**Tutorial Oracle Cloud: DataBase Actions**

Ciclo 3: Desarrollo de Software

AÑO 2022

**Introducción**

El proceso de aprendizaje con la nube PaaS de Oracle presta como alternativa, la posibilidad de implementar de manera rápida el acceso a la información de una base de datos a través de la creación *no code* de web services. Esta funcionalidad representa una oportunidad de aprendizaje per se, pero además permite aproximarnos al concepto de servicio web y posteriormente de API. Con este documento se explicará paso a paso la utilización de la función DataBase Actions para exponer servicios.

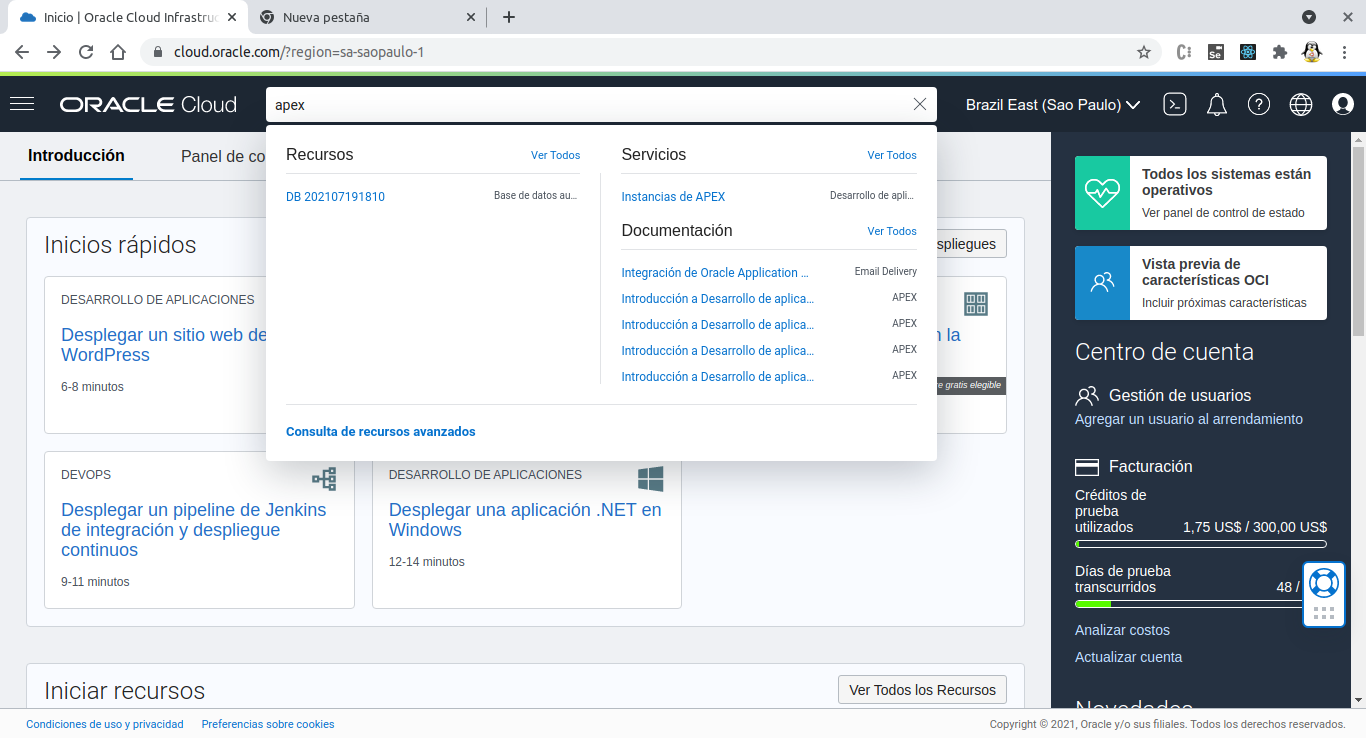
# **Database Actions**

Oracle Cloud como proveedor PaaS (plataforma como servicio) ofrece una variedad de herrramientas que serán muy útiles a la hora de abordar el curso. Para ello iniciaremos con el servicio APEX, el cual nos permite realizar el despliegue rápido de servicios solamente con un par de clics.

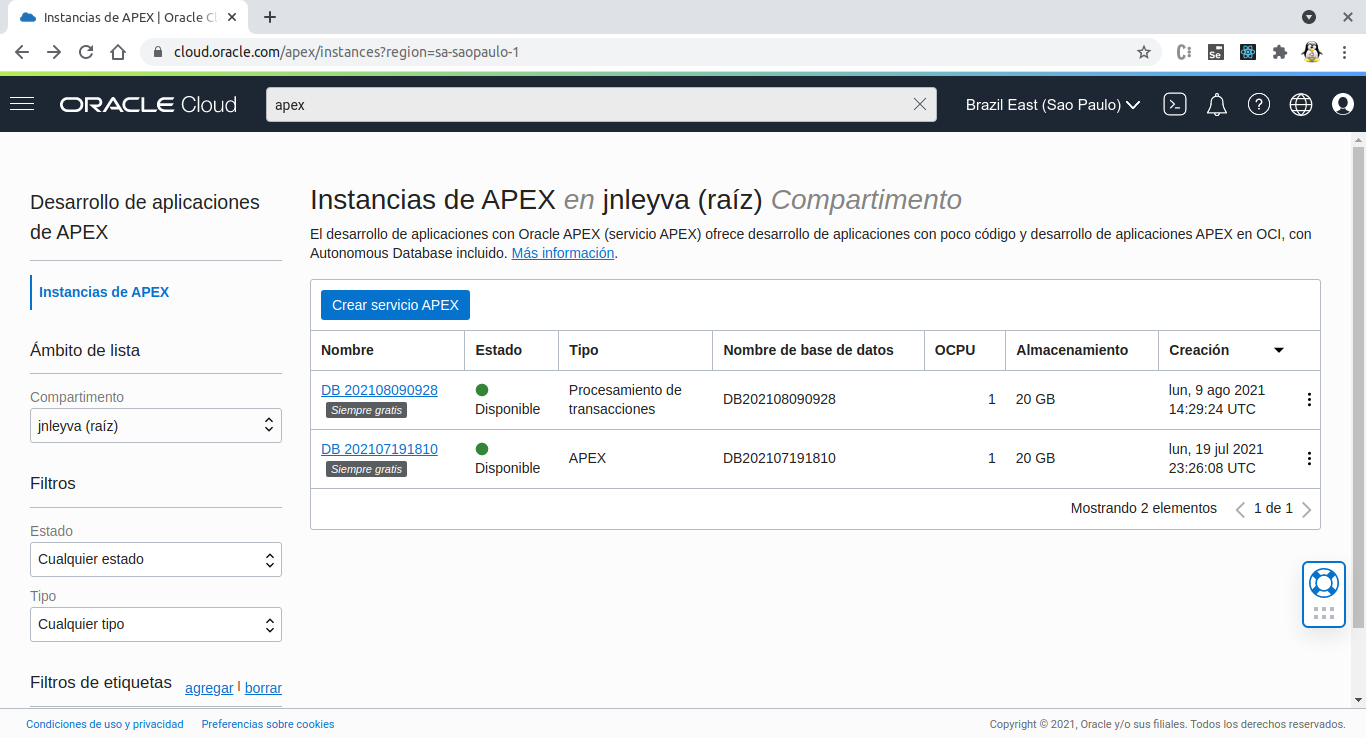
Para este ejercicio ingresaremos al portal de Oracle Cloud

