

Converged Database: Modern Development Workshop

Day 1

Índice

REQUERIMIENTOS INICIALES Y ACCESO AL ENTORNO	3
SESIÓN 1.....	6
DESCRIPCIÓN DEL ENTORNO DEL WORKSHOP	6
UNPLUG Y PLUG DE PDB1 Y CREACIÓN DE PDB2.	8
UNPLUG DE PDB1 Y PLUG DENTRO DEL APPLICATION CONTAINER.....	9
CREACIÓN DE PDB2 DENTRO DEL APPLICATION CONTAINER.....	12
TRANSFORMACIÓN DE DATOS RELACIONALES DE APP1 EN JSON PARA APP2.	14
DESPLIEGUE DE APP2 (NODEJS + SODA) USANDO PDB2	20
CONSULTA COMBINADA DE LA INFORMACIÓN DE APP1 Y APP2.....	26
EXPOSICIÓN DE DATOS EN API REST CON ORDS.	29



Requerimientos iniciales y acceso al entorno

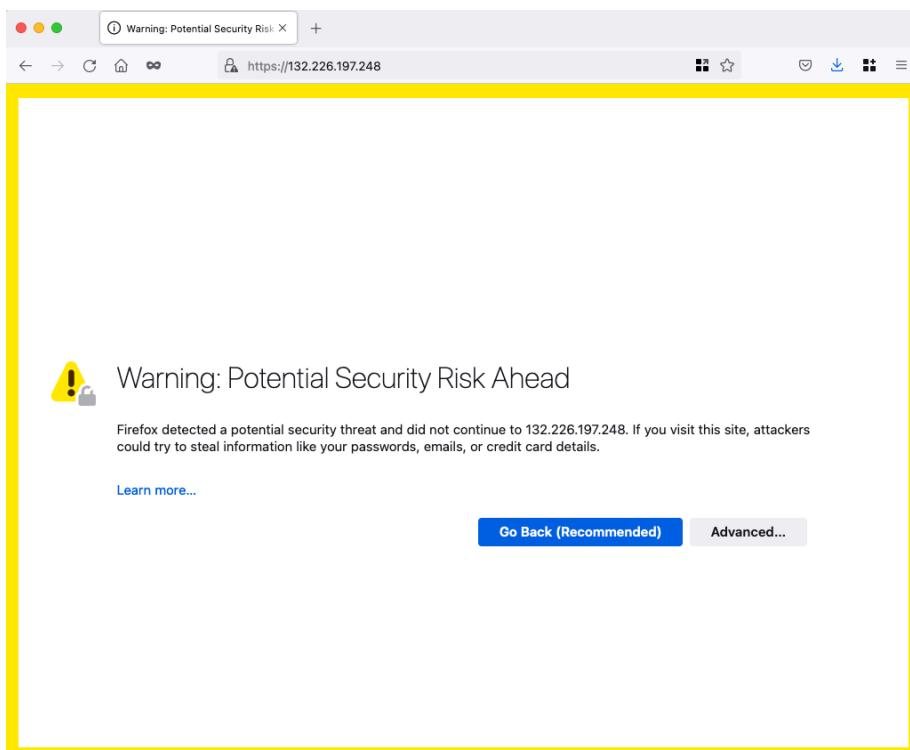
El taller que va a realizar está diseñado para realizarse usando únicamente un navegador web desde el que podrá tener acceso a un escritorio remoto.

Para la realización de este taller necesita:

- Clave de acceso al entorno que le proporcionará el instructor. Debe acceder con el usuario **oracle**.
- Navegador web (Firefox o Chrome recomendados)
- Dirección IP de la maquina.

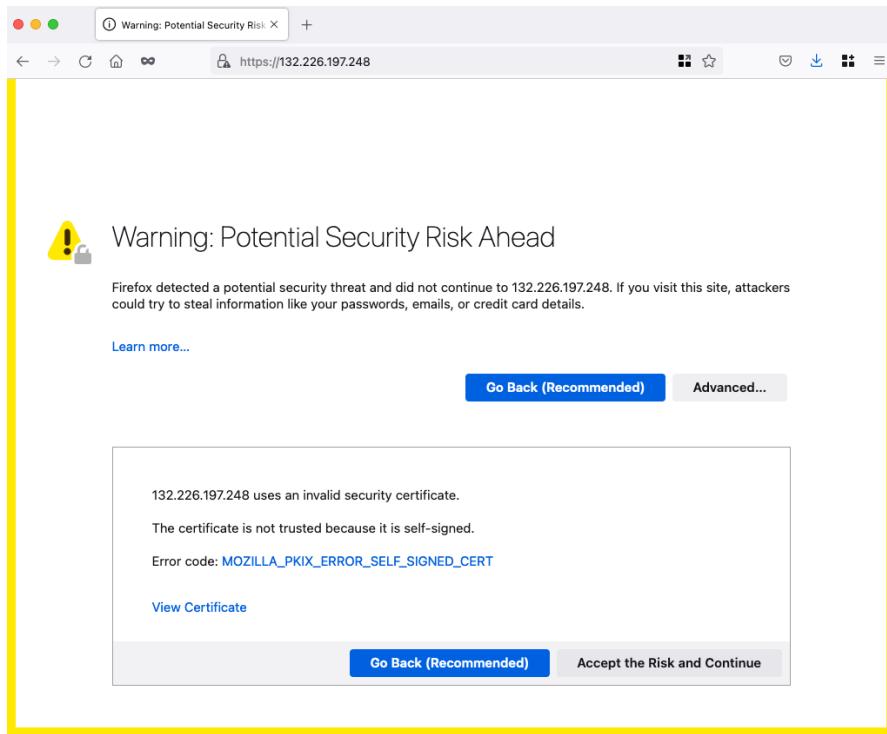
La conexión se realiza por https, pero el certificado es autogenerado por lo que recibirá una alerta de seguridad en el primer intento de conexión informando de este hecho. Debe aceptar el mensaje para poder continuar con la conexión.

Haga click en el botón ‘Advanced...’:

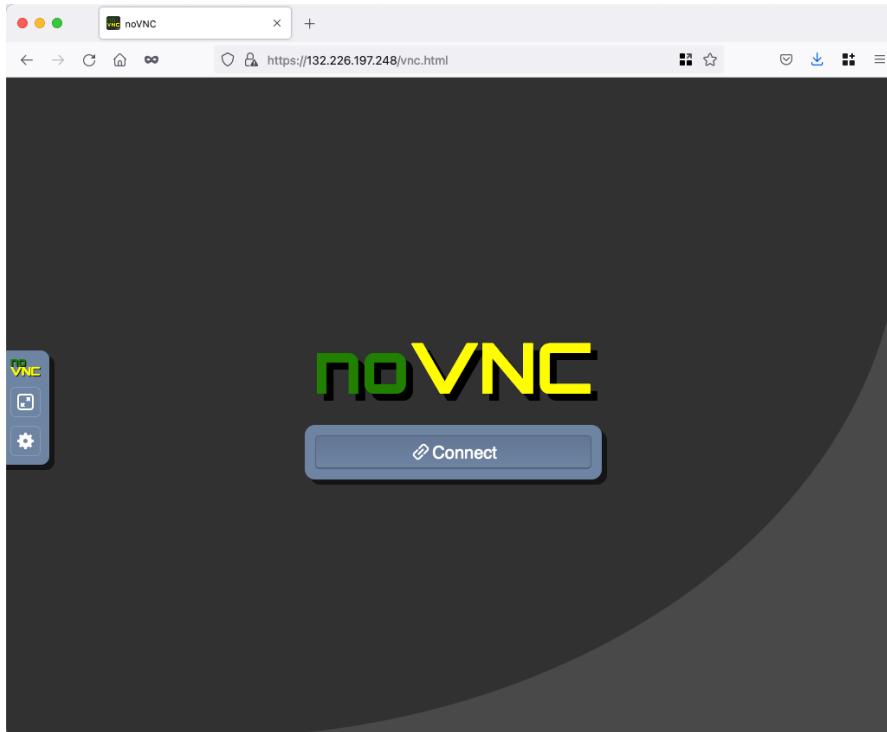


Verá el detalle del potencial problema de seguridad, que es debido al uso de un certificado autogenerado.



