye un mensaje de éxito o error, eliminando la necesidad de enviar el access_token en la respuesta.

Se ha simplificado la función handle_client para que se centre en la lógica del registro del dron, delegando el envío de respuestas al cliente y otros detalles a funciones específicas.

La función consume_drone_registered_messages ahora está más aislada y se encarga específicamente del consumo de mensajes de Kafka relacionados con el registro de drones.

Se ha mejorado la organización y la legibilidad del código mediante el uso de funciones más pequeñas y descriptivas.

En la nueva versión, se ha agregado la generación de claves RSA para encriptar y desencriptar datos. Esto mejora la seguridad en la comunicación, especialmente en lo que respecta al manejo de información sensible de los drones y se implementaron métodos para cargar claves privadas de drones específicos desde MongoDB y para descifrar datos recibidos, lo cual es crucial para asegurar la integridad y confidencialidad de la información.

La nueva versión introduce el cifrado de mensajes antes de enviarlos a Kafka y el almacenamiento de mensajes cifrados en MongoDB. Esta práctica es fundamental para proteger la información en tránsito y en reposo y se ha modificado la forma de registrar los drones, incorporando métodos para registrar drones tanto directamente como a través de una API, utilizando datos cifrados para mayor seguridad.

Se han realizado mejoras en la forma de manejar las conexiones de socket y en la gestión de errores, lo que indica una mayor robustez y confiabilidad del sistema.

La nueva versión incluye la funcionalidad para almacenar y manejar claves criptográficas en MongoDB, lo que representa una mejora significativa en el manejo seguro de claves.

Ambas versiones utilizan argparse para la configuración, lo que facilita la personalización y configuración del servicio.

La nueva versión parece más general y adaptable, lo que sugiere una mejor capacidad para integrarse en diversos entornos y configuraciones.

API_Rest

La API REST proporciona una interfaz para la gestión y control de drones, utilizando Flask como marco de desarrollo. A continuación, se detallan las principales características y funcionalidades del código:

Configuración e Inicialización:

La aplicación Flask se inicializa en el objeto app. Se utiliza el módulo Flask-CORS para habilitar el manejo de solicitudes de origen cruzado (CORS). La función create_app se encarga de configurar la conexión a MongoDB y Kafka.

Conexiones a Bases de Datos y Kafka:

Se establece una conexión a MongoDB utilizando la dirección proporcionada. Se configura un consumidor de Kafka para el tema "drone_position_updates" con la dirección de Kafka proporcionada. Se crea un diccionario drone_positions para almacenar las posiciones de los drones.

Gestión del Clima:

Se define la función get_weather que utiliza la API de OpenWeatherMap para obtener la temperatura de una ciudad.

Rutas y Controladores:

La ruta principal ("/") devuelve la plantilla HTML "index.html". La ruta "/weather/<city>" proporciona la temperatura de una ciudad específica.

Autenticación y Seguridad:

Se configura Flask-JWT-Extended para manejar la autenticación mediante tokens JWT. Se definen rutas para el registro de usuarios, inicio de sesión y acceso a rutas protegidas mediante JWT.

Kafka Listener y Exposición de Datos:

La función kafka_listener escucha los mensajes del tema "drone_position_updates" de Kafka y actualiza las posiciones de los drones en drone_positions. La ruta "/get_drone_positions" devuelve las posiciones actuales de los drones.

Operaciones CRUD en MongoDB:

Se implementan rutas para registrar, listar, modificar y eliminar drones en la base de datos MongoDB.

Manejo de Errores y Auditoría:

Se define una función error_response para manejar respuestas de error de manera uniforme. Se utiliza un archivo de registro ("registro_auditoria.log") para registrar eventos y errores.

Inicio de la Aplicación:

Se inicia la aplicación Flask en el puerto 5000. Se utiliza la biblioteca argparse para permitir la configuración de las direcciones de MongoDB y Kafka mediante argumentos de línea de comandos.

Hilos y Daemonización:

La función kafka_listener se ejecuta en un hilo separado y se marca como un hilo demonio para que termine cuando la aplicación principal finalice.

SSL (Comentado):

Existe un bloque de código comentado para habilitar la comunicación segura a través de SSL.

Interfaz Web:

Se define una ruta ("/index") que devuelve la plantilla "index.html", la cual probablemente contenga un formulario para interactuar con la aplicación.