https://www.oracle.com/co/cloud/sign-in.html

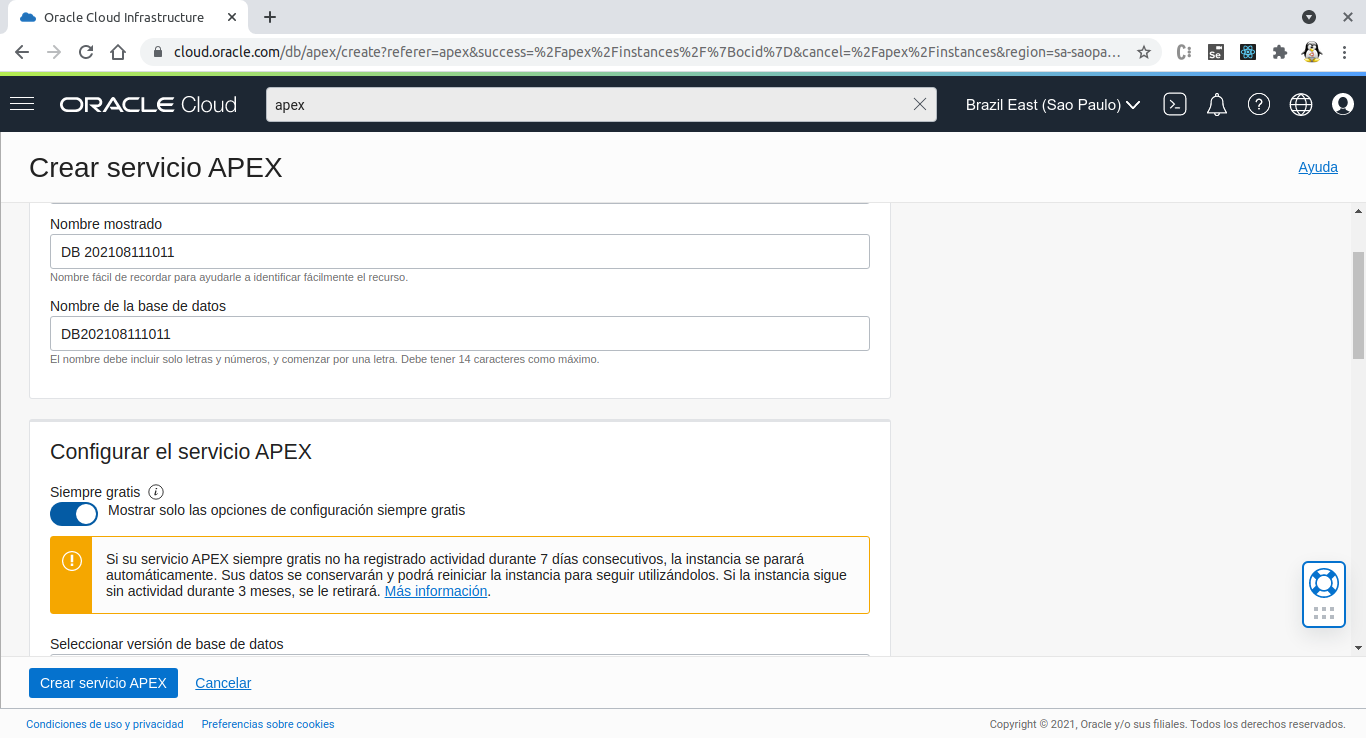
después de autenticarnos, entraremos a la Sección Oracle APEX, para ello se puede escribir en la barra de búsqueda superior APEX y Seleccionar *Instancias de APEX* bajo la sección de *Servicios*

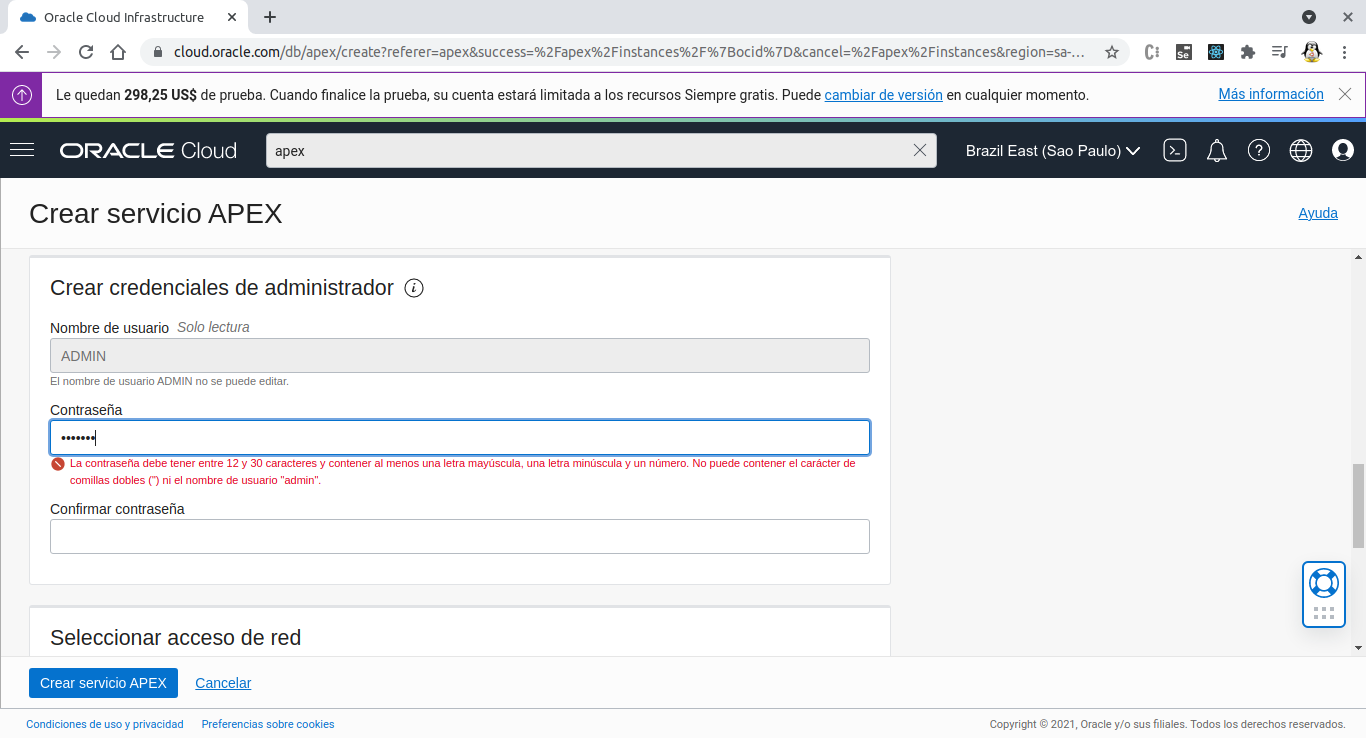


Una vez allí, haremos clic sobre el botón Crear Servicio APEX

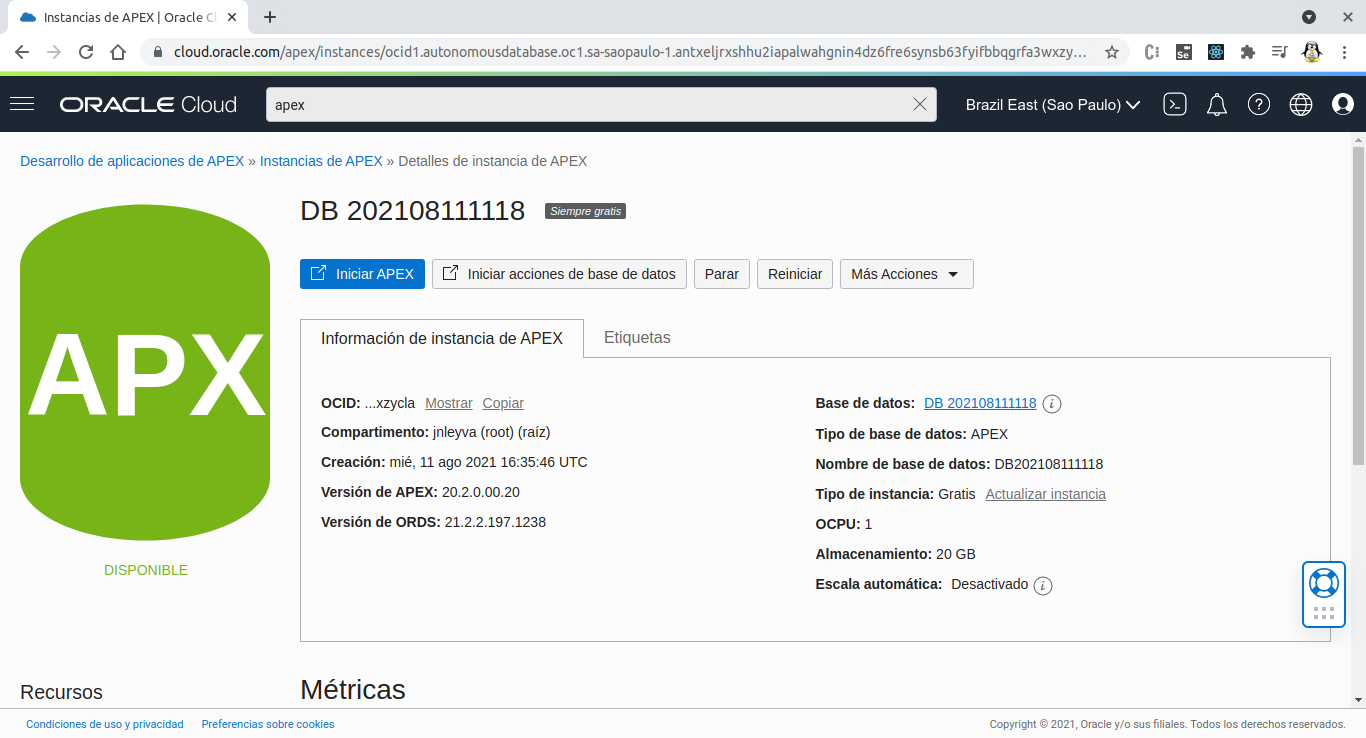


Es importante seleccionar a continuación la opción *Siempre Gratis* de esa manera podemos hacer uso de recursos que no descontarán los créditos iniciales. Para aplicaciones que requieran mejores prestaciones siempre será posible elegir otro tipo de configuración.