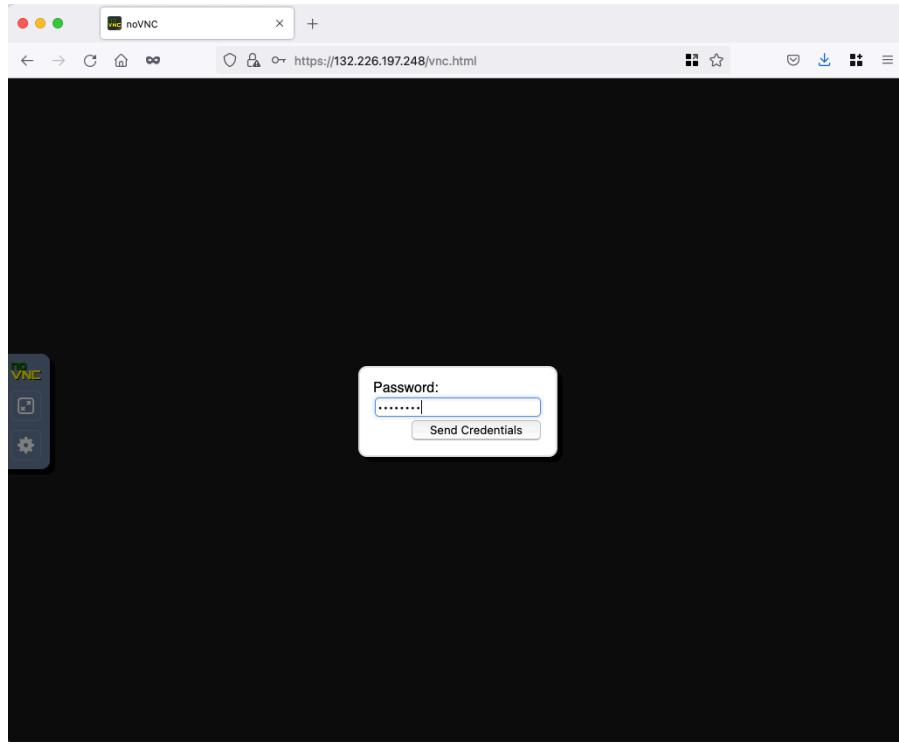


Pulse el botón ‘Accept the Risk and Continue’ tras lo cual deberá ver la pantalla de acceso a noVNC, que proporciona el acceso al escritorio remoto.

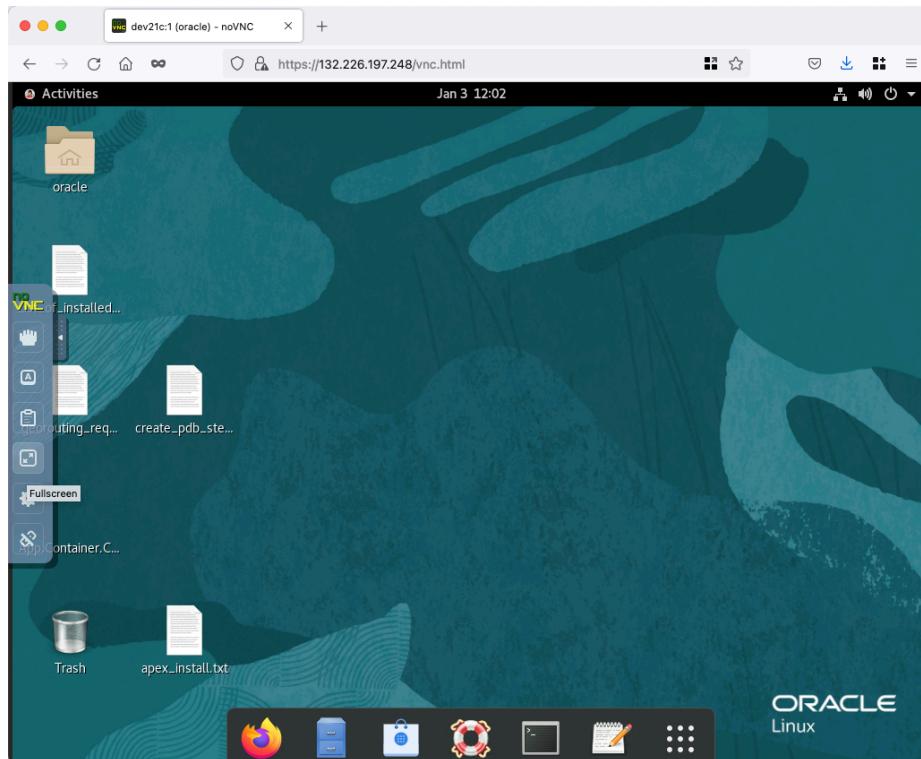


Haga click en el botón ‘Connect’ e introduzca la clave proporcionada por el instructor.





Tras estos pasos verá en su pantalla el escritorio remoto de la máquina del workshop. En el menú del lateral izquierdo puede hacer que el escritorio ocupe toda la pantalla (opción Fullscreen).



Sesión 1

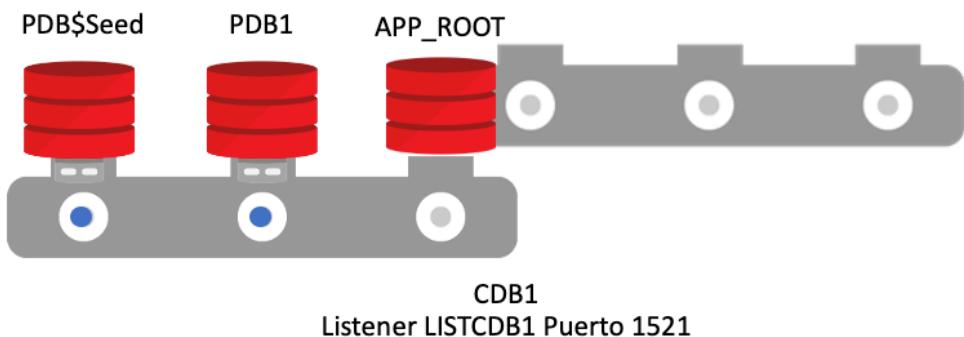
Descripción del entorno del workshop

La máquina a la que usted accede tiene todos los elementos necesarios para las actividades preinstalados y configurados.

Inicialmente dispone de un container o instancia de base de datos Oracle 21c con nombre **ORCLCDB** con las siguientes características:

```
ORCLCDB
=====
PDBs existentes:
  PDB$SEED
  PDB1
  APP_ROOT

Listener LISTENER puerto 1521
Oracle Home /opt/oracle/product/21c/dbhome_1
```



La PDB1 contiene un esquema SOE usado por una aplicación de gestión de pedidos e inventario ya existente (basada en Swingbench).