Manejo de Excepciones:

Se define una ruta de manejo de errores (@app.errorhandler(Exception)) que registra errores en el archivo de registro y devuelve respuestas de error.

Inicio de la Aplicación con Argumentos:

En la parte final, se verifica si el script se ejecuta directamente (name == 'main') y se inicia la aplicación Flask con los argumentos proporcionados.

La nueva versión introduce el uso de criptografía para proteger los datos transmitidos. Esto incluye la generación de claves RSA y el uso de encriptación y desencriptación para manejar la comunicación segura, especialmente en lo que respecta a la información sensible de los drones.

Ambas versiones intentan conectar a Kafka y MongoDB y manejar errores en estas conexiones. La nueva versión mantiene esta funcionalidad pero con un enfoque más robusto y seguro.

La nueva versión incluye mejoras en el manejo de mensajes Kafka para posiciones de drones. Se introduce un mejor manejo de los datos recibidos, incluyendo la decodificación y verificación del formato JSON.

Ambas versiones implementan un registro de auditoría y manejo de errores, pero la nueva versión mejora en la forma en que se manejan y registran los errores, lo que implica una mayor robustez del sistema.

La nueva versión utiliza una clave secreta JWT obtenida de una variable de entorno, lo que mejora la seguridad en comparación con una clave secreta estática. Este cambio es crucial para la seguridad de la autenticación y la autorización.

Se mantienen las rutas de la API para diversas funcionalidades como el registro de usuarios, login, protección de rutas, entre otras. La nueva versión parece tener una mejor organización y manejo de estas rutas.

En la nueva versión, se configura el contexto SSL para la ejecución de la aplicación Flask, lo que indica una preocupación por la seguridad en la transmisión de datos.

La nueva versión muestra un manejo más cuidadoso de los datos de entrada y una mejor validación, especialmente en las rutas de la API que interactúan con la base de datos MongoDB.

Index

El HTML define la estructura básica de una página web con metadatos, enlaces a recursos y el cuerpo principal.

La barra superior muestra el título "ART WITH DRONES". La barra de navegación tiene un botón desplegable para acceder a secciones como mapa, registro y errores.

Existe un contenedor para mostrar el mapa de drones (aunque no se proporcionan detalles sobre su implementación específica).

Hay un área para ingresar el nombre de una ciudad y un botón para obtener información climática asociada.

Sección que presenta una lista de drones. Los botones "Añadir" y "Borrar" permiten interactuar con la lista de drones.

Se enlaza un archivo de scripts externo que probablemente define funciones para manipular la interfaz de usuario y realizar acciones interactivas. Se utilizan funciones como getWeather, addDrone, y delDrone que están vinculadas a acciones específicas.

Los botones en la barra de navegación permiten navegar entre secciones del sitio. La entrada de clima proporciona una forma de obtener información meteorológica. La lista de drones es manipulable mediante los botones "Añadir" y "Borrar".

Se aplica un estilo a través de un archivo CSS externo (style.css) para mejorar la presentación y experiencia del usuario.

Stylesheet

Estilo General:

Se establece un estilo general para el cuerpo (body) de la página, incluyendo la fuente, margen, relleno y color de fondo.

Barra Superior (top-bar) y Navegación (nav):

Estilos para la barra superior y la navegación, definiendo colores, posición, y formato de texto.Se crea un botón desplegable (dropdown-btn) con estilos específicos y un ícono de hamburguesa (burger-icon).

Estilos para Mapa de Drones (#drone-map):

Se establece un diseño de cuadrícula para el mapa de drones con celdas, márgenes y dimensiones específicas. Las clases .drone y .drone-cell definen el estilo de los drones y las celdas en el mapa.

Estilos para la Lista de Drones (#drone-list-section):

Posicionamiento y estilo de la sección que muestra la lista de drones, con colores, sombras y bordes. Se define el estilo de los elementos de la lista (#drone-list li), así como los botones para añadir y borrar drones.

Estilos Generales (black-container, orange-background, etc.):

Estilos reutilizables para contenedores, fondos, colores de texto, botones y sus interacciones.

Estilos para Entrada de Clima (#weather-input):

Se define el estilo del contenedor de entrada de clima, con colores, márgenes y disposición de elementos.