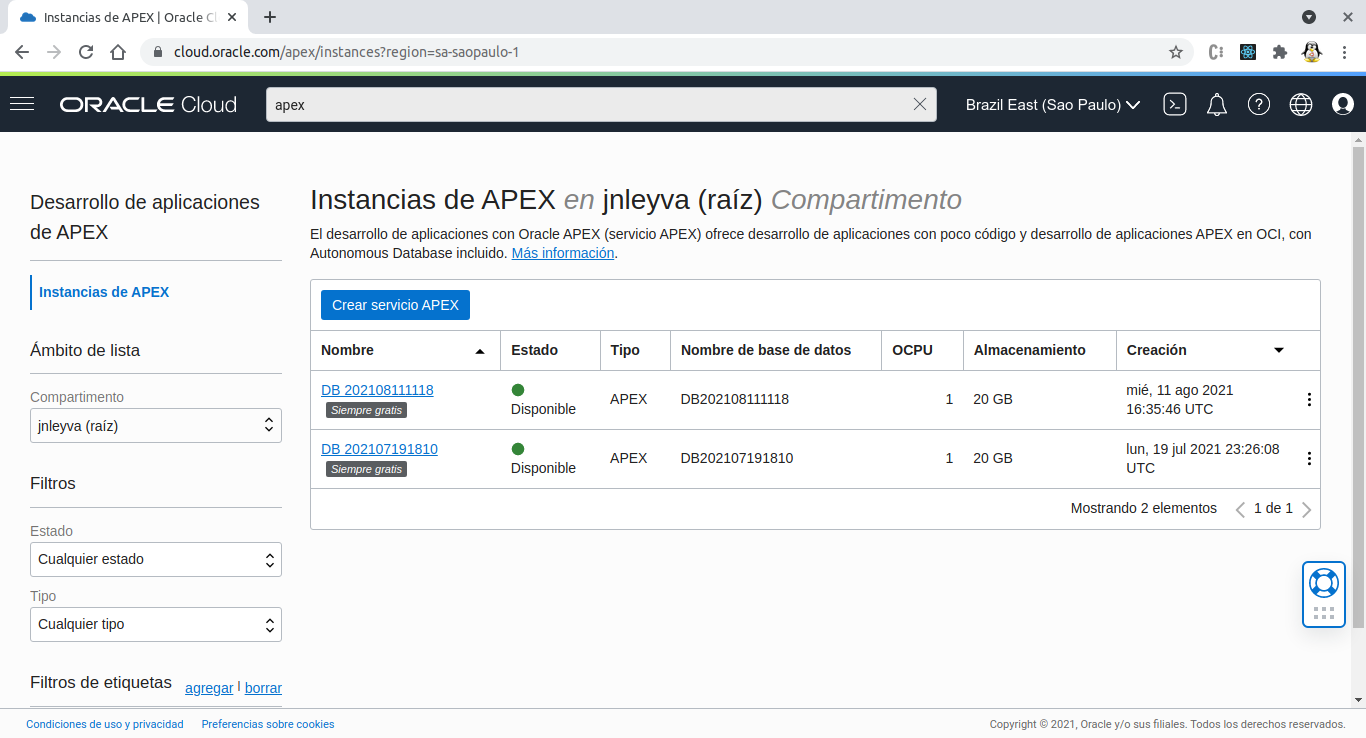
Una vez seleccionada la opción siempre gratis, resta establecer la contraseña y repetirla. **Es muy importante no olvidarla** debido a que será necesaria cada vez que necesitemos acceder a esta funcionalidad. Hay que tener en cuenta que el usuario siempre será ADMIN y que la contraseña debe cumplir con las normas establecidas. Una vez establecida la contraseña y confirmada hacer clic en el botón *Crear Servicio APEX.*



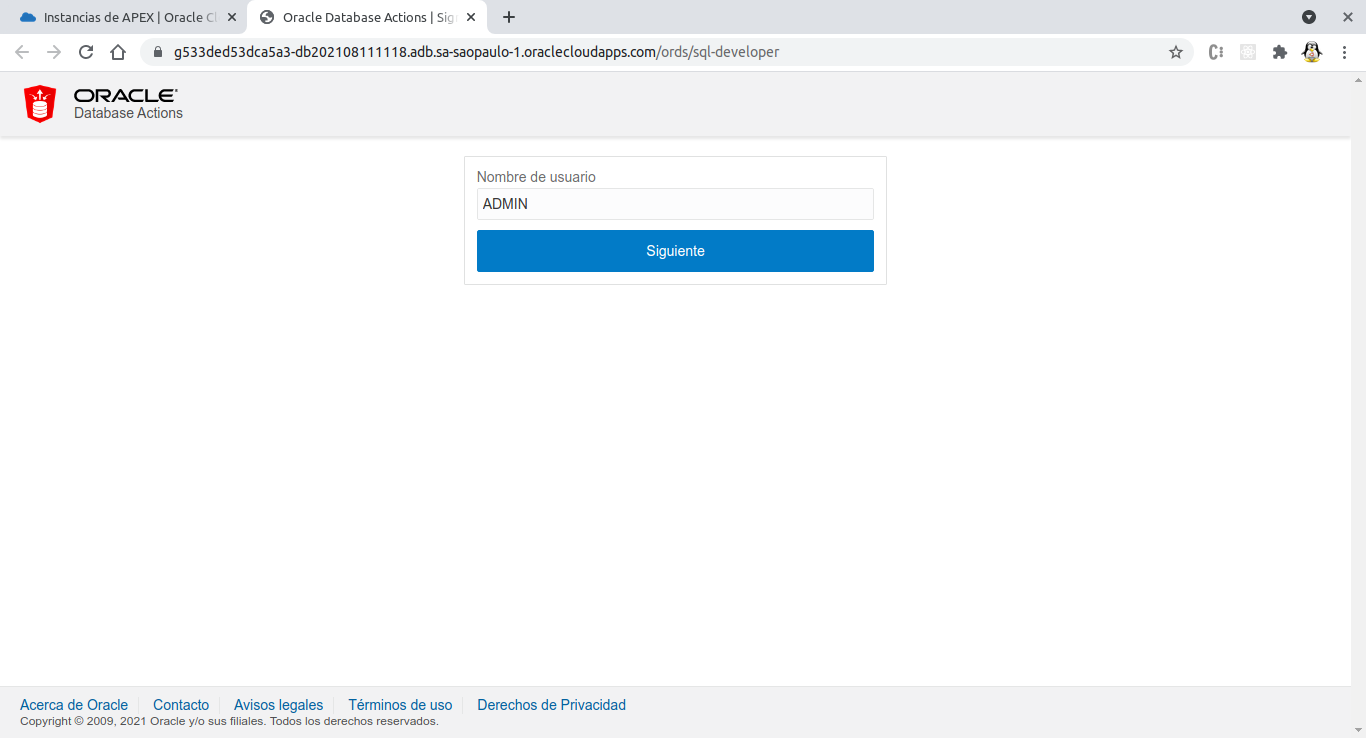
Una vez creado se verá la siguiente pantalla:



