Práctica 3: API REST con Bottle

Sistemas Distribuidos

Ejercicios

1. Probar los ejemplos de la carpeta Examples disponible en el Campus Virtual.

    - Ejemplos de peticiones `get`: openWeatherMap-example.py y cars-example.py
    - Ejemplo de petición `post`: pastebin-example.py
    - Ejemplo de petición `put`: renameNames-example.py
    - Ejemplo con varios `route` (endpoints): twoRoutes-example.py
    - Ejemplo de comprobación de tipo de parámetros: checkParameters-example.py
    - Ejemplo de servidor y cliente con clases y varios métodos: subjectsServer-example.py y subjectsClient-example.py

    Nota: Usar el comando `curl` es el comando para transferir datos hacia o desde un servidor, utilizando cualquiera de los protocolos soportados. Es la abreviatura de «Client URL».

    Sintaxis de uso:

    ```
    curl [options] [URL...]
    ```

    Ejemplos de uso

    - Método `GET`:

      ```
      curl localhost:8086/listCourses

      curl -X GET localhost:8086/listCourses
      ```

    - Método `POST`:

      ```
      curl -X POST -H "Content-Type: application/json" -d '{"code": "1", "name": "bd"}' localhost:8086/addCourse
      ```

- Método PUT:

```
curl -X PUT -H "Content-Type: application/json" -d
'{"code": "1", "name": "sd"}' localhost:8086/updateCourse/1
```

2. Escriba el código correspondiente a un endpoint de tipo POST con el framework Bottle en Python.

El nombre del endpoint será /inserta. Como entrada tomará un JSON, en el que se encontrará un número entero cuya clave será elemento. El servidor buscará el número indicado en el JSON de entrada en un array denominado mis_elementos, si el elemento no existe en el array lo insertará en la última posición, en caso contrario, no modificará el array.

Por último, devolverá al cliente, también en formato JSON, el array resultante con la clave mis_elementos. Si la clave elemento no se encuentra en el JSON de entrada, el servicio deberá devolver un error al cliente.

3. Desarrolle un servicio web de gestión de habitaciones de un hotel. El servicio debe poseer una lista de habitaciones y cada habitación poseerá, mínimo, los siguientes atributos:

- Identificador.
- Número de plazas.
- Lista de equipamiento (por ejemplo: armario, aire acondicionado, caja fuerte, escritorio, etc.).
- Ocupada (sí/no).

Se deben implementar los siguientes endpoints:

a) Dar de alta una nueva habitación.

b) Modificar los datos de una habitación.

c) Consultar la lista completa de habitaciones.

d) Consultar una habitación mediante identificador.

e) Consultar la lista de habitaciones ocupadas o desocupadas.

Queda a juicio del estudiante identificar qué tipo de operaciones HTTP debe implementar cada endpoint (PUT, POST o GET).

Para los endpoints b), d) y e), es obligatorio el uso de parámetros en el path de la URL.

Aunque hemos visto que REST permite transmitir los datos en cualquier formato, se recomienda utilizar JSON, tal y como se ha visto en el seminario.

Se valorará positivamente el manejo adecuado de los errores (parámetros incorrectos, `endpoint` no encontrado, operaciones no permitidas) en el servicio y su notificación al cliente.

Como mejoras puede practicar lo siguiente:

- Crear un `endpoint` para eliminar una habitación mediante una operación `HTTP DELETE`.
- Hacer que la información que maneja el servicio sea persistente, almacenando la información de las habitaciones en un fichero.
- Crear una clase "`Room`" para facilitar el manejo de la información por parte del servicio.
- Crear los `endpoints` adicionales que el estudiante considere oportuno.

Por último, se debe crear un cliente que permita a un usuario interactuar con la API desarrollada.

```
Elige que opción deseas realizar:
    1. Dar de alta una nueva habitación
    2. Modificar los datos de una habitación
    3. Consultar la lista completa de habitaciones
    4. Consultar una habitación mediante identificador
    5. Consultar la lista de habitaciones ocupadas o desocupadas
    6. Salir

> 4

Introduce el identificador de la habitación:

> 80

Habitación 80:
    - 3 plazas.
    - Equipamiento: TV, Wifi y escritorio.
    - Libre.

Elige que opción deseas realizar:
    ...
```

Fig. 1: Ejemplo de funcionamiento del cliente

Este cliente, le mostrará un menú al usuario con las posibles operaciones que se podrán realizar (de acuerdo a las implementadas en el servicio), solicitará los parámetros necesarios y realizará la llamada al servicio.

Posteriormente, mostrará la información al usuario (ver Figura 1).