Estilos de Placeholder y Botones (#city-name::placeholder, .orange-button):

Estilos específicos para el placeholder de la entrada de ciudad y los botones de color naranja.

Estilos Hover:

Se establecen estilos específicos cuando se pasa el mouse sobre algunos elementos, como los botones naranjas.

Scripts

Interfaz de Usuario:

- Manejo de menús desplegables.
- Creación dinámica de un mapa de drones con celdas.
- Actualización de la lista de drones y posiciones en el mapa.

Obtención de Datos Externos:

- Solicita al servidor la temperatura de una ciudad y muestra la información al usuario.
- Solicita y gestiona la obtención de un token JWT para la autenticación.

Registro e Interacción con Drones:

- Permite al usuario registrar nuevos drones, solicitando ID y alias.
- Elimina drones existentes, solicitando el ID al usuario.

Actualización Periódica:

- Actualiza automáticamente las posiciones de los drones en el mapa cada 2 segundos.
- Resalta las posiciones finales de algunos elementos en el mapa.

Gestión de Errores y Alertas:

- Maneja errores durante las interacciones, mostrando mensajes de alerta al usuario.

Integración con el Servidor:

- Realiza solicitudes al servidor para operaciones como registro de usuarios, drones y obtención de datos.

Carpeta ssl

certificado_CA.crt:

Este archivo contiene el certificado público de la Autoridad de Certificación (CA). Se utiliza para verificar la autenticidad de otros certificados firmados por la CA.

certificado_registry.crt:

Contiene el certificado público específico para la entidad "registry". Es probablemente el certificado utilizado para habilitar conexiones seguras para el servicio o entidad denominada "registry".

clave_privada_CA.pem:

Es la clave privada asociada a la Autoridad de Certificación (CA). Esta clave se utiliza para firmar certificados de otras entidades, estableciendo la confianza en la cadena de certificados.

clave_privada_registry.pem:

Corresponde a la clave privada asociada al certificado de la entidad "registry". Esta clave se utiliza para descifrar la información cifrada con el certificado y garantizar la autenticidad de la entidad.

openssl-sans.cnf:

Este archivo de configuración se utiliza con OpenSSL y probablemente contiene configuraciones específicas para la generación de certificados con Subject Alternative Names (SAN), que permiten especificar múltiples nombres de dominio.

peticion_csr_registry.csr:

Representa la solicitud de firma de certificado (CSR) para la entidad "registry". Este archivo contiene la información necesaria para que la CA genere el certificado correspondiente.

GUÍA DE DESPLIEGUE

Paso 1: Preparación del entorno

Iniciamos docker-compose con sudo docker-compose up

Paso 2: Iniciar AD_Registry

Desde la terminal en la máquina destinada a AD_Registry, navega al directorio donde se encuentra el archivo AD_Registry.py y ejecuta el siguiente comando, proporcionando la información necesaria:
python AD_Registry

Paso 3: Iniciar AD_Engine

Desde la terminal en la máquina destinada a AD_Engine, navega al directorio donde se encuentra el archivo AD_Engine.py y ejecuta el siguiente comando, proporcionando la información necesaria:
python AD_Engine

Paso 4: Iniciar AD_Drone

Desde la terminal en la máquina destinada a AD_Drone, navega al directorio donde se encuentra el archivo AD_Drone.py y ejecuta el siguiente comando, proporcionando la información necesaria:
python AD_Drone