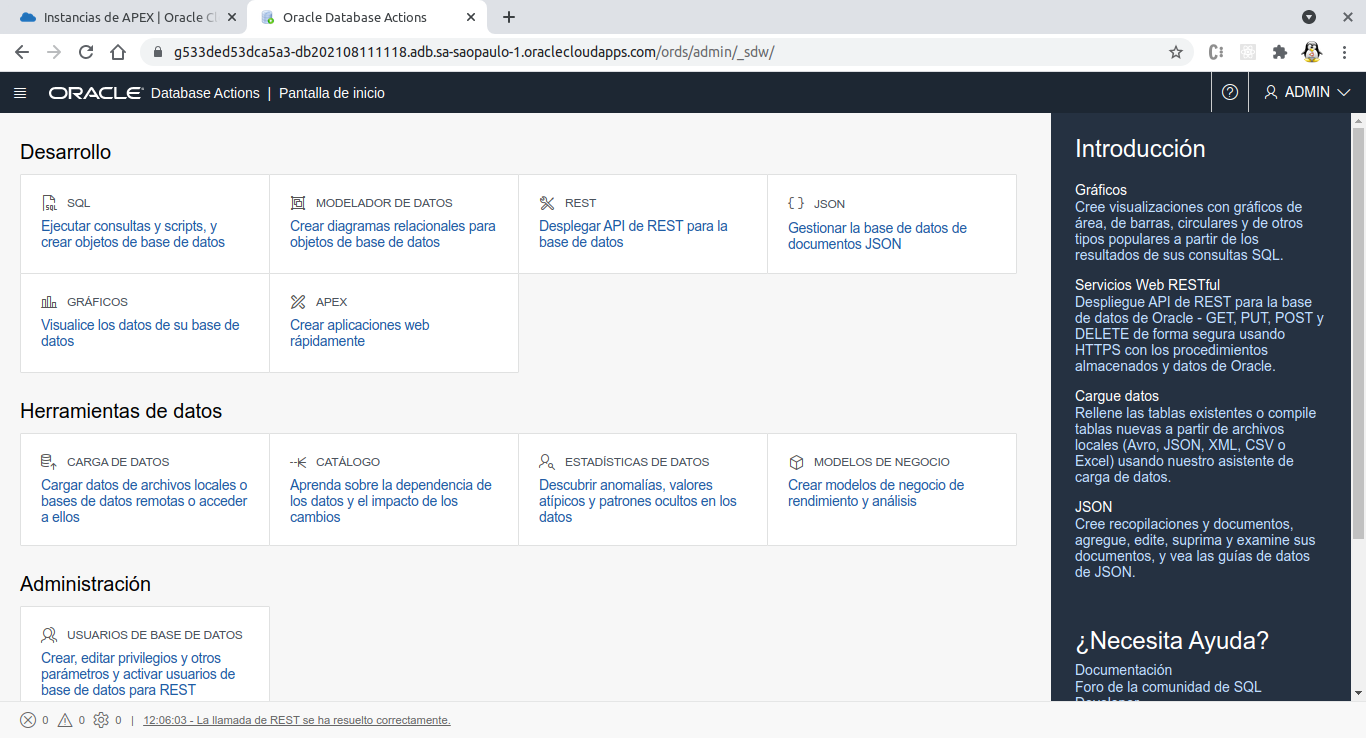
En caso que salgamos de esta pantalla, siempre podremos volver a ella haciendo cli en en el nombre de la instancia que se ha creado.



Una vez en el detalle de la instancia, haremos clic en el botón Iniciar acciones de base de datos, esto nos llevará a Oracle DataBase Actions, en el que escribiremos el usuario (ADMIN) y posteriormente la contraseña que hemos definido en la creación.



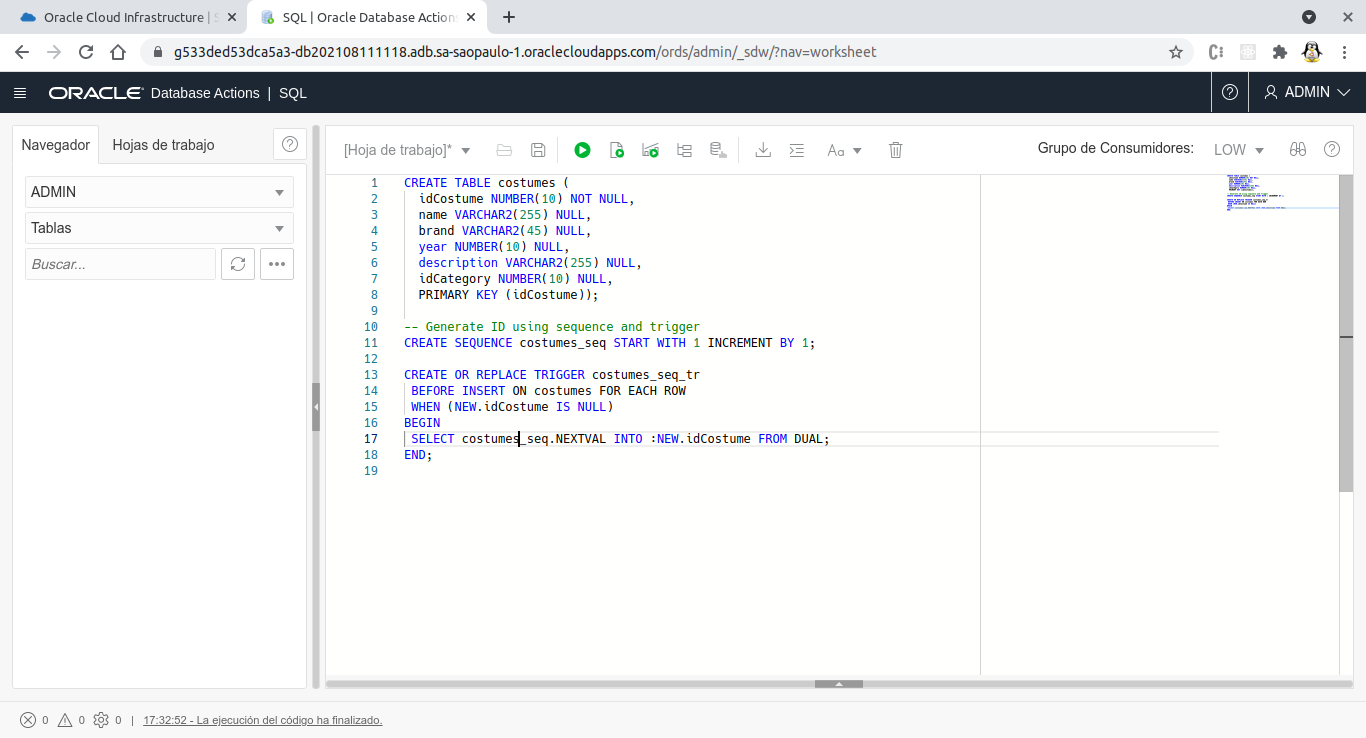
Al ingresar, nos encontraremos con un panel que nos muestra todas las acciones que podemos hacer, nosotros vamos a elegir *Ejecutar consultas y Scripts* en esta sección ejecutaremos un Script que tenemos prediseñado para este ejemplo (Se puede encontrar al final del documento).



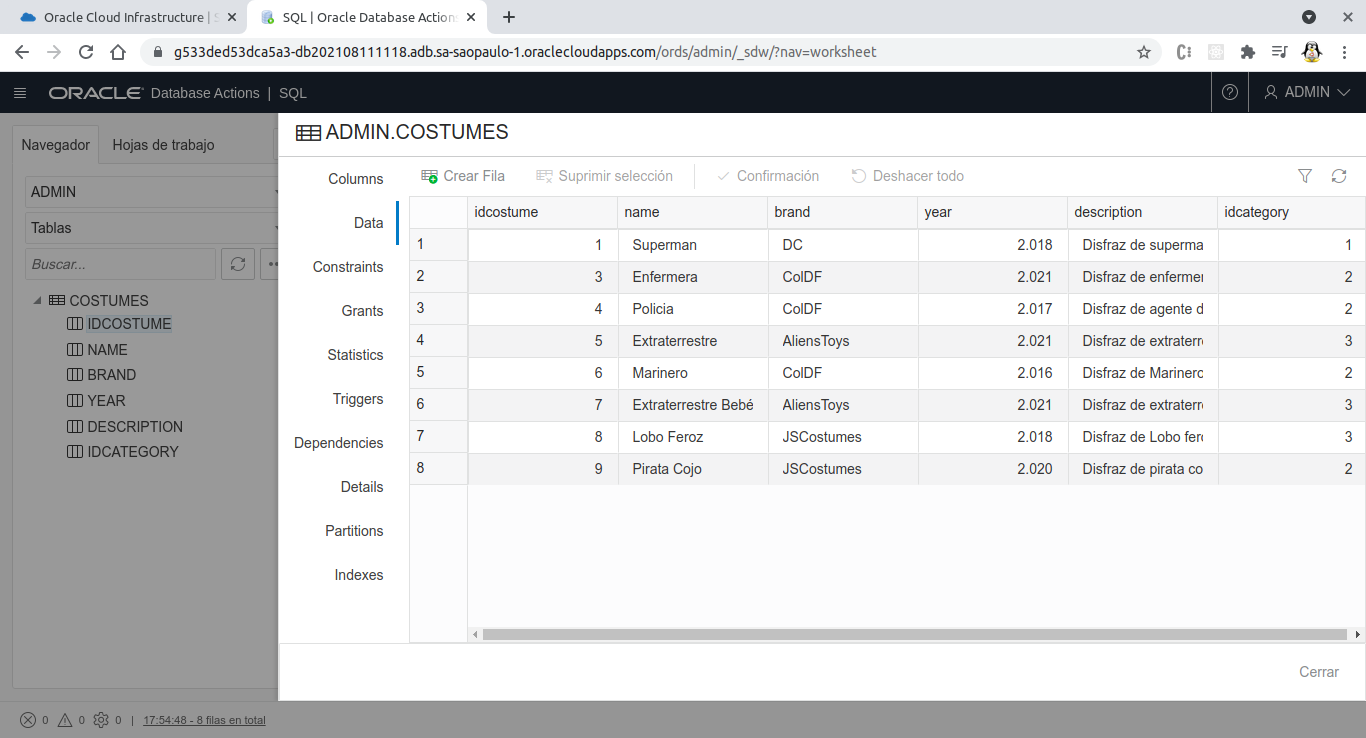
La ejecución del Script utilizado, será aquel que utilice la sintaxis correspondiente a ORACLE, debido a que la base de datos que utilizaremos estará en dicho motor.