Puede inspeccionar los contenidos del esquema desde SQLcl abriendo un Terminal:

```
[oracle@dev21c ~]$ sql soe/soe@localhost:1521/pdb1

SQLcl: Release 21.2 Production on Mon Jan 03 12:30:24 2022
Copyright (c) 1982, 2022, Oracle. All rights reserved.

Connected to:
Oracle Database 21c Enterprise Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

SQL> select table_name from user_tables;
```



```
TABLE_NAME
```

```
PRODUCT_INFORMATION
```

```
LOGON
```

```
ORDERS
```

```
CARD_DETAILS
```

```
INVENTORIES
```

```
PRODUCT_DESCRIPTIONS
```

```
CUSTOMERS
```

```
ORDER_ITEMS
```

```
ORDERENTRY_METADATA
```

```
ADDRESSES
```

```
WAREHOUSES
```

```
PURCHASE_ORDERS
```

```
12 rows selected.
```

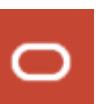
```
SQL> exit
```

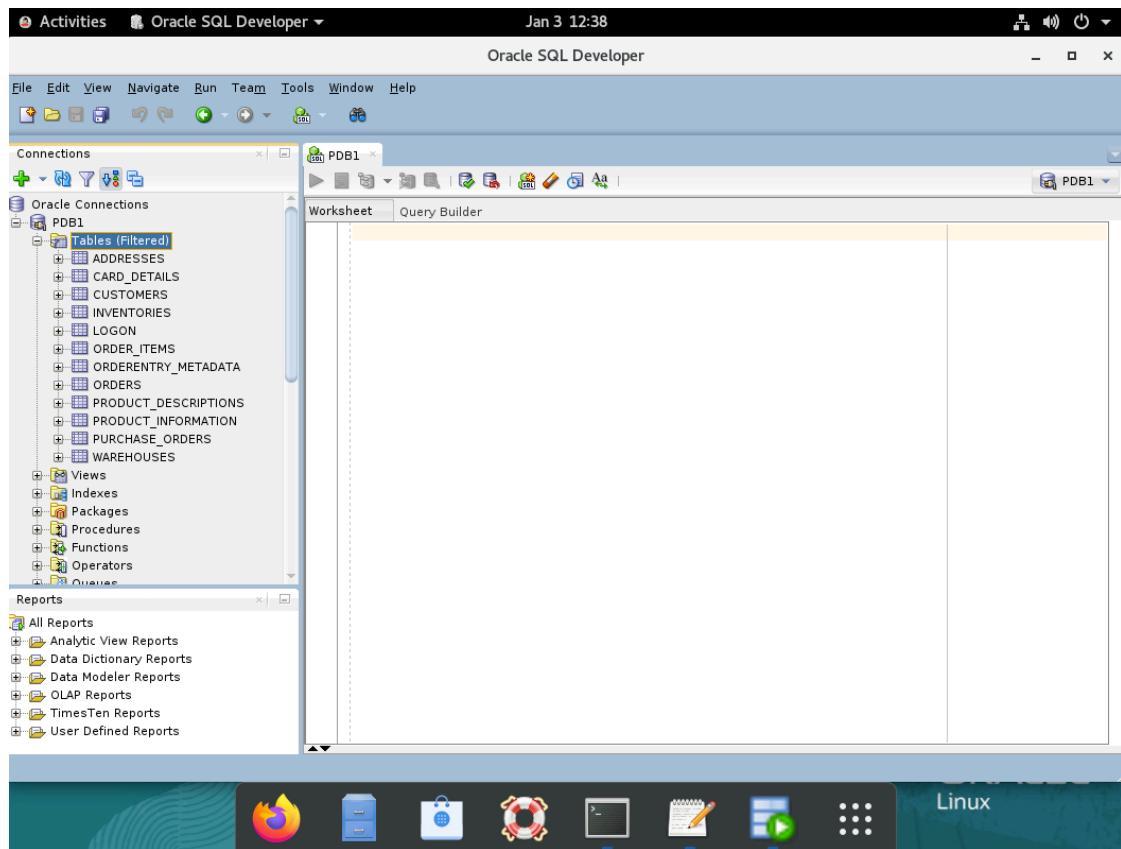
```
Disconnected from Oracle Database 21c Enterprise Edition Release 21.0.0.0.0 - Production  
Version 21.3.0.0.0
```

También puede usar SQL*Developer que se encuentra preinstalado y configurado. Haga click en el dock de la parte inferior de su escritorio remoto (1) y seleccione SQL Developer (2).



Una vez arrancado SQL*Developer use la conexión PDB1 para inspeccionar el esquema SOE.

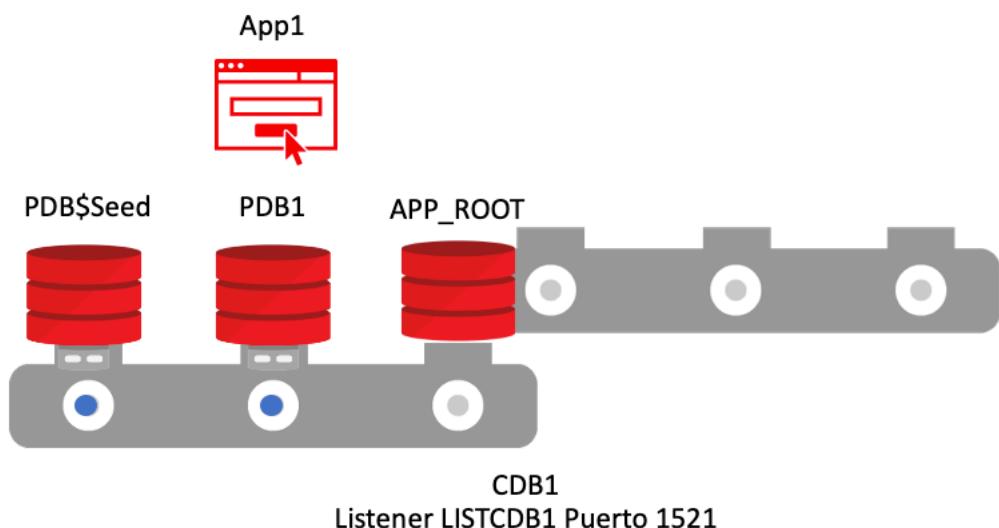




Cierre la aplicación SQL*Developer cuando haya terminado.

Unplug y plug de PDB1 y creación de PDB2.

En esta actividad se va a usar un tipo especial de PDB denominado **Application Container** que se usará durante el resto del taller.



El Application Container se llama ‘APP_ROOT’ y gracias a él se dispondrá de un nuevo nivel en la jerarquía donde organizar las PDB que estén asociadas a una aplicación.

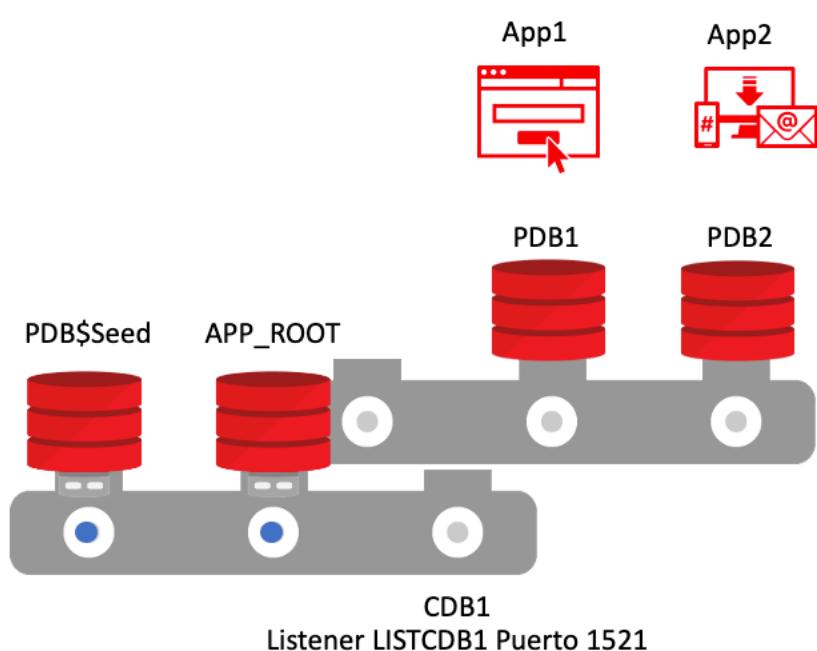
El objetivo de APP_ROOT es alojar varias PDBs, concretamente:

- PDB1 que daba servicio a la aplicación de pedidos e inventario ya existente App1
- PDB2 una nueva PDB para la nueva aplicación de pedidos basada en JSON App2

PDB1 contiene el esquema relacional de la aplicación heredada y contiene datos de actividad de negocio, por ello va a ser movida dentro del Application Container sin ninguna pérdida de datos. Una vez movida la aplicación heredada la seguirá utilizando de forma transparente.