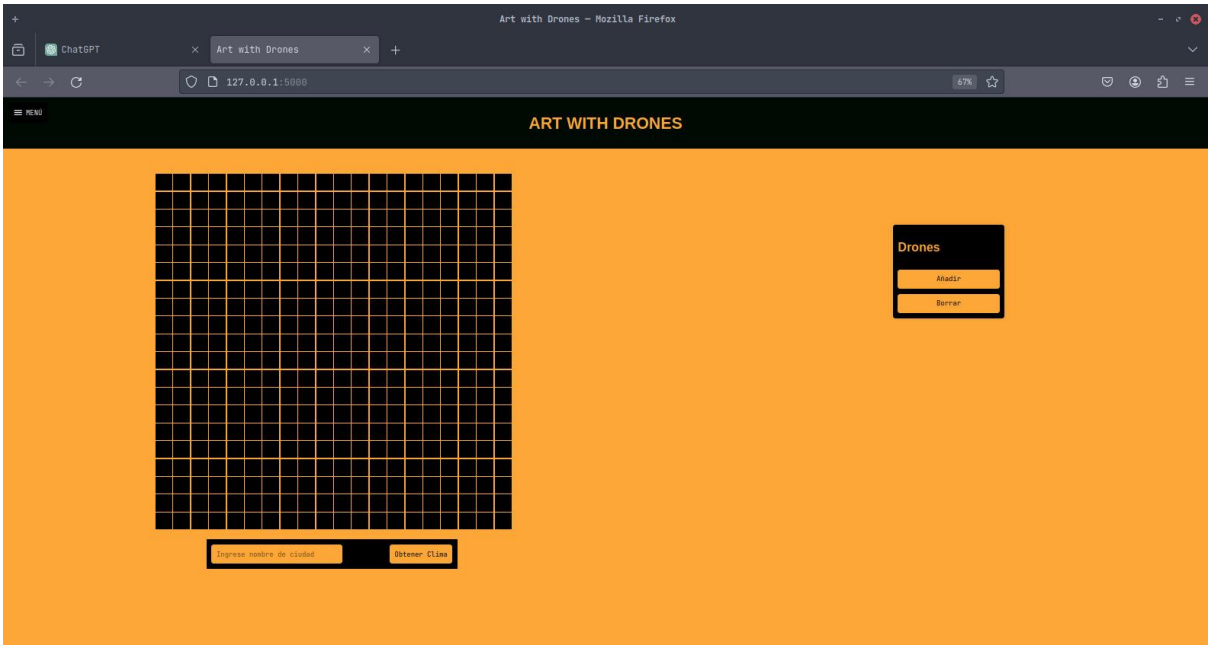
Paso 5: Iniciar API_Rest

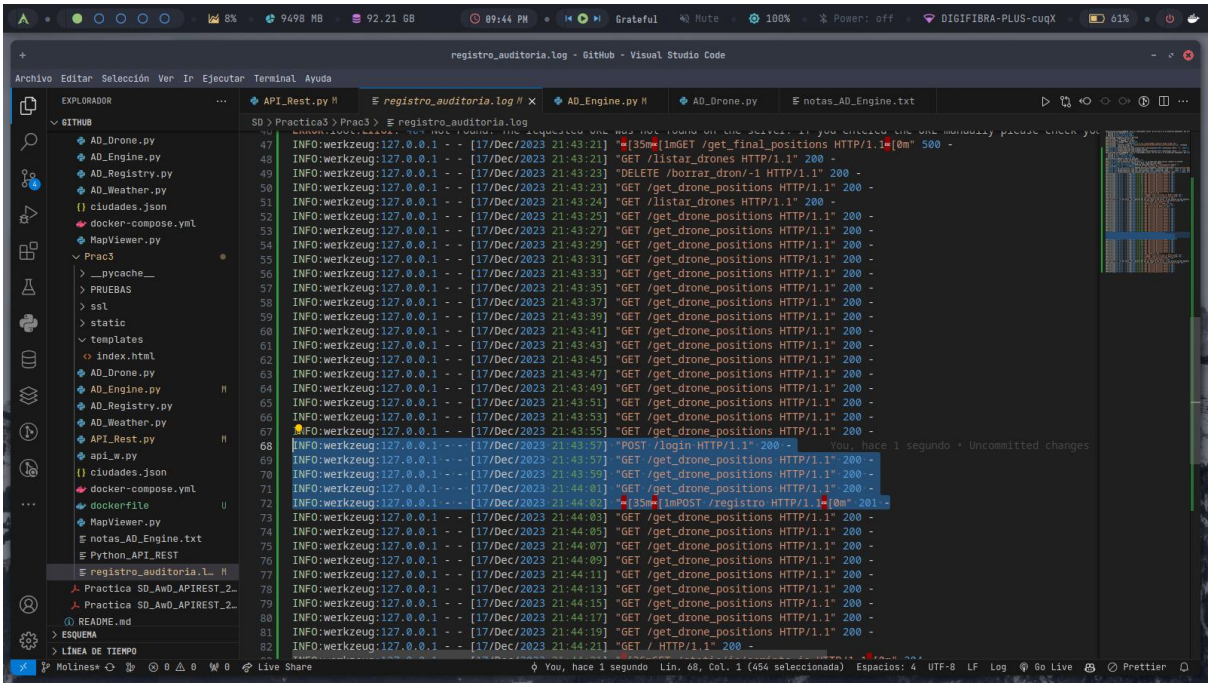
Desde la terminal en la máquina destinada a API_Rest, navega al directorio donde se encuentra el archivo API_Rest.py y ejecuta el siguiente comando, proporcionando la información necesaria:
python API_Rest