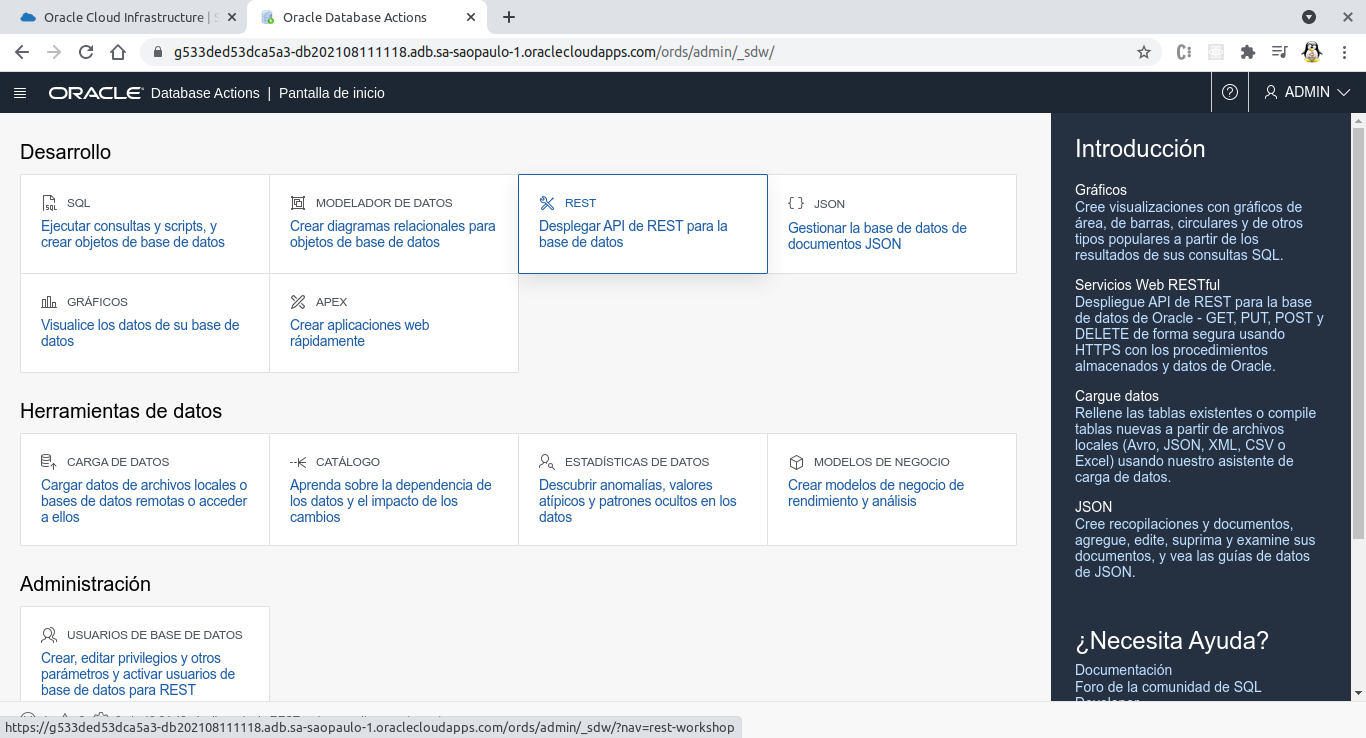
Bastará copiar y pegar el código que encontremos en el Script, y posteriormente ejecutarlo presionando el botón verde acompañado del ícono de un documento, que significa ejecutar Script.

Así se verá la ejecución de nuestro Script:

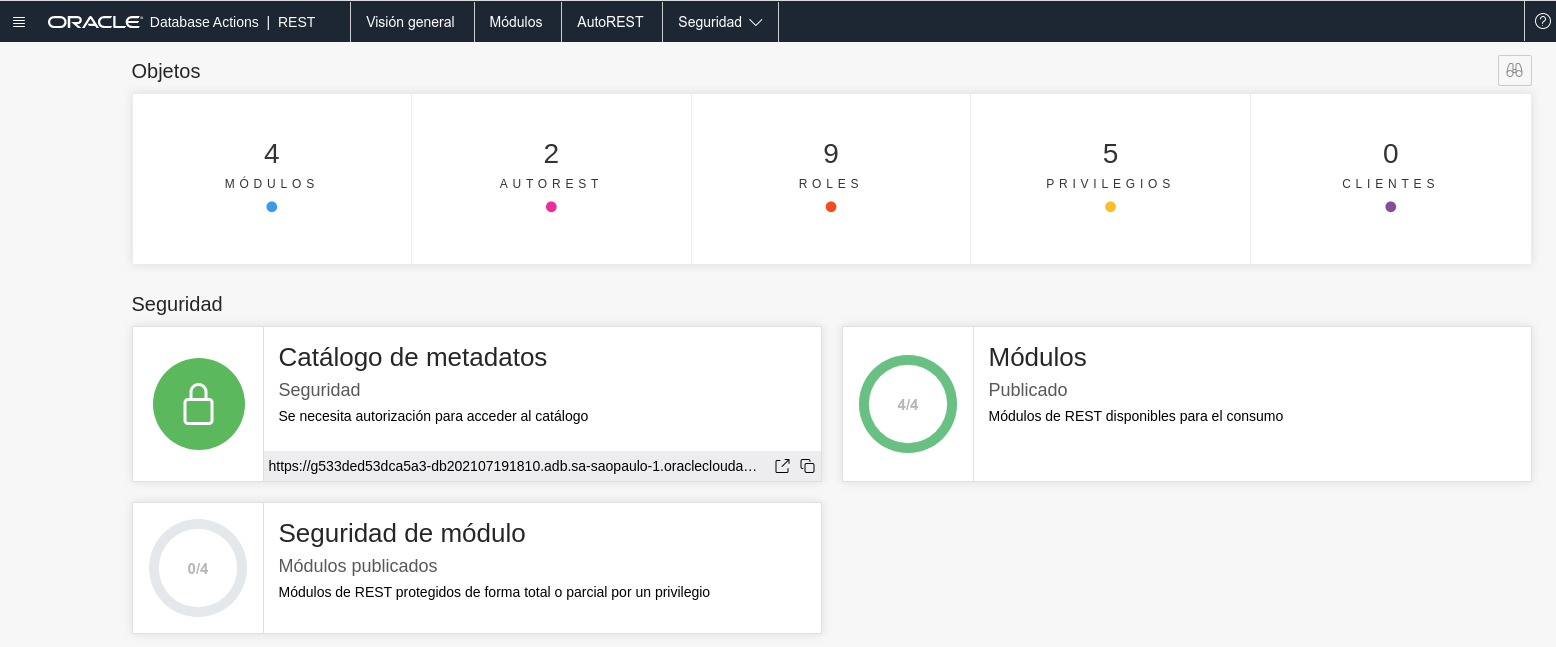


Es posible ejecutar el script por partes, primero la creación de la tabla y después las sentencias que insertan información. Una vez cargada la información, se puede validar haciendo clic en el botón refrescar que está justo bajo de la palabra Tablas en el panel lateral y posteriormente hacer clic derecho en el nombre de la tabla y en el menú que se despliega, elegir Abrir. La sección de Data debe verse así:

Al cerrar este panel que aparece, haremos clic en el encabezado de *Database Actions* para regresar al menú principal. Allí seleccionaremos la opción *REST: Desplegar API de REST para la base de datos.*

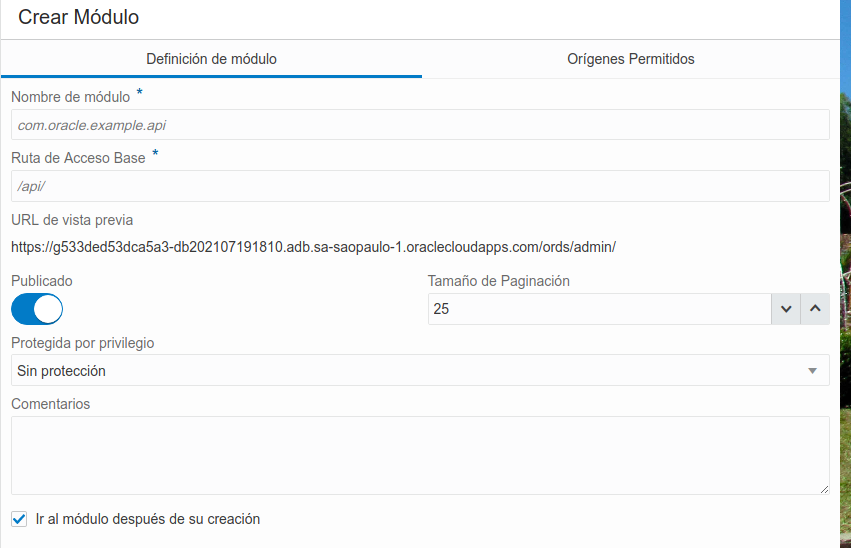


En la siguiente ventana, seleccionamos módulos, en la parte superior.

