

## API Rest con Spring

Se pide desarrollar una aplicación REST API en Java que conecte a **H2**, una base de datos relacional embebida, usando **Spring boot + Spring Data JPA + Hibernate**.

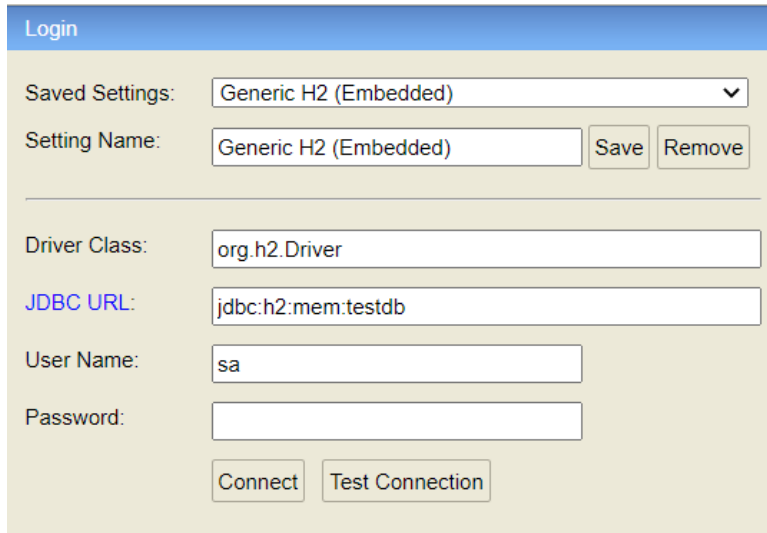
La aplicación se lanzará en **localhost:8081**, y lo primero que aparezca en el path root "/" deberá ser una página html con documentación acerca de la interfaz REST que implementa, es decir, métodos de acceso a ella, y que hace cada uno de ellos con la herramienta que utilizamos en clase. Deberá estar detallada claramente puesto que el usuario no tiene ni idea.

Los métodos implementados en la interfaz REST recibirán y enviarán peticiones en formato **JSON**.

Para la temática de la aplicación se deberá elegir una basada en una API real.

### Requisitos mínimos (para llegar al 6)

- 1.- Se creará un proyecto con Spring y gradle en IntelliJ con la estructuración que hemos usado en clase. Entidades, Servicios, Repositorios y controladores Rest.
- 2.- La aplicación deberá conectar a la base de datos H2 con la siguiente configuración:



- 3.- Utilizará el fichero **data.sql** para insertar datos reales. Para generar datos de prueba reales puedo utilizar la web <https://www.mockaroo.com/> es muy sencillo y los genera en formato sql, json, etc.
- 4.- La base de datos deberá tener como mínimo 4 tablas y relaciones entre éstas.
- 5.- La API REST debe implementar endpoints similares a los siguientes ejemplos utilizando JSON:

❖ Listar datos (2 métodos mínimo)

GET: **/flights** → muestra una lista de todos los vuelos.

GET: **/airports** → muestra una lista de todos los aeropuertos.

❖ Información paginada u ordenada (1 método mínimo)

GET: **/passengers-page?page=nPage&size=nItems&sortBy=name** → Devuelve la lista de pasajeros paginada y ordenada por nombre (número de página y tamaño de ítems por página)

❖ Añadir, actualizar y borrar información (1 método de cada)

## UD4. Acceso a datos en apps REST

POST: `/passenger-list` → añade una lista de pasajeros

POST: `/airports` → añade un nuevo aeropuerto

PUT: `/change-flight` → actualiza el vuelo

DELETE: `/flight/{id}/{passengerId}` → borra al pasajero de ese vuelo

- ❖ Extraer información del sistema (métodos con parámetros en la url utilizando las dos formas vistas en clase, con path y request param, consultas de interés para el sistema y consultas que involucren más de una tabla). (6 métodos mínimo)

GET: `/flights/cheapest` → devuelve el vuelo más barato.

GET: `/airports/{airline}` → dada una aerolínea devolver la lista de aeropuertos en los que tiene vuelos.

GET: `/flights/{airport}/{airline}` → devuelve una lista de vuelos pertenecientes a un aeropuerto dado y a una aerolínea dada.

GET: `/flights/occupation/{flightNumber}` → devuelve el número de asientos ocupados para ese número de vuelo.

GET: `/flights?date=20210115` → devuelve todos los vuelos pertenecientes a esa fecha.

GET: `/flights/arrival?airport=code&arrival=arrivalDate&begin=beginHour&end=endHour`

→ devuelve los vuelos de llegada de un aeropuerto y un día entre dos horas.

Ejemplo: `/flights/arrival?airport=YYZ&arrival=20210128&begin=12&end=21` devuelve los vuelos del aeropuerto de Toronto con llegada el 28/01/2021 entre 12 y las 21.

### Funcionalidades extra (+ 1 punto)

- ❖ Guardar información del sistema (1 método)

GET: `/flights/today/file-json` → obtiene los vuelos para hoy y los almacena en un fichero json. Devuelve el string donde se ha creado el fichero.

- ❖ Estadísticas (1 método mínimo)

GET: `/statistics/flights/{airport}` → obtiene la cantidad de vuelos que se han realizado en ese aeropuerto.

### Funcionalidades extra (+1 punto)

Utiliza Thymeleaf para crear páginas html en las que el usuario pueda interactuar. Incluye acciones para añadir, ver, actualizar y eliminar datos de alguna entidad.

### Funcionalidades extra (+ 2 puntos)

Crea una aplicación cliente para conectar con el backend utilizando react.

### Entrega

Entrega el proyecto en un .zip, que incluya:

- Todos los .java y fichero de configuración.
- El .jar de la aplicación para poder ejecutarla.

### Consideraciones

- Cualquier proyecto entregado que no incluya los requisitos mínimos estará suspenso.
- Si al ejecutar el fichero .jar se obtiene algún tipo de excepción no controlada por el alumno, se considerará suspendido. Así que probar todo bien.
- **NO SE ACEPTAN ENTREGAS FUERA DEL PLAZO.**
- **FECHA DE ENTREGA domingo 19 de febrero de 2023.**