Paso 6: Ir al front:

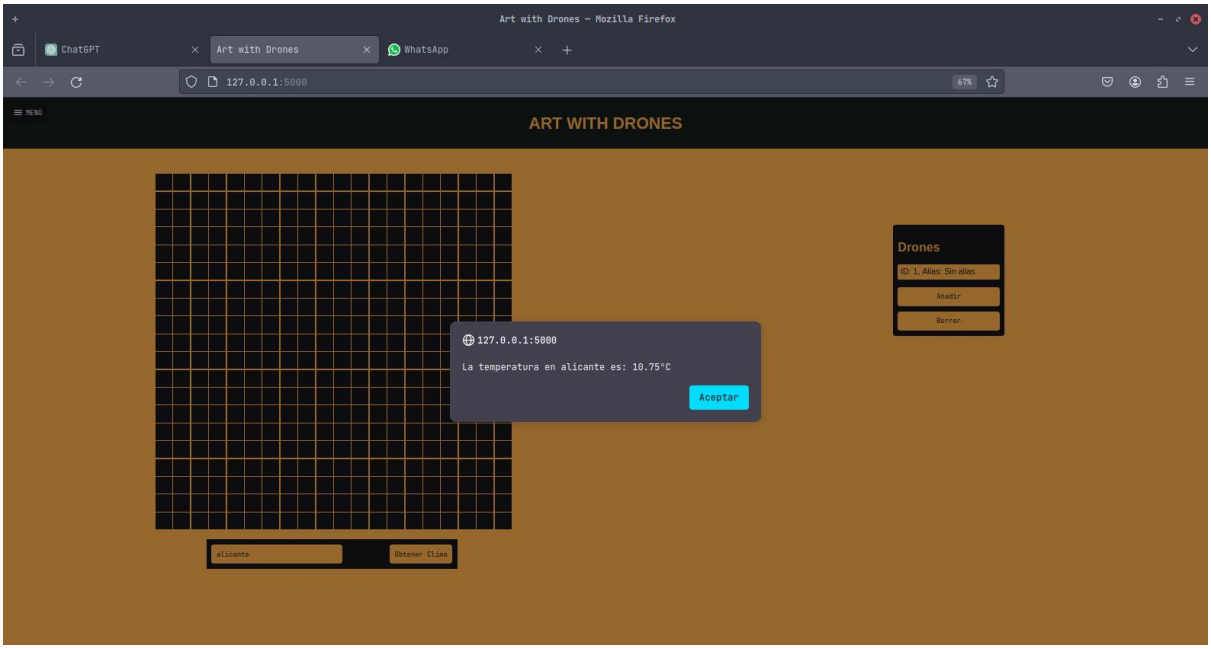
Habrá que abrir el navegador de preferencia (preferentemente google chrome) y navegar hacia la IP que te proporciona API_Rest. Una vez hecho esto podremos tener acceso total a los servicios del programa