Una vez en la sección de módulos, haremos clic en el botón Crear módulo.

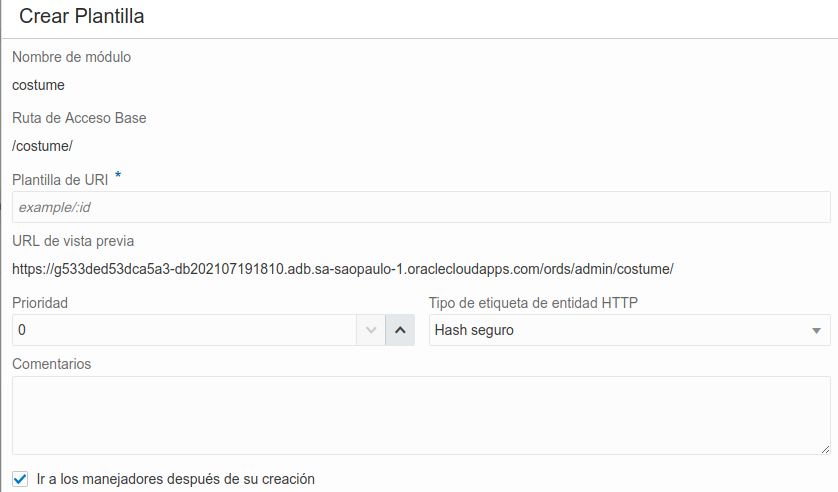
Un módulo, nos permite gestionar un recurso mediante la definición de sus manejadores que representan las acciones GET, POST, PUT y DELETE.

La creación del módulo nos mostrará la siguiente ventana en la que pondremos el nombre del módulo (Proponemos que coincida con la tabla o recurso que se espera se gestione. La ruta de acceso base, puede ser api o puede ser también el nombre del recurso.



Una vez en el módulo ya creado, presionamos el botón Crear Plantilla. La plantilla nos permitirá establecer una URL para una grupo de manejadores (GET, POST, PUT y DELETE). La plantilla tendrá como acceso la ruta base definida del módulo y posteriormente la URI que se defina. Es posible volver a escribir el mismo nombre de la entidad.

En los llamados REST, el parámetro correspondiente al ID se escribe como parte de la url seguido del carácter /. Para esto, crearemos dos plantillas: una con con la ruta para acceso general y otra para el acceso particular a entidades a partir del método GET.



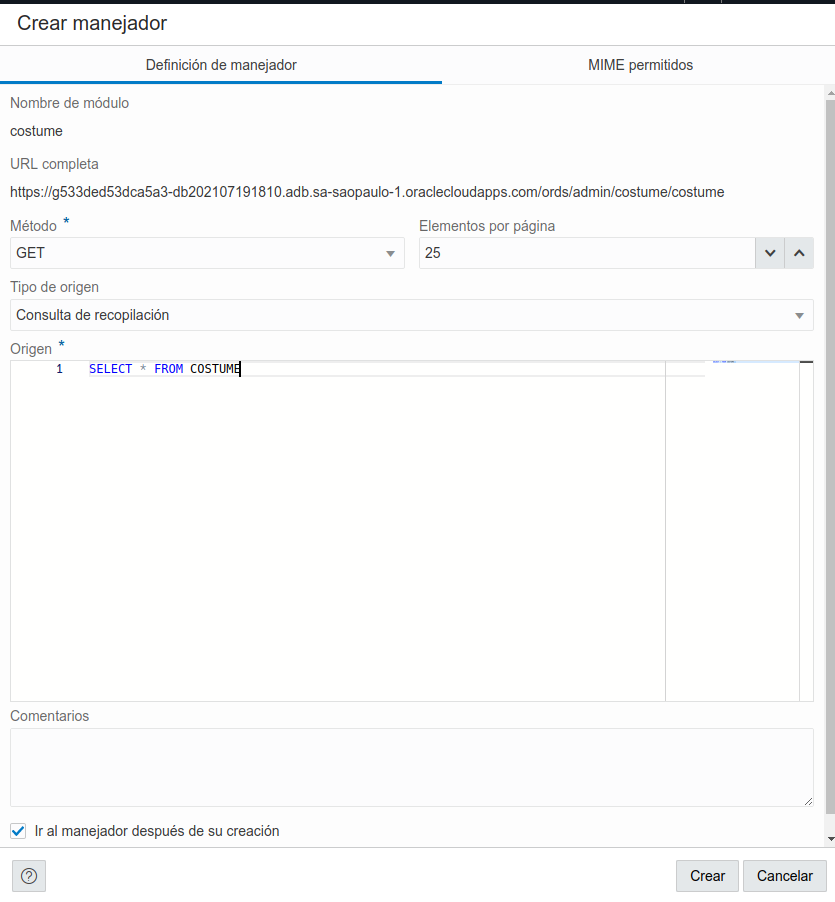
En este ejemplo pondremos costume como Plantilla de URL. Ya dentro de la plantilla crearemos los manejadores:

**GET**:

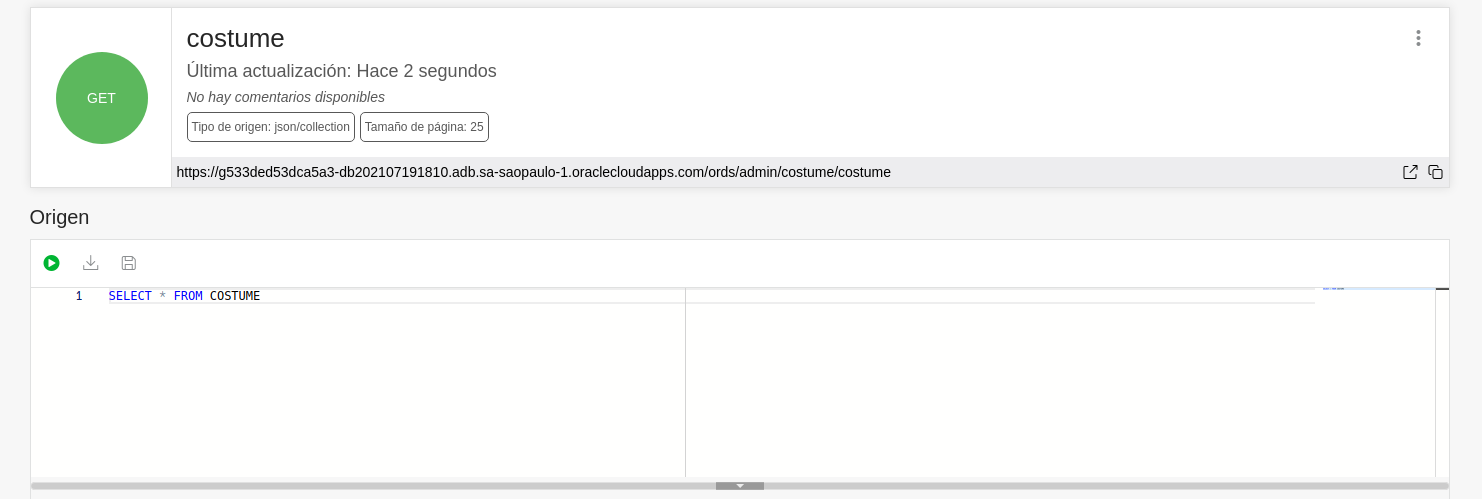
Clic en el botón crear manejador. En la ventana que aparece, elegiremos el Método GET, el tipo de origen es la manera como recolectará la información. Para este ejemplo elegiremos Consulta de recopilación y en la sección Origen, escribiremos el código SQL que permite traer todos los elementos de la tabla Costume. En nuestro caso escribiremos

SELECT \* FROM COSTUME

Es importante que el nombre sea tal cual lo hemos escrito al crear la tabla. Sin embargo, para evitar errores, la herramienta nos ofrece un texto predictivo en el que encontraremos con facilidad la tabla tal como la escribimos al crearla.



Presionamos el botón Crear. A continuación nos presentará el detalle y podremos utilizar el botón verde justo arriba de la sentencia SQL para probar si es correcta. Este botón ejecutará la sentencia y nos mostrará los resultados en la parte inferior de la pantalla. También nos mostrará si estamos equivocados en su sintaxis. Es importante realizar esta prueba pues es de allí que nuestro servicio utilizará la información para presentarla.



