# Servicio REST

El servicio Rest es un servidor en el que existen una serie de métodos implementados que nos ayudarán a mostrar la información contenida en la base de datos.

Inicialmente en el servicio Rest es la conexión con la base de datos, mediante el uso de la sentencia mySQLClient que contendrá el verticle desplegado y una serie de configuraciones, entre las cuales se encuentra el host, que en este caso sería localhost, un usuario, una contraseña, el nombre de la base de datos y el puerto en el que escucha.

## Métodos GET:

1. handlerGetTablaPeriodica();

Éste método es el encargado de tomar toda la información de la base de datos referente a todas las tablas “TablaPeriodica”, donde en ella se encuentra un identificador “oid\_tabla\_periodica”, de la propia tabla que será un tipo INT y de manera autoincrementar por ser la PK de la tabla. Y “nombre\_tabla\_periodica”, que como se explicó anteriormente, sirve para que el usuario tome la decisión de escoger la tabla física en la que quiere interactuar, siendo ésta de tipo String y un valor único, es decir, que no se pueda repetir.

Para ello creamos una lista vacía de tipo TablaPeriodica(Entitie), se recorrerá mediante un for incluyendo todos los Json que se reciben. Además, hemos implementado una serie de sentencias que gestionan los errores cometidos, así como el cierre de conexión tras obtener el resultado.

1. handlerGetTablaPeriodicaNombre();

Éste método es un extendido del anterior, es decir, realiza exactamente lo mismo que handlerGetTablaPeriodica(), pero aquí aplicamos un filtro que consiste en mostrar toda la información de **una sola** tabla periódica correspondiente al “nombre\_tabla\_periodica” que especificamos en la URL.

1. handlerGetTodosUsuarios();

Este GET toma toda la información acerca de todos los usuarios desde la tabla “Usuarios”, donde se encuentra un “oid\_usuario” de tipo INT y de manera autoincremental por ser la PK de esta tabla. También un “nombre\_usuario” de tipo String y único, mediante el cual un usuario se registrará en la aplicación web con ese nombre y se guardará en la base de datos. Además, hay un “nombre\_tabla\_periodica” de tipo String y único, que se especifica el nombre de la tabla física con la que se interactuará y se elegirá también en la aplicación web.

Como explicamos en el primer GET, todos realizan lo mismo, una lista que se recorre y se guarda los Json que se reciben.

1. handlerGetUsuarioNombre();

Este es el extendido de handlerGetTodosUsuario(), en el que se le aplica el filtro por un “nombre\_usuario” especificado en la URL, y mostrará la información de **un solo** usuario registrado.

1. handlerGetTodosElementos();

Este método GET toma toda la información relacionada con todos los elementos contenidos de una tabla periódica, es decir, mostrará información de todos los elementos. Contiene un “oid\_elemento” de tipo INT y de manera autoincremental al ser la PK de esta tabla. También un “nombre\_elemento” de tipo String, que hará referencia al nombre del elemento que busquemos, un “símbolo\_elemento” de tipo String que hace referencia a su símbolo, un “numero\_atomico” de tipo INT que tiene el valor del peso atómico del elemento, una “url\_video” de tipo String que será una url del video que se mostrará por pantalla al elegir un elemento específico y un “nombre\_tabla\_periodica” de tipo String que hará referencia al nombre de la tabla física.

El procedimiento es el mismo que los anteriores.

1. handlerGetElementoNombre();

Este método realiza filtro de los elementos pasándole un “nombre\_elemento” a la URL, así escogerá **solo** el elemento solicitado.

1. handlerGetTodasOperaciones();

Este método recoge información sobre todas las operaciones realizadas en la tabla periódica, es decir, sobre los elementos pulsados en la tabla. Contiene un “oid\_operacion” de tipo INT y de manera autoincremental por ser PK de la tabla. También un “nombre\_tabla\_periódica” de tipo String, que como hemos explicado antes, sirve para referenciar la tabla física. También un “nombre\_elemento” de tipo String que será el elemento que solicitemos al pulsar un botón específico. Y por último un “nombre\_usuario” de tipo String que se registrará quien ha realizado dicha operación.

El procedimiento es el mismo que los anteriores.

1. handlerGetOperacionesNombreUsuario();

Mediante éste método, recogemos la información de **una sola** operación, pasándole como filtro un “nombre\_usuario” en la URL, para determinar las operaciones que ha realizado un determinado usuario.