PDB2 contendrá un nuevo esquema para una nueva aplicación de pedidos que empleará JSON. Esta nueva PDB se creará directamente dentro del Application Container.

El resultado final será como se muestra a continuación:



Unplug de PDB1 y plug dentro del Application Container

La existente PDB1 debe ser alojada dentro del Application Container APP_ROOT.

Esto se consigue en varios pasos:

- Unplug de PDB1 conservando sus datafiles
- Plug de PDB1 dentro del Application Container APP_ROOT

A continuación puede ver los comandos (de nuevo en rojo) de unplug de PDB1.

```
$ sqlplus / as sysdba
SQL> alter pluggable database PDB1 close;
```



```

Pluggable database PDB1 altered.

SQL> ALTER PLUGGABLE DATABASE PDB1 UNPLUG INTO '/home/oracle/pdb1.xml';

Pluggable database PDB1 altered.

SQL> DROP PLUGGABLE DATABASE PDB1 KEEP DATAFILES;

Pluggable database PDB1 dropped.

SQL> show pdbs;

  CON_ID CON_NAME           OPEN MODE  RESTRICTED
----- -----
    2 PDB$SEED             READ ONLY   NO
    4 APP_ROOT              READ WRITE  NO

```

Ahora se realiza el plug de la PDB dentro del Application Container APP_ROOT.

```

SQL> alter session set container=app_root;

Session altered.

SQL> CREATE PLUGGABLE DATABASE PDB1 USING '/home/oracle/pdb1.xml';

Pluggable database PDB1 created.

SQL> alter pluggable database pdb1 open;

Warning: PDB altered with errors.

Pluggable database PDB1 altered.
SQL> alter pluggable database pdb1 open;
ORA-24344: success with compilation error
24344. 00000 - "success with compilation error"
*Cause: A sql/plsql compilation error occurred.
*Action: Return OCI_SUCCESS_WITH_INFO along with the error code

Pluggable database PDB1 altered.

SQL> show pdbs;

  CON_ID CON_NAME           OPEN MODE  RESTRICTED
----- -----
    4 APP_ROOT              READ WRITE NO
    5 PDB1                  READ WRITE YES

SQL> select message,time from pdb_plug_in_violations;

MESSAGE
-----
TIME
-----
Database option OML4PY mismatch: PDB installed version NULL. CDB installed versi
on 21.0.0.0.0.
21-MAR-22 10.56.09.463152 AM

```



Application ORDS in Application Root does not exist in Application PDB.
21-MAR-22 11.09.43.586546 AM

Application APEX in Application Root does not exist in Application PDB.
21-MAR-22 11.09.43.587025 AM

MESSAGE

TIME

Non-Application PDB plugged in as an Application PDB, requires pdb_to_appdb.sql
be run.

21-MAR-22 11.09.43.596803 AM

MESSAGE

TIME

Database option OML4PY mismatch: PDB installed version NULL. CDB installed version
21.0.0.0.0. 03-JAN-22 01.40.00.155040000 PM

Database option OML4PY mismatch: PDB installed version NULL. CDB installed version
21.0.0.0.0. 03-JAN-22 01.50.26.032530000 PM

Non-Application PDB plugged in as an Application PDB, requires pdb_to_appdb.sql be run.
03-JAN-22 01.50.26.105950000 PM

Se observa un error porque es necesario ejecutar un script (pdb_to_appdb.sql) que convierte una PDB normal en una Application PDB.

```
SQL> conn sys/Oracle_4U@localhost:1521/pdb1 as sysdba
Connected.
SQL> @?/rdbms/admin/pdb_to_appdb.sql
[...]
SQL> conn sys/Oracle_4U@localhost:1521/app_root as sysdba
Connected.
SQL> show pdbs

  CON_ID CON_NAME          OPEN MODE  RESTRICTED
----- -----
        4 APP_ROOT          READ WRITE NO
        5 PDB1              READ WRITE YES

SQL> alter pluggable database pdb1 close;

Pluggable database PDB1 altered.

SQL> alter pluggable database pdb1 open;

Pluggable database PDB1 altered.

SQL> alter pluggable database pdb1 save state;

Pluggable database PDB1 altered.
```



```
SQL> show pdbs
```

CON_ID	CON_NAME	OPEN MODE	RESTRICTED
4	APP_ROOT	READ WRITE	NO
5	PDB1	READ WRITE	NO

Este cambio es transparente para la aplicación App1, no es necesario cambio alguno en la cadena de conexión porque se ha mantenido el nombre de la PDB. Se puede probar que la conexión existente en SQL*Developer sigue funcionando sin necesidad de cambio alguno.

Creación de PDB2 dentro del Application Container

En último lugar, dentro del Application Container APP_ROOT se crea la nueva PDB PDB2 que dará servicio a la nueva aplicación. PDB2 está vacía, por lo que se va a crear un tablespace y un usuario para alojar los datos de la nueva aplicación.

```
SQL> CREATE PLUGGABLE DATABASE PDB2 ADMIN USER pdbadmin IDENTIFIED BY "Oracle_4U";
```

Pluggable database PDB2 created.

```
SQL> alter pluggable database PDB2 open;
```

Pluggable database PDB2 altered.

```
SQL> alter pluggable database pdb2 save state;
```

Pluggable database PDB1 altered.

```
SQL> show pdbs;
```

CON_ID	CON_NAME	OPEN MODE	RESTRICTED
3	PDB2	READ WRITE	NO
4	APP_ROOT	READ WRITE	NO
5	PDB1	READ WRITE	NO

```
SQL> alter session set container=PDB2;
```

Session altered.

```
SQL> create tablespace TBS_SOE datafile size 400M;
```

Tablespace TBS_SOE created.

```
SQL> create user SOE identified by soe default tablespace TBS_SOE temporary tablespace TEMP;
```

User SOE created.

```
SQL> alter user SOE quota unlimited on TBS_SOE;
```

User SOE altered.

```
SQL> grant connect, resource to SOE;
```



```
Grant succeeded.  
SQL> grant create view to SOE;  
Grant succeeded.  
SQL> grant create database link to SOE;  
Grant succeeded.  
SQL> grant soda_app to SOE;  
Grant succeeded.  
SQL> exit
```

El grant de creación de database link se usará en la siguiente sección para migrar los datos originales de los pedidos. Tenga en cuenta que los database links entre PDBs de un mismo container se ejecutan en memoria proporcionando una velocidad y rendimiento notables.