Otro elemento importante es la URL completa que tendremos ahí junto con la opción de copiar o de vistiar directamente. Esta URL será la misma para todos los métodos de la plantilla y solamente con el método GET tendríamos una prueba directa a través del navegado, debido a que toda petición que se haga a través de la barra de direcciones de nuestro navegador, es en sí una petición GET.

Una vez probado y ejecutado el código, podemos entrar al servicio. Si hemos de hacer algún cambio en la sentencia, no debemos olvidar presionar el botón de guardado para que una vez salgamos de ahí, los cambios se mantengan.

**POST:**

El método POST será el que utilizaremos para la creación de nuevos registros, es decir que a través del método post, guardaremos información en la tabla.

Para este método es particularmente útil que recordemos la manera como está creada la tabla sobre la que estamos trabajando, pues la sentencia SQL que utilizaremos hará uso de esos valores. Para nuestro ejemplo tendremos los campos:

ID, BRAND, MODEL, CATEGORY\_ID y NAME.

Al hacer clic en el botón crear manejador, ya no encontraremos la opción de método GET, puesto que ya está creado. Restan POST, PUT y DELETE. Elegiremos POST. El tipo de Origen por defecto (e inmutable) es PL/SQL. La sentencia que escribiremos será la siguiente:

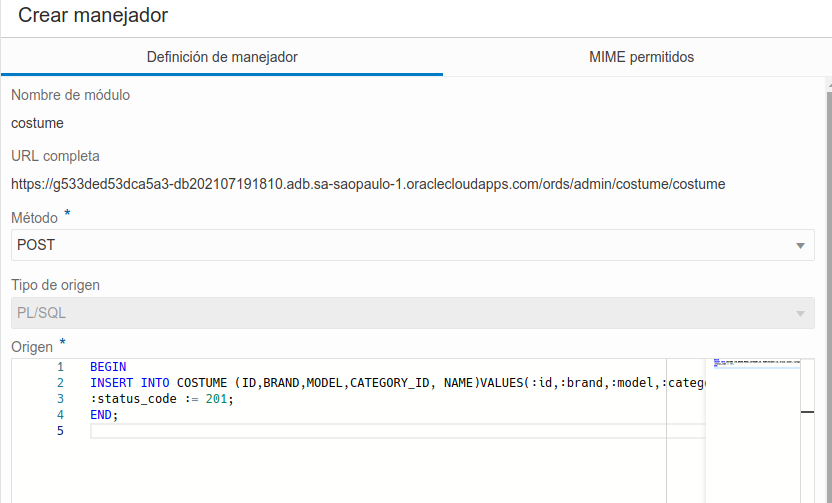
BEGIN

INSERT INTO COSTUME (ID,BRAND,MODEL,CATEGORY\_ID, NAME)VALUES(:id,:brand,:model,:category\_id,:name);

:status\_code := 201;

END;

En la pestaña MIME permitidos, corresponde al tipo de información que la petición está dispuesta a recibir, seleccionaremos "application/json".



Una vez creado el manejador, nos remite a la vista de detalle. Para probar en este caso, Oracle nos preguntará a través de ventanas emergentes la información que hemos de suministrar para la creación del registro.

En la sentencia anterior, podemos ver que realiza un INSERT en la tabla, señalando campo a campo el orden de los valores. Los parámetros son establecidos como variables a través de la anteposición del carácter dos puntos (:), así, la variable :name almacenará el nombre que tendrá el futuro registro. Se resalta que la correspondencia entre parámetro y columna se hace por la coincidencia de orden entre las columnas y los valores.

**PUT**

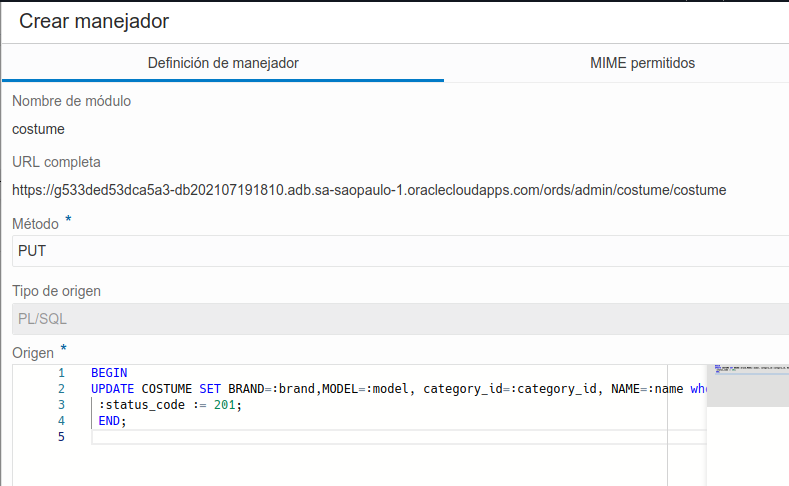
El método PUT será utilizado para editar la información existente en la báse de datos. Esta edición se hará mediante la búsqueda del id y se reemplazará cada campo con su correspondiente . Al igual que la petición POST, en esta petición el cuerpo del mensaje debe ir en formato JSON. La sentencia de este método corresponde a:

BEGIN

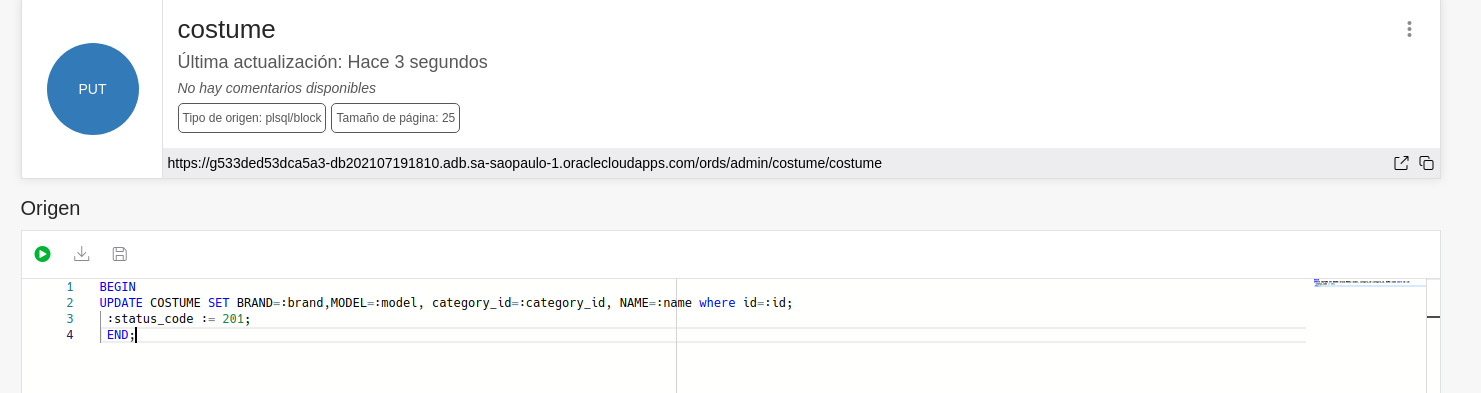
UPDATE COSTUME SET BRAND=:brand,MODEL=:model, category\_id=:category\_id, NAME=:name where id=:id;

:status\_code := 201;

END;



La sentencia en este caso, sí crea una correspondencia de cada parámetro con el correspondiente atributo. Para ello, igual que en el método anterior es muy importante que se envíe la información correcta y en cuanto a los nombres de los campos en el JSON correspondiente al cuerpo de la petición.



Igual que en el anterior método, una vez probado, es posible invocar el método get con el fin de validar la informaciónguardad.

**DELETE**

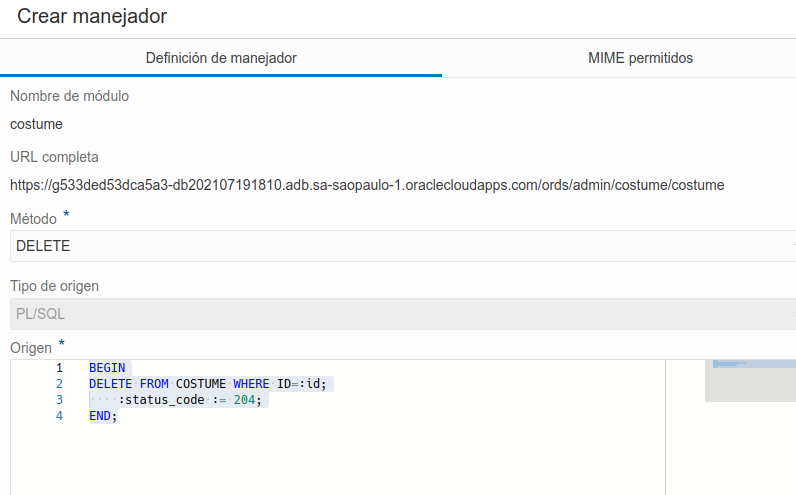
El útlimo método de esta plantilla será el método delete. Este método recibirá solamente el id en el cuerpo de la petición en un objeto json. La creación se hará igual que en los otros métodos, pero en este caso .