4. Cree un servicio web de directorio similar al Directorio de la Universidad de Cádiz.

El servicio deberá contar con un listado (evidentemente ficticio) del personal de la UCA, y cada miembro contará con los siguientes atributos:

- DNI
- Nombre completo
- Correo electrónico
- Departamento
- Categoría (a elegir entre PAS/PDI/becario)
- Lista de asignaturas (solo si es PDI)

El servicio web deberá contar, al menos, con los siguientes `endpoints`:

a) Dar de alta un miembro nuevo.

b) Modificar los datos de un miembro.

c) Consultar la lista de todos los miembros de la Universidad.

d) Hacer una búsqueda por DNI.

e) Obtener una lista de miembros según categoría.

Queda a juicio del estudiante identificar qué tipo de operaciones HTTP debe implementar cada `endpoint` (`PUT`, `POST` o `GET`).

Para los endpoints b), d) y e), es obligatorio el uso de parámetros en el `path` de la `URL`.

Aunque hemos visto que REST permite transmitir los datos en cualquier formato, se recomienda utilizar JSON, tal y como se ha visto en el seminario.

Se valorará positivamente el manejo adecuado de los errores (parámetros incorrectos, `endpoint` no encontrado, operaciones no permitidas) en el servicio y su notificación al cliente.

Adicionalmente, implemente la siguiente funcionalidad:

- Crear un endpoint para eliminar un miembro mediante una operación `HTTP DELETE`.

- Utilizar alguna forma de persistencia de datos, almacenando la información de los miembros en un fichero o en una base de datos.
- Habilitar una búsqueda parcial por nombre.
- Habilitar una búsqueda paramétrica en la que se pueda elegir por qué criterio buscar.
- Implementar una búsqueda inversa por asignaturas, listando los miembros PDI que pertenezcan a una asignatura.
- Verificar que los datos introducidos al añadir un nuevo miembro son correctos, como por ejemplo la letra del DNI.
- Otros `endpoints` adicionales que el estudiante considere oportunos.

Por último, se debe crear un cliente que permita a un usuario interactuar con la API desarrollada.

Este cliente, le mostrará un menú al usuario con las posibles operaciones que se podrán realizar (de acuerdo a las implementadas en el servicio), solicitará los parámetros necesarios y realizará la llamada al servicio.

Posteriormente, mostrará la información al usuario (ver Figura 2).

```
Elige qué opción desea realizar:
    1. Dar de alta un nuevo miembro en el directorio.
    2. Modificar los datos de un miembro.
    3. Consultar la lista de todos los miembros de la Universidad.
    4. Hacer una búsqueda por DNI.
    5. Consultar los miembros según categoría.
    6. Salir

> 4

Introduce el DNI del miembro

> 12345678Z

Miembro con DNI 12345678Z:
    - Juan García Pérez
    - juan.garciaperez@uca.es
    - Departamento de Química Orgánica
    - PDI
    - Asignaturas: Química Orgánica
```

Fig. 2: Ejemplo de funcionamiento del cliente

5. Desarrollar con Bottle un servicio web que gestione el registro de usuarios de un sistema genérico (llamado “registroUsuarios.py”). El servicio web debe mantener una base de datos en memoria (puede estar vacía al arrancar el servicio) donde almacenar toda la información de los usuarios. Cada usuario contendrá los siguientes atributos (el alumno debe decidir qué tipo de datos usar para cada uno):

    - Nombre de usuario
    - Contraseña (puede almacenarse sin cifrar)
    - Cuenta activada (valor booleano)
    - Correo electrónico
    - Nombre y apellidos

    Se deben implementar los siguientes endpoints:

    - Registro de usuario: se añade un nuevo usuario indicando los atributos anteriormente indicados. El atributo de cuenta activada deberá establecerse como “falso”. No se podrá registrar un usuario si su nombre de usuario o correo electrónico ya están registrados en la base de datos.
    - Activación de cuenta: dado un nombre de usuario o correo electrónico (que debe estar previamente registrado), se modificará la entrada de ese usuario en la base de datos, estableciendo como “verdadero” el atributo de cuenta activada. Es decir, el usuario debe existir previamente y el atributo de cuenta activada debe contener el valor “falso”.
    - Búsqueda de usuarios: dada una cadena de texto, se devolverán todos los usuarios que contengan esa cadena en el nombre de usuario o en el correo electrónico.

    Por último, se debe desarrollar una aplicación de terminal que sirva como cliente y tenga opciones de usar cualquiera de los endpoints descritos anteriormente.

    Se valorará negativamente una incompleta gestión de errores.