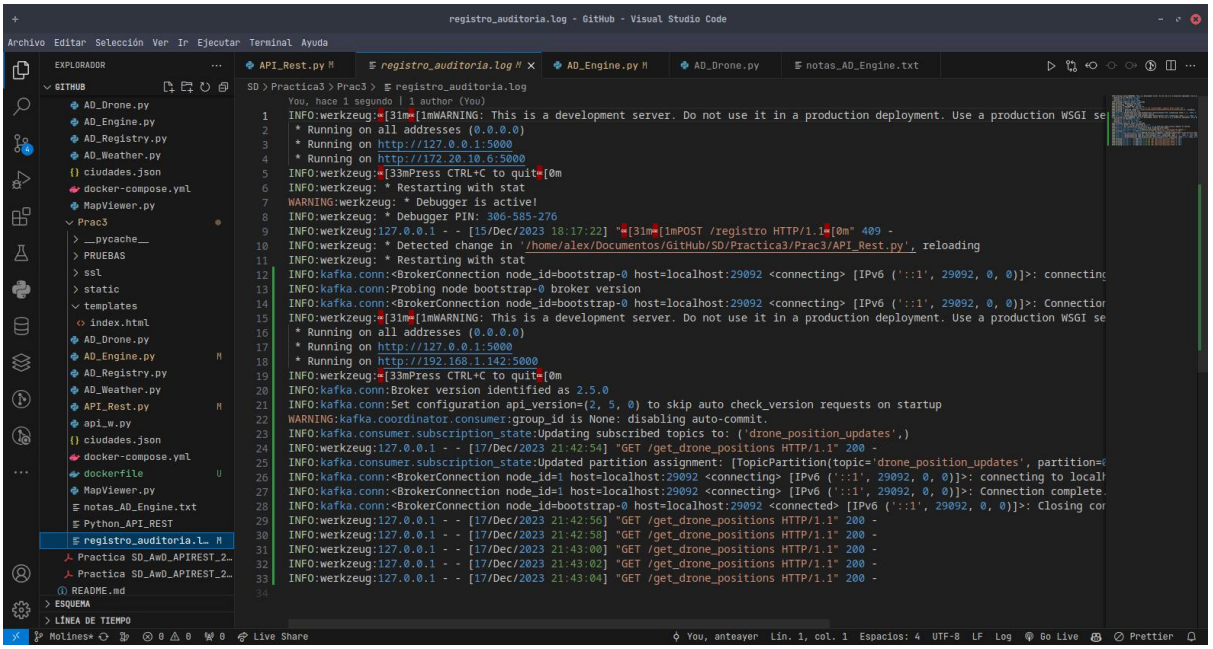
CAPTURAS

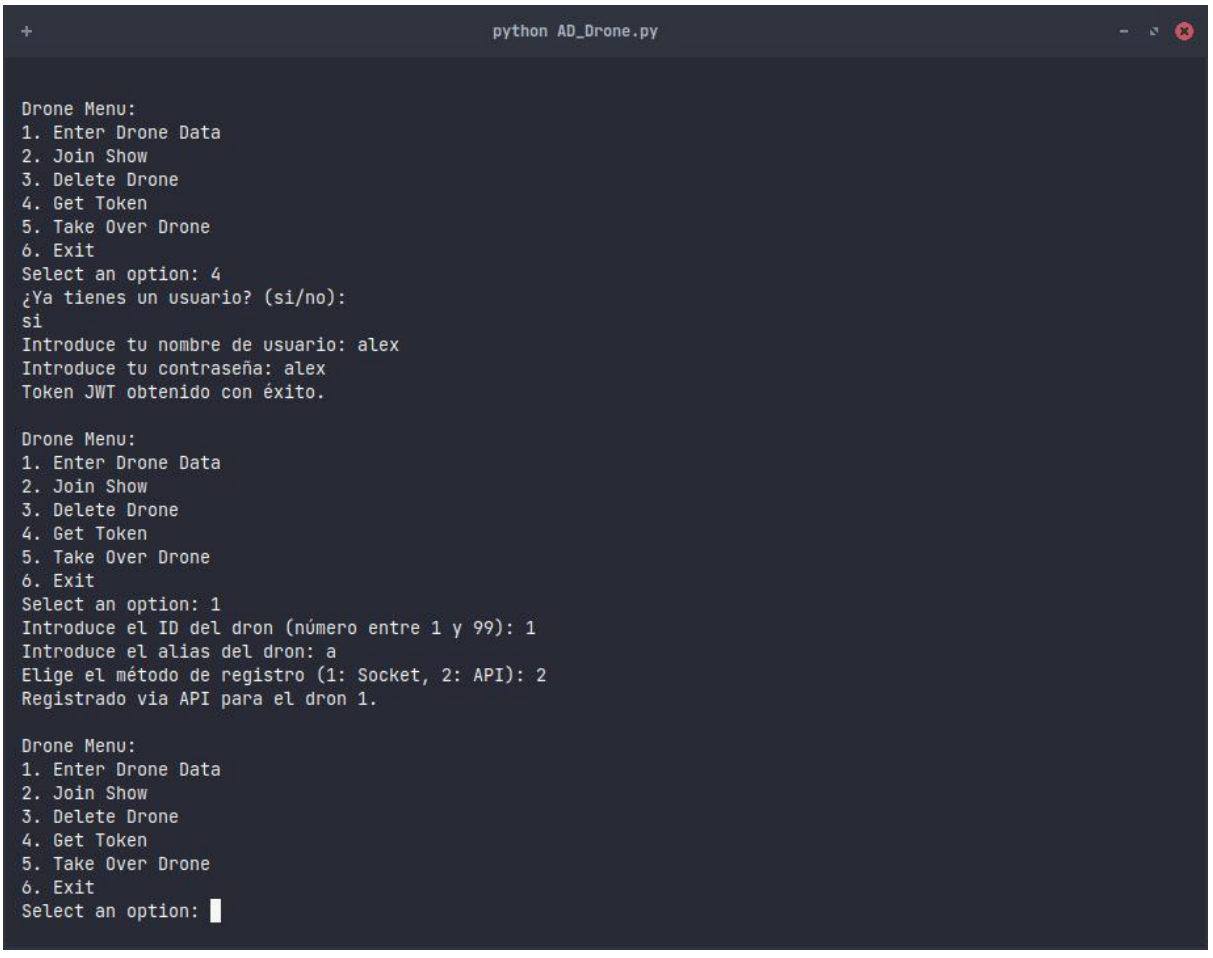

Art with Drones – Mozilla Firefox
ChatGPT
Art with Drones
127.0.0.1:5000
MENU
ART WITH DRONES
Drones
Añadir
Borrar
Ingrese nombre de ciudad
Obtener Clima


registro_auditoria.log - GitHub - Visual Studio Code
Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda
EXPLORADOR
GITHUB
API_Rest.py
registro_auditoria.log
AD_Engine.py
AD_Drone.py
notas_AD_Engine.txt
SD > Practica3 > Prac3 > registro_auditoria.log
AD_Drone.py
AD_Engine.py
AD_Registry.py
AD_Weather.py
ciudades.json
docker-compose.yml
MapViewer.py
Prac3
__pycache__
PRUEBAS
ssl
static
templates
index.html
dockerfile
api_w.py
notas_AD_Engine.txt
Python_API_REST
README.md
ESQUEMA
LÍNEA DE TIEMPO
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:21] "GET /get_final_positions HTTP/1.1" 500 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:21] "GET /listar_drones HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:23] "DELETE /borrar_dron/-1 HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:23] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:24] "GET /listar_drones HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:25] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:27] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:29] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:31] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:33] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:35] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:37] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:39] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:41] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:43] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:45] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:47] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:49] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:51] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:53] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:55] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:57] "POST /login HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:57] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:59] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:44:01] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:44:02] "POST /registro HTTP/1.1" 201 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:44:03] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:44:05] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:44:07] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:44:09] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:44:11] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:44:13] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:44:15] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:44:17] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:44:19] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:44:21] "GET / HTTP/1.1" 200 -
You, hace 1 segundo • Uncommitted changes
Live Share
You, hace 1 segundo
Lín. 68, Col. 1 (454 seleccionada)
Espacios: 4
UTF-8
LF
Log
Go Live
Prettier

```
python AD_Drone.py

2. Join Show
3. Delete Drone
4. Get Token
5. Take Over Drone
6. Exit
Select an option: 1
Introduce el ID del dron (número entre 1 y 99): 1
Introduce el alias del dron: a
Elige el método de registro (1: Socket, 2: API): 2
Registrado via API para el dron 1.

Drone Menu:
1. Enter Drone Data
2. Join Show
3. Delete Drone
4. Get Token
5. Take Over Drone
6. Exit
Select an option: 4
¿Ya tienes un usuario? (si/no):
si
Introduce tu nombre de usuario: alex
Introduce tu contraseña: alex
Token JWT obtenido con éxito.

Drone Menu:
1. Enter Drone Data
2. Join Show
3. Delete Drone
4. Get Token
5. Take Over Drone
6. Exit
Select an option: 2
Drone 1 has received final position: (3, 4)
ADDrone: Esperando mensajes de Kafka...
```