BEGIN

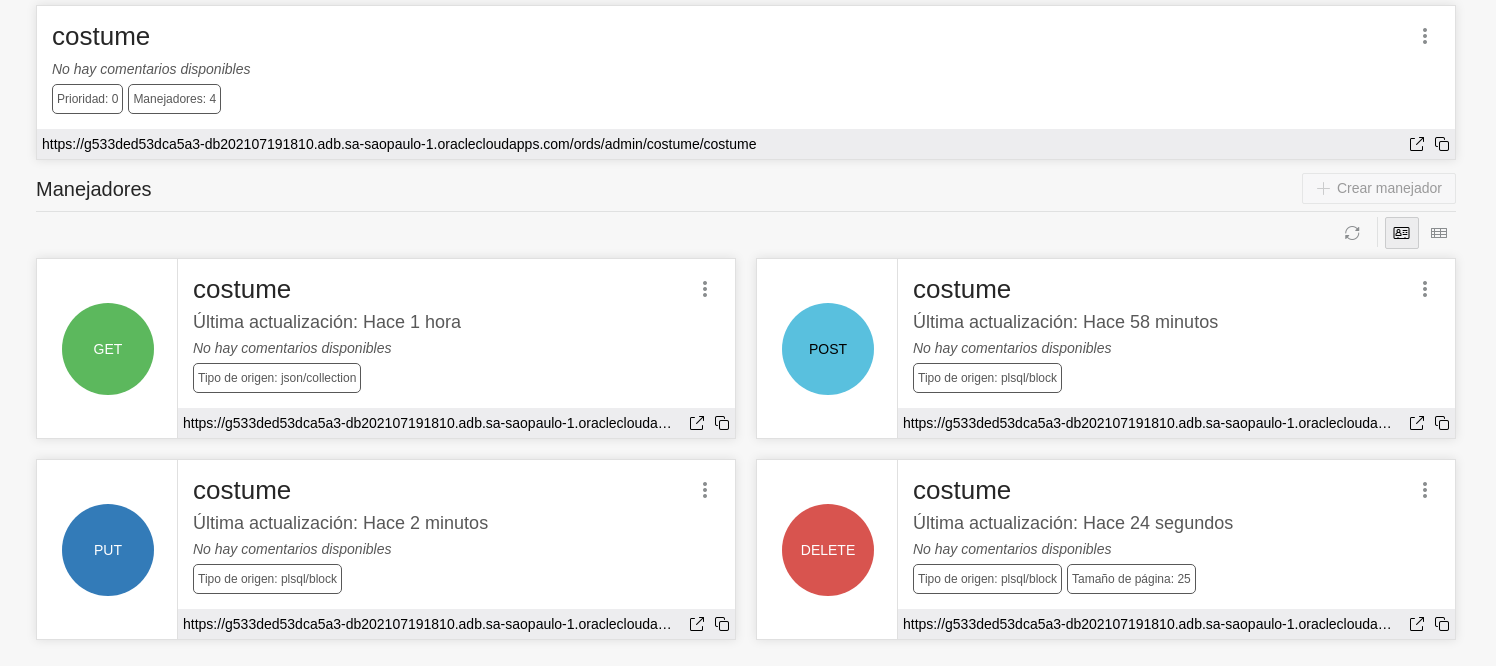
DELETE FROM COSTUME WHERE ID=:id;

:status\_code := 204;

END;

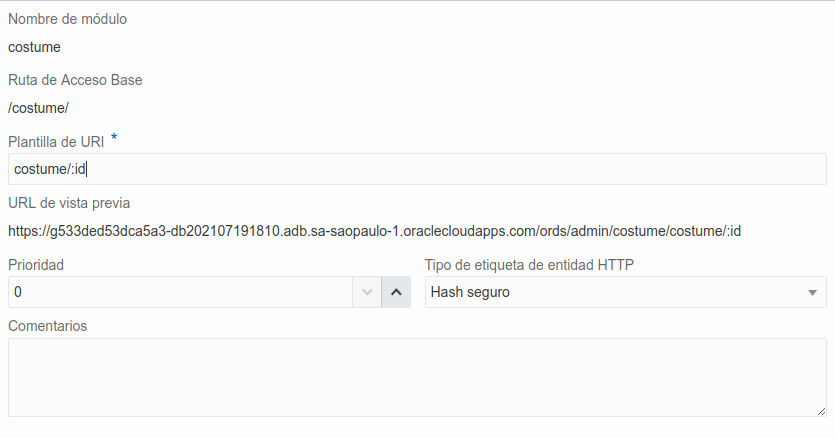


Ahora nuestros manejadores se encuentran completos, y podemos ver que los cuatro métodos están creados, todos trabajarán bajo la misma dirección URL y podemos empezar a hacer pruebas a través de postman o cualquier otro cliente de peticiones REST.



Ahora, volveremos a la sección de plantillas y crearemos una plantilla que reciba como parámetro el id. Esta plantilla tendrá un manejador GET que se encargará de buscar según id.

Para lograr que el parámetro esté en la URL, escribiremos :id en el final de la Plantilla de URI

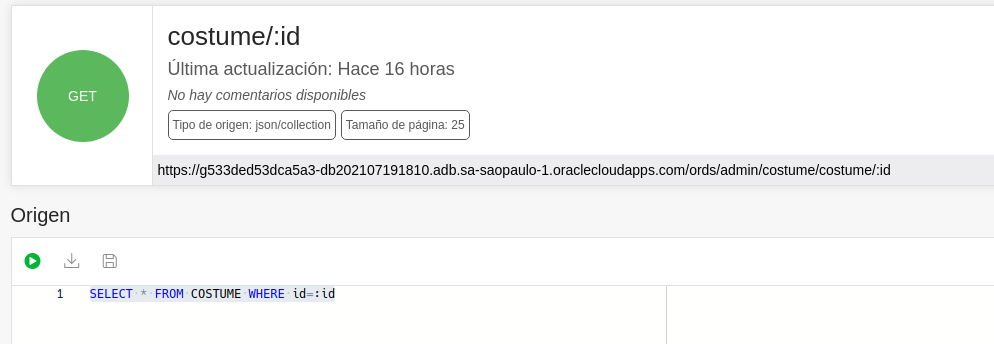


Así quedará costume/:id. Todo aquello que en la url vaya después de costume/ será interpretado como id.

Dentro de la plantilla crearemos un manejador GET que ejecutará la sentencia:

SELECT \* FROM COSTUME WHERE id=:id

Se verá así:



Ahora, ya se podrá validar desde el navegador esta petición GET con cada elemento de la tabla.

Para terminar, adjuntamos el SCRIPT de creación de la tabla y un conjunto de datos iniciales para lograr una correcta experimentación.

--ORACLE

CREATE TABLE COSTUME (

ID NUMBER(10) NOT NULL,

NAME VARCHAR2(255) NULL,

BRAND VARCHAR2(45) NULL,

MODEL NUMBER(10) NULL,

CATEGORY\_ID NUMBER(10) NULL,

PRIMARY KEY (ID));

--Inserts

INSERT INTO COSTUME (id,name, brand, model, category\_id) VALUES (1,'Disfraz de superman para adulto', 'DC', '2018', '1');

INSERT INTO COSTUME(id, name, brand, model, category\_id) VALUES (2,'Disfraz de Batman para niño', 'DC', '2019', '1');

INSERT INTO COSTUME (id, name, brand, model, category\_id) VALUES (3,'Enfermera', 'ColDF', '2021', '2');

INSERT INTO COSTUME (id,name, brand, model, category\_id) VALUES (4,'Disfraz de agente de policía para niño', 'ColDF', '2017', '2');

INSERT INTO COSTUME (id,name, brand, model, category\_id) VALUES (5,'Extraterrestre', 'AliensToys', '2021', '3');

INSERT INTO COSTUME (id,name, brand, model, category\_id) VALUES (6,'Marinero', 'ColDF', '2016','2');

INSERT INTO COSTUME (id,name, brand, model, category\_id) VALUES (7,'Extraterrestre Bebé', 'AliensToys', '2021', '3');

INSERT INTO COSTUME (id,name, brand, model, category\_id) VALUES (8,'Lobo Feroz', 'JSCostumes', '2018', '3');

INSERT INTO COSTUME (id,name, brand, model, category\_id) VALUES (9,'Disfraz de pirata cojo con pata de palo y cara de malo.', 'JSCostumes', '2020', '2');