6. Desarrollar con Bottle un servicio web que gestione un servicio de alquiler de vehículos. El servicio web debe mantener dos bases de datos en memoria (pueden estar vacías al arrancar el servidor) donde almacenar toda la información de los vehículos y los clientes activos (con vehículos alquilados). Cada vehículo contendrá los siguientes atributos (el alumno debe decidir qué tipo de datos usar para cada uno):

- ID vehículo
- Marca
- Modelo
- Estado de alquiler (alquilado o no alquilado)
- Actual o último usuario que lo ha alquilado

Cada usuario contendrá los siguientes atributos (el alumno debe decidir qué tipo de datos usar para cada uno):

- ID usuario
- Número de veces que ha alquilado un vehículo

Se deben implementar los siguientes endpoints (el alumno debe decidir qué tipo de método usar en cada caso):

- Registro de usuario: se añade un nuevo usuario asignándole automáticamente un ID único e inicializando el número de alquileres a 0.
- Registro de vehículo: se añade un nuevo vehículo a la flota disponible para alquilar. Hay que asignarle automáticamente un ID único, y pasarle la información de marca y modelo mediante un JSON. Se debe inicializar el estado del alquiler a no alquilado.
- Alquiler de un vehículo libre: dado un ID de usuario, localizar un vehículo cualquiera libre y asignárselo al usuario. El alumno debe actualizar el campo estado de alquiler del vehículo y el campo número de veces que ha alquilado un vehículo, el usuario.
- Alquiler de un vehículo específico: dados los IDs de usuario y vehículo, asignar el vehículo al usuario. El alumno debe actualizar el campo estado de alquiler del vehículo y el campo número de veces que ha alquilado un vehículo, el usuario.

El alumno debe gestionar los posibles errores que puedan ocurrir al usar estos endpoints (por ejemplo, que se quiera alquilar un vehículo, pero no queden vehículos libres).

7. Desarrollar con Bottle un servicio web para la gestión de un sistema de reserva de pistas de paddle (implementado en un fichero llamado "reservaPistas.py"). Las entradas y salidas de la API deberán hacerse en formato JSON. El servicio web debe mantener una base de datos en memoria (puede estar vacía al arrancar el servicio) donde almacenar toda la información de la reserva de pistas. Cada reserva de pista deberá contar con la siguiente información:

- Identificador de reserva (generado por el sistema).
- Número de pista reservada.
- Hora de inicio (de 7 a 22, no hace falta guardar los minutos).
- Hora de fin (de 8 a 23, no hace falta guardar los minutos).
- Nombre del jugador que ha hecho la reserva.

Se deben implementar los siguientes `endpoints`:

- `/reservar`: deberá permitir añadir una nueva reserva para una pista concreta. Si la reserva es correcta, el sistema generará un identificador aleatorio que devolverá al usuario. En caso contrario, se deberá cancelar el proceso e informar al usuario si la pista ya está reservada para esa franja horaria.
- `/cancelar`: dado un identificador de reserva, se eliminará la reserva asociada al identificador, quedando la pista liberada para esa franja horaria. En caso de que no exista ninguna reserva para ese identificador, deberá informarse al usuario del error.
- `/mostrar`: dado un número de pista, el sistema deberá mostrar en qué franjas horarias la pista está ocupada y por quién. El formato del JSON de retorno queda a juicio del alumno, pero deberá ser suficiente para que un cliente pudiese parsearlo para mostrar un listado con un aspecto similar al siguiente:

```
Estado de la pista número 5:

      - 9:00 a 13:00: reservada por Juan

      - 18:00 a 20:00: reservada por Alfonso
```

Notas:

- El sistema gestionará un número N de pistas configurable.
- Las reservas se hacen por horas y pueden durar una o más horas.
- Las pistas se pueden reservar a partir de las 7:00, siendo la última hora reservable de 22:00 a 23:00.
- El identificador de reserva deberá ser aleatorio.

8. Desarrollar con Bottle un servicio web en un fichero (llamado aeropuerto.py) para la gestión de llegadas (aterrizajes) y salidas (despegues) de aviones de un aeropuerto con dos pistas numeradas 10 y 30. El servicio web debe mantener una base de datos en memoria (puede

estar vacía al arrancar el servicio) donde almacenar toda la información de los aviones que han pasado por el aeropuerto. Cada avión contendrá los siguientes atributos (el alumno debe decidir qué tipo de datos usar para cada uno):

a. Matrícula (asignada manualmente).

b. Fecha y hora de llegada.

c. Número de pista de llegada.

d. Fecha y hora de salida.

e. Número de pista de salida.

Se deben implementar los siguientes endpoints:

- Registro de aterrizaje: se añade un nuevo avión al aeropuerto indicando la hora de aterrizaje y dejando vacíos (None) los atributos d) y e). El número de pista de llegada, asignada automáticamente, deberá ser aquella que aparezca con menor frecuencia en los aviones de la base de datos que siguen en tierra (aquellos con b) y c) rellenos, y None en d) y e)). El alumno puede elegir cualquier criterio de asignación de pista cuando no haya aviones en la base de datos.

- Registro de despegue: indicando la matrícula del avión (que previamente debe estar en tierra) y una hora de salida, el número de pista de salida se asigna automáticamente siguiendo el mismo criterio que en el `endpoint 1`.

- Listado de todos los aviones del aeropuerto y el valor de sus atributos.