registro_auditoria.log - GitHub - Visual Studio Code
Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda
EXPLORADOR
GITHUB
AD_Drone.py
AD_Engine.py
AD_Registry.py
AD_Weather.py
ciudades.json
docker-compose.yml
MapViewer.py
Prac3
__pycache__
PRUEBAS
ssl
static
templates
index.html
AD_Drone.py
AD_Engine.py
AD_Registry.py
AD_Weather.py
API_Rest.py
api_w.py
ciudades.json
docker-compose.yml
dockerfile
MapViewer.py
notas_AD_Engine.txt
Python_API_REST
registro_auditoria.l...
Practica SD_AwD_APIREST_2...
Practica SD_AwD_APIREST_2...
README.md
ESQUEMA
LÍNEA DE TIEMPO
API_Rest.py M
registro_auditoria.log M
AD_Engine.py M
AD_Drone.py
notas_AD_Engine.txt
SD > Practica3 > Prac3 > registro_auditoria.log
You, hace 1 segundo | 1 author (You)
INFO:werkzeug:[31m[1mWARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI se
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.20.10.6:5000
INFO:werkzeug:[33mPress CTRL+C to quit[0m
INFO:werkzeug: * Restarting with stat
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 306-585-276
INFO:werkzeug:127.0.0.1 - - [15/Dec/2023 18:17:22] "[31m[1mPOST /registro HTTP/1.1[0m" 409 -
INFO:werkzeug: * Detected change in '/home/alex/Documentos/GitHub/SD/Practica3/Prac3/API_Rest.py', reloading
INFO:werkzeug: * Restarting with stat
INFO:kafka.conn:<BrokerConnection node_id=bootstrap-0 host=localhost:29092 <connecting> [IPv6 ('::1', 29092, 0, 0)]>: connectin
INFO:kafka.conn:Probing node bootstrap-0 broker version
INFO:kafka.conn:<BrokerConnection node_id=bootstrap-0 host=localhost:29092 <connecting> [IPv6 ('::1', 29092, 0, 0)]>: Connectior
INFO:werkzeug:[31m[1mWARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI se
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://192.168.1.142:5000
INFO:werkzeug:[33mPress CTRL+C to quit[0m
INFO:kafka.conn:Broker version identified as 2.5.0
INFO:kafka.conn:Set configuration api_version=(2, 5, 0) to skip auto check_version requests on startup
WARNING:kafka.coordinator.consumer:group_id is None: disabling auto-commit.
INFO:kafka.consumer.subscription_state:Updating subscribed topics to: ('drone_position_updates',)
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:42:54] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:kafka.consumer.subscription_state:Updated partition assignment: [TopicPartition(topic='drone_position_updates', partition=
INFO:kafka.conn:<BrokerConnection node_id=1 host=localhost:29092 <connecting> [IPv6 ('::1', 29092, 0, 0)]>: connecting to local
INFO:kafka.conn:<BrokerConnection node_id=1 host=localhost:29092 <connecting> [IPv6 ('::1', 29092, 0, 0)]>: Connection complete.
INFO:kafka.conn:<BrokerConnection node_id=bootstrap-0 host=localhost:29092 <connected> [IPv6 ('::1', 29092, 0, 0)]>: Closing cor
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:42:56] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:42:58] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:00] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:02] "GET /get_drone_positions HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Dec/2023 21:43:04] "GET /get_drone_positions HTTP/1.1" 200 -
Molines* 0 0 0 Live Share
You, anteayer Lín. 1, col. 1 Espacios: 4 UTF-8 LF Log Go Live Prettier


python AD_Drone.py
Drone Menu:
1. Enter Drone Data
2. Join Show
3. Delete Drone
4. Get Token
5. Take Over Drone
6. Exit
Select an option: 4
¿Ya tienes un usuario? (si/no):
si
Introduce tu nombre de usuario: alex
Introduce tu contraseña: alex
Token JWT obtenido con éxito.
Drone Menu:
1. Enter Drone Data
2. Join Show
3. Delete Drone
4. Get Token
5. Take Over Drone
6. Exit
Select an option: 1
Introduce el ID del dron (número entre 1 y 99): 1
Introduce el alias del dron: a
Elige el método de registro (1: Socket, 2: API): 2
Registrado via API para el dron 1.
Drone Menu:
1. Enter Drone Data
2. Join Show
3. Delete Drone
4. Get Token
5. Take Over Drone
6. Exit
Select an option: