MTIS	Metodologías y Tecnologías de Integración de Sistemas
	Práctica 2
	Diseño Api Rest: OpenApi

Preámbulo

Una de las tareas más importantes a la hora de crear una SOA es definir su modelo de servicios: o sea, qué servicios hay y qué tareas en concreto hace cada uno. En una SOA basada en servicios web tipo Rest debemos de poder diseñar nuestros servicios al igual que hacíamos con los servicios web tipo SOAP mediante los contratos WSDL, para poder definir en un documento todas las funcionalidades que presenta nuestra API Rest.

La especificación OpenApi hace que sea fácil administrar todo el ciclo de vida de la API, desde el diseño hasta el uso compartido. Es conciso, solo se escribe lo que se necesita definir y es reutilizable. Es un diseño API legible por máquina y amigable para los humanos.

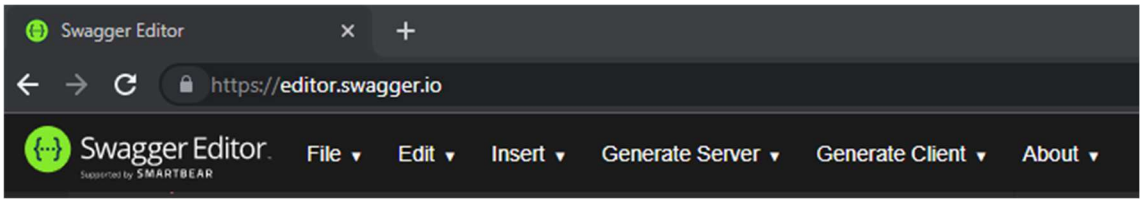
Objetivo

El Objetivo de esta práctica es documentar una serie de servicios mediante OpenApi, sin necesidad de crear un servicio web. Posteriormente crearemos los servicios web asociados basándonos en los documentos OpenApi y las herramientas de swagger para tal fin y por último generaremos un cliente mediante las herramientas de codegen de swagger.

Pasos a realizar

Para el empleo de la especificación OpenApi, haremos uso de las herramientas que presenta swagger para el diseño, generación de código y visualización de documentos, en concreto de la utilidad swagger editor:

https://editor.swagger.io/



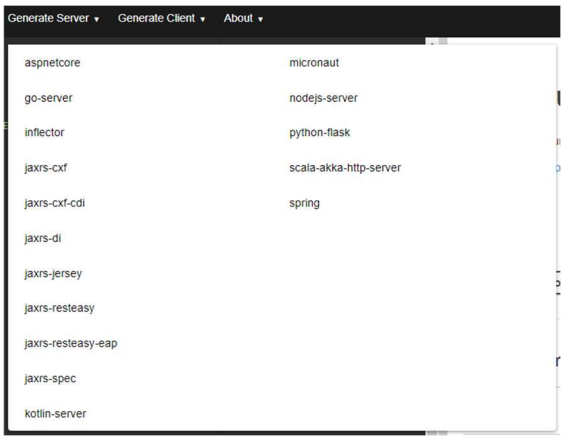
Mediante esta herramienta podremos crear especificaciones OpenAPI, visualizar ejemplos (Petstore OAS 3.0), generar código para servidor, generar código para cliente, etc…

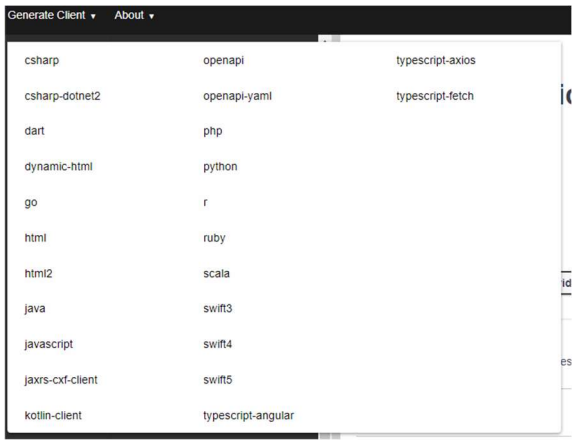
A continuación realizaremos el siguiente ejemplo para familiarizarnos con el entorno:

```
openapi: 3.0.0
servers:
  # Comentario
  - description: Servidor Apis DTIC
    url: http://dtic.org/HolaMundo/1.0.0
info:
  description: Esto es un Hola Mundo
  version: "1.0.0"
  title: Hola Mundo Dtic
  contact:
    email: info@dtic.org
  license:
    name: Apache 2.0
    url: 'http://www.apache.org/licenses/LICENSE-2.0.html'
tags:
  - name: tag hola mundo dtic
    description: Desarrollado en dtic.ua.es
paths:
  /holamundo:
    get:
      description:  GET Hola Mundo
      parameters:
        - in: query
          name: nombre Hola Mundo
          description: nombre Hola Mundo
          required: true
          schema:
            type: string
      responses:
        '200':
          description: Hola Mundo Correcto
          content:
            application/json:
              schema:
                type: object
                properties:
                  nombre:
                    type: string
                  contador:
                    type: integer
        '400':
          description: Hola Mundo Error
          content:
            application/json:
              schema:
                type: object
                properties:
                  message:
                    type: string
    post:
      description: POST Hola Mundo
      parameters:
        - in: query
          name: nombre Hola Mundo
          description: nombre Hola Mundo
          required: true
          schema:
            type: string
      responses:
        '200':
          description: Hola Mundo POSTCorrecto
        '400':
          description: Hola Mundo Error
          content:
            application/json:
              schema:
                type: object
                properties:
                  message:
                    type: string
```

Quedando la vista de dicho documento

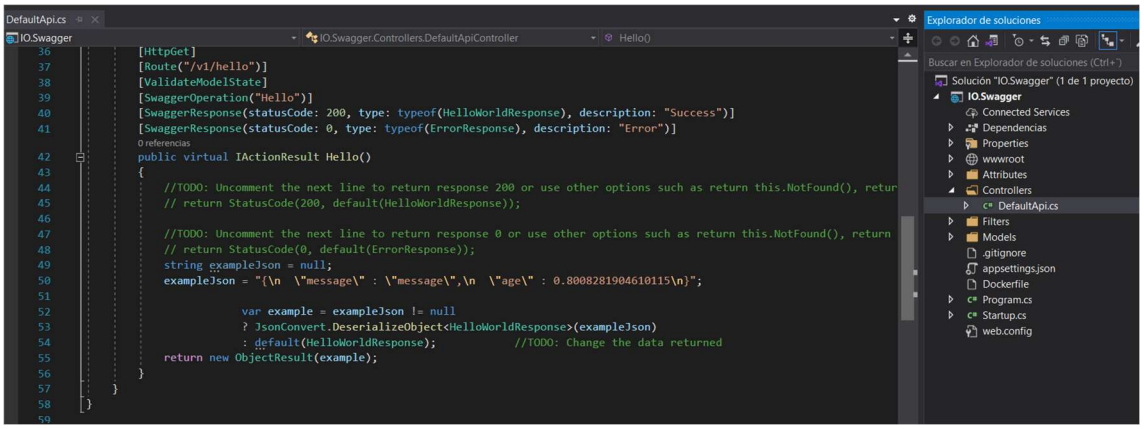
Finalmente, mediante las herramientas de generación de código que incorpora:





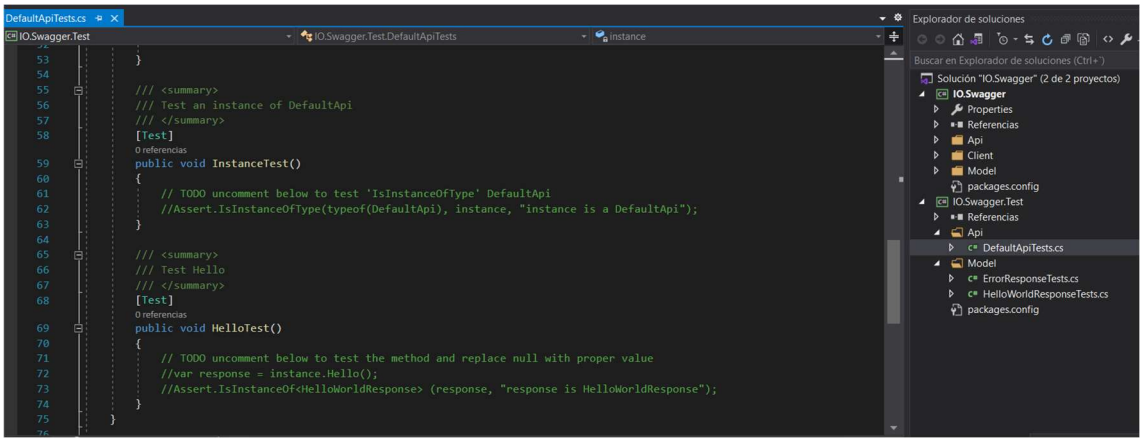
Podremos generar entre otros, el esqueleto del servicio web y el cliente entre los diversos frameworks propuestos, para finalmente añadir la lógica de negocio a dicho esqueleto.

En el ejemplo propuesto *Hola Mundo*, hemos generado mediante *Generate Server* api → *aspnetcore* el servicio web en asp .net core:



Faltaría completar, como se puede ver la lógica de negocio correspondiente.

Para el cliente al igual que en el paso anterior, podemos generar mediante *codegen* el esqueleto correspondiente, por ejemplo para c#:



En esta solución generada podemos ver un proyecto IO.Swagger el cual se encargará de llamar a nuestra Api Rest y otro IO.Swagger.Test el cual es un proyecto de pruebas donde podemos comprobar el funcionamiento de la Api Rest.

Enunciado

1. Se deberán crear los siguientes documentos mediante la especificación OpenApi que contengan las siguientes funcionalidades:

Prosiguiendo con el edificio inteligente, en esta práctica crearemos una serie de servicios web para gestionar dicho edificio con nuevas funcionalidades.

Nuestros Servicios Web se basarán en el mismo esquema de BD que la práctica 1.

El primer servicio web se denominará **Salas** y debe de proporcionar los siguientes métodos:

nuevo, el cual creará una nueva sala en BD, a partir de la estructura indicada en BD.

borrar, se encargará de borrar de BD, el registro indicado por el *codigoSala*.

modificar, se encargará de actualizar en BD el registro, se le pasará la estructura completa que define una sala y devolverá un *booleano* para indicar si ha sido correcta la actualización del registro en BD.

consultar, devolverá a partir de un *codigoSala* la estructura completa de una sala.

El segundo servicio web se denominará **Niveles** y debe de proporcionar los siguientes métodos:

nuevo, el cual creará un nuevo nivel en BD, a partir de la estructura indicada en BD.

borrar, se encargará de borrar de BD, el registro indicado por el *nivel*.

modificar, se encargará de actualizar en BD el registro, se le pasará la estructura completa que define un nivel y devolverá un *booleano* para indicar si ha sido correcta la actualización del registro en BD.

consultar, devolverá a partir de un *nivel* la estructura completa de un nivel.

El tercer servicio web se denominará **Dispositivos** y debe de proporcionar los siguientes métodos:

nuevo, el cual creará un nuevo dispositivo en BD, a partir de la estructura indicada en BD.

borrar, se encargará de borrar de BD, el registro indicado por el *codigo*.

modificar, se encargará de actualizar en BD el registro, se le pasará la estructura completa que define un dispositivo y devolverá un *booleano* para indicar si ha sido correcta la actualización del registro en BD.

consultar, devolverá a partir de un *codigo* la estructura completa de un dispositivo.

El cuarto servicio web se denominará **Notificaciones** y debe proporcionar los siguientes métodos:

NotificarPresenciaSala, notificará por email a los empleados que se encuentran presenten en alguna sala indicando el nombre de la sala, obteniendo los valores de las tablas *controlpresencia, empleados y salas*.

NotificarUsuarioValido, notificará por email a un empleado si es válido, obteniendo los valores de la tabla *empleados*. Como parámetro de entrada recibirá el *nif* del *empleado*.

NotificarError, notificará por email a un empleado de un error, siendo parámetros de entrada: *error* (string) y *nif* del *empleado*.

Para todos los métodos la dirección de email la obtendremos del campo *email* de la tabla *empleados*.

Consideraciones a tener en cuenta:

Necesitaremos disponer de la Base de Datos de la práctica 1 en MySql.
Utilizaremos el servidor de pruebas fakeSMTP para comprobar el envío de emails desde lo servicios web.

2. **A partir de los documentos OpenApi creados, generar los servicios web, mediante un framework de los disponibles en el codegen de swagger editor.**

Para la comprobación del correcto funcionamiento de los servicios web, emplearemos las herramientas: SoapUI u otras tipo Postman.

3. **Se deberá crear una aplicación cliente, mediante la herramienta codegen de swagger editor (el framework seleccionado será de libre elección), para la comprobación de dichos servicios.**
4. **A la hora de valorar la práctica para la máxima nota, se tendrán en cuenta las siguientes consideraciones:**

Todos los servicios contendrán un parámetro adicional de entrada, a los especificados a continuación, el cual será un *string*, llamado ***RestKey***, este contendrá una clave, la cual deberemos validar que existe en nuestra Base de Datos, para realizar o no, el método solicitado.

Adicionalmente a la salida, todos los métodos contarán con un parámetro extra (string), para notificar posibles errores.

Entrega

Se deberá de entregar por el control creado para tal fin en Moodle:

- Una pequeña memoria detallando la puesta en marcha y aspectos a destacar de la práctica como puedan ser las implementaciones para la máxima nota.
- Contratos OpenApi y proyectos servicios web y proyecto cliente.

La fecha límite de entrega será el 07.03.2023. La práctica se corregirá el 08.03.2023 en horario de prácticas. En caso de entregar la práctica en una fecha posterior, únicamente se podrá optar a un 5 como nota máxima, siendo la fecha límite para cualquier entrega el 24.05.2023.