El usuario SOE de PDB2 dispone también de un grant soda_app que lo habilita para la funcionalidad **SODA** (Simple Oracle Document Access) que son un conjunto de APIs que permiten la creación y almacenamiento de colecciones de documentos JSON así como su recuperación y consulta sin necesidad de usar SQL o preocuparse de cómo se graban en la base de datos.

Existen implementaciones de SODA para JAVA, Python, C, PL/SQL, REST y Node.js siendo este último el que se va a mostrar en las siguientes actividades de este taller.

Más información en <https://docs.oracle.com/en/database/oracle/simple-oracle-document-access/>

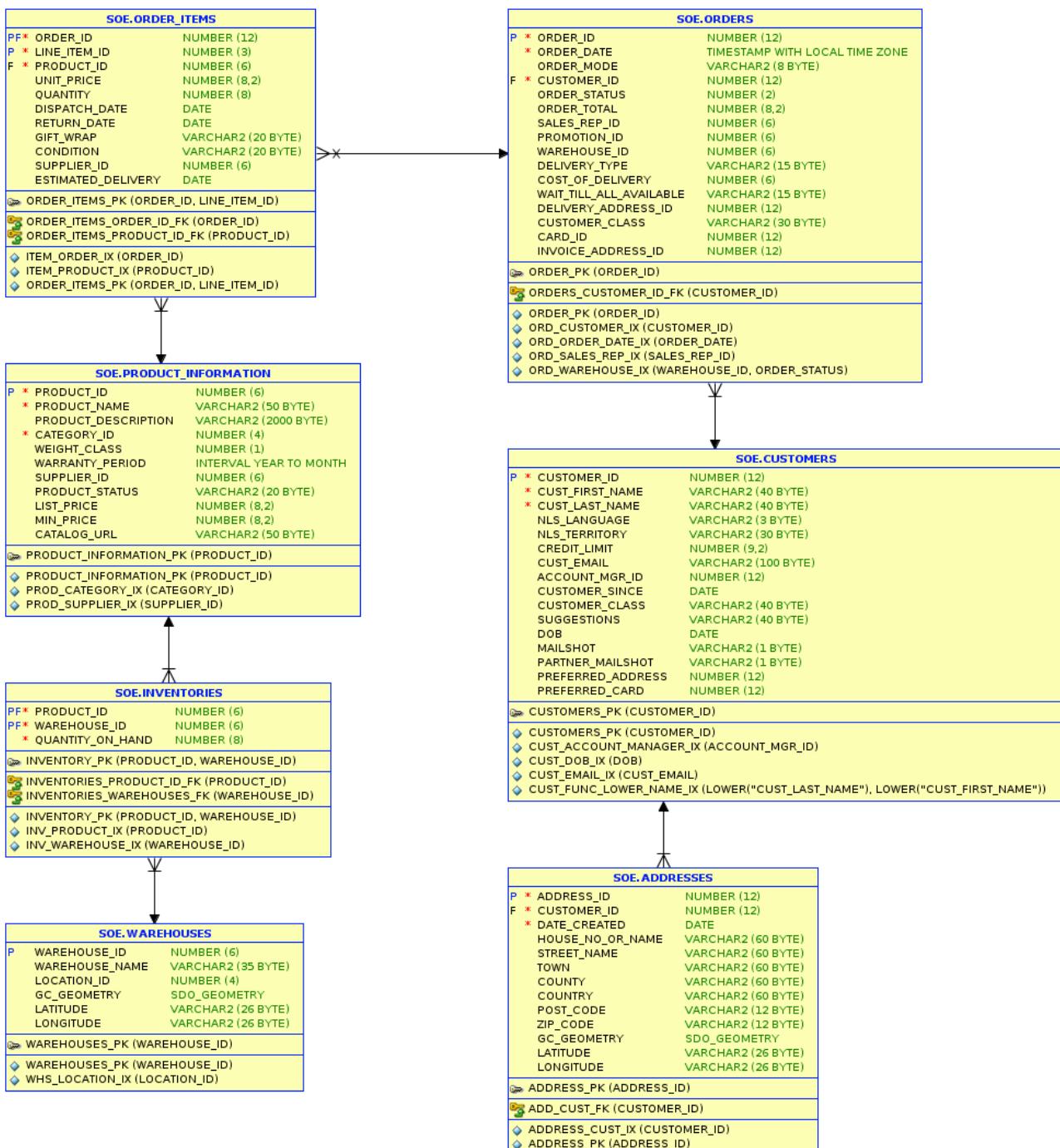
Enhorabuena, ha completado esta sección del workshop.



Transformación de datos relacionales de App1 en JSON para App2.

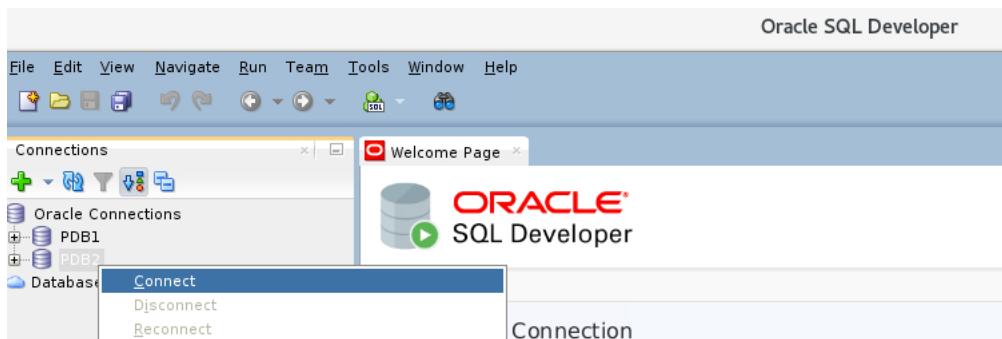
En esta actividad se va a extraer la información de los pedidos existente en el esquema SOE en PDB1 usado por App1 (que está almacenado en formato tabular relacional) para cargarlo en el esquema SOE en PDB2 en documentos JSON que usará App2 que es el microservicio que implementará ahora la funcionalidad de los pedidos. App2 está creado con NodeJS.

El modelo relacional en PDB1 es como se muestra a continuación:



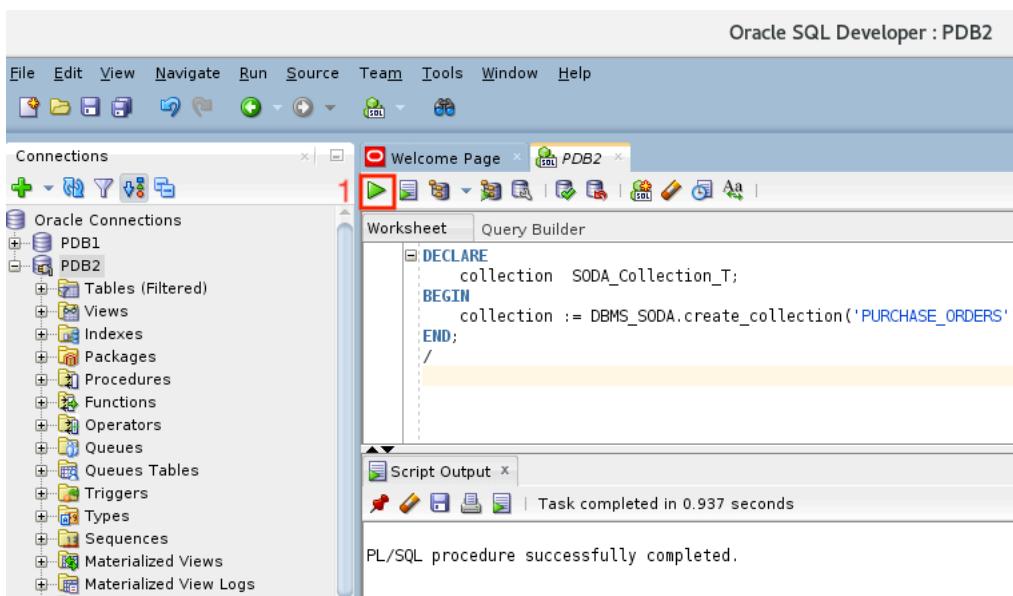
Una vez inspeccionado el modelo, se va a crear una nueva colección SODA en PDB2 que almacenará los datos de los pedidos (ORDERS) existentes en formato JSON para de esta manera facilitar la carga de esa información existente en App1 en el nuevo microservicio App2 que trabajará nativamente con este formato.

Usando SQL*Developer conecte con el usuario SOE a la PDB2.



Desde la pestaña de SQL Worksheet de SQL*Developer, cree la nueva colección SODA de documentos JSON PURCHASE_ORDERS copiando esta sentencia y pulsando el botón de ejecución (1).

```
DECLARE
    collection  SODA_Collection_T;
BEGIN
    collection := DBMS_SODA.create_collection('PURCHASE_ORDERS');
END;
/
```



Como se puede observar la colección PURCHASE_ORDERS tiene una columna de tipo JSON llamada JSON_DOCUMENT donde se almacenarán los documentos JSON.



COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID	VARCHAR2(255 BYTE)	No	(null)	1	(null)
2 CREATED_ON	TIMESTAMP(6)	No	sys_extract_utc(SYSTIMESTAMP)	2	(null)
3 LAST_MODIFIED	TIMESTAMP(6)	No	sys_extract_utc(SYSTIMESTAMP)	3	(null)
4 VERSION	VARCHAR2(255 BYTE)	No	(null)	4	(null)
5 JSON_DOCUMENT	JSON	Yes	(null)	5	(null)

Para transformar la información existente de pedidos, tras estudiar el modelo de datos en PDB1 se ha generado la siguiente consulta que encapsula los datos de las tablas ORDERS, CUSTOMERS, ADDRESSES y ORDER_ITEMS en un documento JSON.

```

select JSON_OBJECT (
    'ORDER_ID' value O.ORDER_ID,
    'ORDER_DATE' value O.ORDER_DATE,
    'ORDER_MODE' value O.ORDER_MODE,
    'CUSTOMER_ID' value O.CUSTOMER_ID,
    'CUSTOMER_NAME' value C.CUST_FIRST_NAME,
    'CUSTOMER_SURNAME' value C.CUST_LAST_NAME,
    'STREET_NAME' value A.STREET_NAME,
    'TOWN' value A.TOWN,
    'ORDER_STATUS' value O.ORDER_STATUS,
    'ORDER_TOTAL' value O.ORDER_TOTAL,
    'WAREHOUSE_ID' value O.WAREHOUSE_ID,
    'DELIVERY_TYPE' value O.DELIVERY_TYPE,
    'CUSTOMER_CLASS' value O.CUSTOMER_CLASS,
    'CARD_ID' value O.CARD_ID,
    'INVOICE_ADDRESS_ID' value O.INVOICE_ADDRESS_ID,
    'ITEMS' value json_arrayagg (
        json_object (
            'ORDER_ID' value OI.ORDER_ID,
            'LINE_ITEM_ID' value OI.LINE_ITEM_ID,
            'PRODUCT_ID' value OI.PRODUCT_ID,
            'UNIT_PRICE' value OI.UNIT_PRICE,
            'QUANTITY' value OI.QUANTITY,
            'DISPATCH_DATE' value OI.DISPATCH_DATE,
            'RETURN_DATE' value OI.RETURN_DATE,
            'SUPPLIER_ID' value OI.SUPPLIER_ID,
            'ESTIMATED_DELIVERY' value OI.ESTIMATED_DELIVERY
        )))
as jsonorders
from ORDERS O, order_items OI, CUSTOMERS C, ADDRESSES A
    where O.order_id = OI.order_id and
          O.CUSTOMER_ID = C.CUSTOMER_ID and
          O.DELIVERY_ADDRESS_ID = A.ADDRESS_ID
group by
O.ORDER_ID,
O.ORDER_DATE,
O.ORDER_MODE,
O.CUSTOMER_ID,
C.CUST_FIRST_NAME,
C.CUST_LAST_NAME,
A.STREET_NAME,
A.TOWN,
O.ORDER_STATUS,
O.ORDER_TOTAL,
O.WAREHOUSE_ID,
O.DELIVERY_TYPE,
O.CUSTOMER_CLASS,
O.CARD_ID,
O.INVOICE_ADDRESS_ID;

```



Esta consulta se puede probar desde SQL*Developer conectando a la PDB1 para observar los documentos JSON generados.

```

Connections          Welcome Page × PDB2 × PDB1 ×
Worksheet | Query Builder
select JSON_OBJECT (
    'ORDER_ID' value O.ORDER_ID,
    'ORDER_DATE' value O.ORDER_DATE,
    'ORDER_MODE' value O.ORDER_MODE,
    'CUSTOMER_ID' value O.CUSTOMER_ID,
    'CUSTOMER_NAME' value C.CUST_FIRST_NAME,
    'CUSTOMER_SURNAME' value C.CUST_LAST_NAME,
    'STREET_NAME' value A.STREET_NAME,
    'TOWN' value A.TOWN,
    'ORDER_STATUS' value O.ORDER_STATUS,
    'ORDER_TOTAL' value O.ORDER_TOTAL,
    'WAREHOUSE_ID' value O.WAREHOUSE_ID,
    'DELIVERY_TYPE' value O.DELIVERY_TYPE,
    'CUSTOMER_CLASS' value O.CUSTOMER_CLASS,
    'CARD_ID' value O.CARD_ID,
    'INVOICE_ADDRESS_ID' value O.INVOICE_ADDRESS_ID,
    'ITEMS' value json_arrayagg (
        ...
    )
) JSONORDERS
from ORDERS O
join CUSTOMERS C on O.CUSTOMER_ID = C.CUSTOMER_ID
join ADDRESS A on O.SHIP_TO_ID = A.ADDRESS_ID
where O.ORDER_DATE > '2007-01-01'
order by O.ORDER_ID
Query Result x
SQL | Fetched 50 rows in 0.139 seconds
JSONORDERS
1 {"ORDER_ID":1,"ORDER_DATE":"2007-05-12T04:00:00.000000Z","ORDER_MODE":"direct","CUSTOMER_ID":56,"CUSTOMER_NAME":"joseph"
2 {"ORDER_ID":2,"ORDER_DATE":"2007-02-28T08:00:00.000000Z","ORDER_MODE":null,"CUSTOMER_ID":12,"CUSTOMER_NAME":"ralph","CUS
3 {"ORDER_ID":3,"ORDER_DATE":"2008-12-29T19:00:00.000000Z","ORDER_MODE":"direct","CUSTOMER_ID":39,"CUSTOMER_NAME":"jacob",
4 {"ORDER_ID":4,"ORDER_DATE":"2007-07-20T09:00:00.000000Z","ORDER_MODE":"direct","CUSTOMER_ID":6,"CUSTOMER_NAME":"rick","C
5 {"ORDER_ID":5,"ORDER_DATE":"2010-04-12T17:00:00.000000Z","ORDER_MODE":"direct","CUSTOMER_ID":52,"CUSTOMER_NAME":"douglas
6 {"ORDER_ID":6,"ORDER_DATE":"2007-07-30T22:00:00.000000Z","ORDER_MODE":"online","CUSTOMER_ID":92,"CUSTOMER_NAME":"duane",
7 {"ORDER_ID":7,"ORDER_DATE":"2007-11-18T16:00:00.000000Z","ORDER_MODE":null,"CUSTOMER_ID":18,"CUSTOMER_NAME":"wendell","C
8 {"ORDER_ID":8,"ORDER_DATE":"2011-09-06T11:00:00.000000Z","ORDER_MODE":"online","CUSTOMER_ID":40,"CUSTOMER_NAME":"randy",
9 {"ORDER_ID":9,"ORDER_DATE":"2008-09-27T05:00:00.000000Z","ORDER_MODE":"online","CUSTOMER_ID":34,"CUSTOMER_NAME":"cliffor
10 {"ORDER_ID":10,"ORDER_DATE":"2008-04-24T03:00:00.000000Z","ORDER_MODE":"direct","CUSTOMER_ID":94,"CUSTOMER_NAME":"jesus"
11 {"ORDER_ID":11,"ORDER_DATE":"2010-12-06T04:00:00.000000Z","ORDER_MODE":"direct","CUSTOMER_ID":17,"CUSTOMER_NAME":"chad",
12 {"ORDER_ID":12,"ORDER_DATE":"2009-02-05T14:00:00.000000Z","ORDER_MODE":null,"CUSTOMER_ID":44,"CUSTOMER_NAME":"leanne"

```

El objetivo es cargar todos estos documentos JSON en la colección SODA de PDB2. Para este proceso de migración se va a crear en PDB2 un database link a PDB1 entre los usuarios soe.

```
create database link db_pdb1 connect to soe identified by "soe" using
'localhost:1521/pdb1';
```

```

Connections          Welcome Page × PDB2 ×
Worksheet | Query Builder
create database link db_pdb1 connect to soe
identified by "soe" using 'localhost:1521/pdb1';
Script Output x
Task completed in 0.111 seconds
Database link DB_PDB1 created.

```

Usando este db link y la consulta anterior que encapsula los datos en JSON, se va a cargar toda la información de los pedidos de las tablas de PDB1 a la colección SODA de PDB2 usando la siguiente sentencia.



```

DECLARE
    collection  SODA_COLLECTION_T;
    document     SODA_DOCUMENT_T;
    status       NUMBER;
    CURSOR c_orders IS
    select JSON_OBJECT (
        'ORDER_ID' value O.ORDER_ID,
        'ORDER_DATE' value O.ORDER_DATE,
        'ORDER_MODE' value O.ORDER_MODE,
        'CUSTOMER_ID' value O.CUSTOMER_ID,
        'CUSTOMER_NAME' value C.CUST_FIRST_NAME,
        'CUSTOMER_SURNAME' value C.CUST_LAST_NAME,
        'STREET_NAME' value A.STREET_NAME,
        'TOWN' value A.TOWN,
        'ORDER_STATUS' value O.ORDER_STATUS,
        'ORDER_TOTAL' value O.ORDER_TOTAL,
        'WAREHOUSE_ID' value O.WAREHOUSE_ID,
        'DELIVERY_TYPE' value O.DELIVERY_TYPE,
        'CUSTOMER_CLASS' value O.CUSTOMER_CLASS,
        'CARD_ID' value O.CARD_ID,
        'INVOICE_ADDRESS_ID' value O.INVOICE_ADDRESS_ID,
        'ITEMS' value json_arrayagg (
            json_object (
                'ORDER_ID' value OI.ORDER_ID,
                'LINE_ITEM_ID' value OI.LINE_ITEM_ID,
                'PRODUCT_ID' value OI.PRODUCT_ID,
                'UNIT_PRICE' value OI.UNIT_PRICE,
                'QUANTITY' value OI.QUANTITY,
                'DISPATCH_DATE' value OI.DISPATCH_DATE,
                'RETURN_DATE' value OI.RETURN_DATE,
                'SUPPLIER_ID' value OI.SUPPLIER_ID,
                'ESTIMATED_DELIVERY' value OI.ESTIMATED_DELIVERY
            )))
    as jsonorders
    from ORDERS@db_pdb1 O, order_items@db_pdb1 OI, CUSTOMERS@db_pdb1 C, ADDRESSES@db_pdb1 A
    where O.order_id = OI.order_id and
          O.CUSTOMER_ID = C.CUSTOMER_ID and
          O.INVOICE_ADDRESS_ID = A.ADDRESS_ID
group by
O.ORDER_ID,
O.ORDER_DATE,
O.ORDER_MODE,
O.CUSTOMER_ID,
C.CUST_FIRST_NAME,
C.CUST_LAST_NAME,
A.STREET_NAME,
A.TOWN,
O.ORDER_STATUS,
O.ORDER_TOTAL,
O.WAREHOUSE_ID,
O.DELIVERY_TYPE,
O.CUSTOMER_CLASS,
O.CARD_ID,
O.INVOICE_ADDRESS_ID;

BEGIN
    -- Open the collection
    collection := DBMS_SODA.open_collection('PURCHASE_ORDERS');

    FOR cur in c_orders
    LOOP
        document := SODA_DOCUMENT_T(b_content => utl_raw.cast_to_raw(cur.jsonorders));
        -- Insert a document
        status := collection.insert_one(document);
    END LOOP;

END;
/
commit;

```

Ejecútala desde SQL*Developer en PDB2 para completar la migración de los pedidos.



The screenshot shows the Oracle SQL Developer interface. On the left, the Connections sidebar lists 'Oracle Connections' with entries for 'PDB1' and 'PDB2'. The 'PDB2' entry is expanded, showing categories like 'Tables (Filtered)', 'Views', 'Indexes', 'Packages', 'Procedures', 'Functions', 'Operators', 'Queues', 'Queues Tables', 'Triggers', 'Types', 'Sequences', 'Materialized Views', 'Materialized View Logs', 'Synonyms', 'Public Synonyms', 'Database Links', 'Public Database Links', 'Directories', 'Editions', and 'Java'. Below this is a 'Reports' section with 'All Reports'. The main workspace is titled 'Welcome Page' and contains a 'Worksheet' tab and a 'Query Builder' tab. The Worksheet tab displays a PL/SQL script for creating a database link:

```
DECLARE
  collection SODA_COLLECTION_T;
  document SODA_DOCUMENT_T;
  status NUMBER;
  CURSOR c_orders IS
    select JSON_OBJECT (
      'ORDER_ID' value O.ORDER_ID,
      'ORDER_DATE' value O.ORDER_DATE,
      'ORDER_MODE' value O.ORDER_MODE,
      'CUSTOMER_ID' value O.CUSTOMER_ID,
      'CUSTOMER_NAME' value C.CUST_FIRST_NAME,
      'CUSTOMER_SURNAME' value C.CUST_LAST_NAME,
      'STREET_NAME' value A.STREET_NAME,
      'TOWN' value A.TOWN,
      'ORDER_STATUS' value O.ORDER_STATUS,
      'ORDER_TOTAL' value O.ORDER_TOTAL,
      'WAREHOUSE_ID' value O.WAREHOUSE_ID,
      'DELIVERY_TYPE' value O.DELIVERY_TYPE,
      'CUSTOMER_CLASS' value O.CUSTOMER_CLASS,
      'CARD_ID' value O.CARD_ID,
    )
```

Below the Worksheet is a 'Script Output' tab showing the result of the execution:

```
Task completed in 0.271 seconds
```

At the bottom, a message indicates the success of the database link creation:

```
Database link DB_PDB1 created.
```

Another message at the bottom right states:

```
PL/SQL procedure successfully completed.
```

Ahora la información de los pedidos está cargada en formato JSON en la colección PURCHASE ORDERS.

A modo de comprobación, la siguiente consulta extrae algunos atributos de los pedidos a partir de los documentos JSON en forma de tabla relacional:

```
select j.JSON_DOCUMENT.ORDER_ID,  
       j.JSON_DOCUMENT.CUSTOMER_NAME,  
       j.JSON_DOCUMENT.CUSTOMER_SURNAME,  
       j.JSON_DOCUMENT.TOWN.string() AS TOWN  
  from PURCHASE_ORDERS j  
 where j.JSON_DOCUMENT.CUSTOMER_NAME.string() like 'jo%';
```

The screenshot shows the Oracle SQL Developer interface. On the left, the Connections tree displays two PDBs: PDB1 and PDB2. The PDB2 node is expanded, showing categories like Tables (Filtered), Views, Indexes, Packages, Procedures, Functions, Operators, Queues, Queues Tables, Triggers, Types, Sequences, Materialized Views, Materialized View Logs, Synonyms, Public Synonyms, Database Links, Public Database Links, Directories, Editions, and Java.

The main workspace contains a Worksheet tab with the following SQL query:

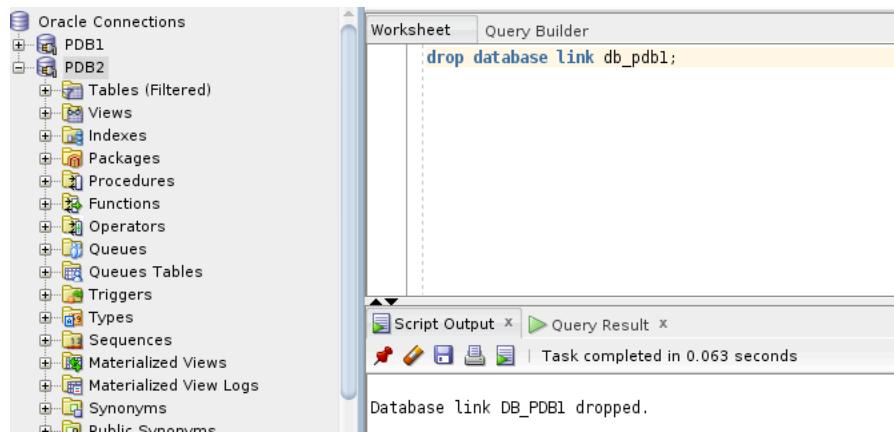
```
select j.JSON_DOCUMENT.ORDER_ID,
       j.JSON_DOCUMENT.CUSTOMER_NAME,
       j.JSON_DOCUMENT.CUSTOMER_SURNAME,
       j.JSON_DOCUMENT.TOWN.string() AS TOWN
  from PURCHASE_ORDERS j
 where j.JSON_DOCUMENT.CUSTOMER_NAME.string() like 'jo%';
```

Below the worksheet is a Script Output tab showing the execution results:

ORDER_ID	CUSTOMER_NAME	CUSTOMER_SURNAME	TOWN
1 13	"john"	"johnson"	Villanueva d
2 14	"john"	"johnson"	Villanueva d
3 22	"john"	"johnson"	Villanueva d
4 1	"joseph"	"rodriguez"	Villanueva d
5 36	"joseph"	"rodriguez"	Villanueva d
6 71	"joseph"	"rodriguez"	Villanueva d
7 48	"john"	"johnson"	Villanueva d

Opcionalmente elimine el database link.

```
drop database link db_pdb1;
```



Enhorabuena, ha completado esta sección del workshop.

Despliegue de App2 (NodeJS + SODA) usando PDB2

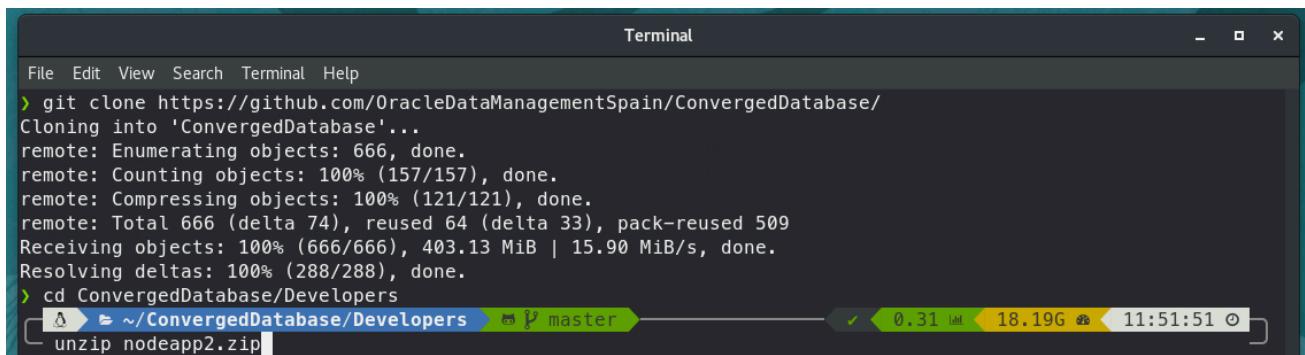
La nueva aplicación de pedidos App2 ha sido desarrollada en NodeJS. Esta aplicación trabaja con documentos JSON que almacena en su propio repositorio que estará en la PDB2. Para ello incluye el driver de SODA que le permite trabajar en modo colección de documentos sin necesidad de preocuparse de usar SQL o cómo se almacenen esos documentos.

En esta actividad se usará Podman (<https://podman.io>) para ejecutar la aplicación desarrollada en NodeJS en forma de un contenedor. Podman es un engine de contenedores desarrollado por RedHat muy parecido al más conocido Docker. Las actividades de este manual se podrían realizar con cualquiera de los dos.

Para recuperar el código de la aplicación, desde un terminal, clone el siguiente repositorio de Github:

```
git clone https://github.com/OracleDataManagementSpain/ConvergedDatabase/
```

Cambie al directorio `ConvergedDatabase/Developers` y descomprima el fichero `nodeapp2.zip`



En la carpeta nodeapp2 verá que hay un fichero Dockerfile que contiene lo necesario para empaquetar esta aplicación en forma de un contenedor así como las credenciales y cadena de conexión a la base de datos.

```
inflating: nodeapp2/views/detail.ejs
inflating: nodeapp2/views/index.ejs
inflating: nodeapp2/views/add.ejs
inflating: nodeapp2/views/orders.ejs
> cd nodeapp2
> ls
app.js          createService.js    getDataServiceCity.js  package-lock.json  views
bin             dbconfig.js        getDataService.js      public
createP0Service.js  deleteAllService.js  getDetailService.js  Readme.md
createServiceCity.js Dockerfile       package.json        routes
>
>
>
>
```

Compruebe que son correctas conectando con SQLcl, por ejemplo como se muestra a continuación:

```
Terminal
File Edit View Search Terminal Help
> head Dockerfile
FROM node:14
EXPOSE 3000

ENV NODE_ENV=production
ENV NODE_ORACLEDB_USER=soe
ENV NODE_ORACLEDB_PASSWORD=soe
ENV NODE_ORACLEDB_CONNECTIONSTRING=iaas21c.subnet.vcn.oraclevcn.com/pdb2
ENV SODA_COLLECTION_NAME=PURCHASE_ORDERS

RUN apt-get update && apt-get install -y libaio1 wget unzip
> sql soe/soe@iaas21c.subnet.vcn.oraclevcn.com/pdb2

SQLcl: Release 21.2 Production on Thu Feb 24 11:54:12 2022
Copyright (c) 1982, 2022, Oracle. All rights reserved.

Connected to:
Oracle Database 21c Enterprise Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

SQL> exit
Disconnected from Oracle Database 21c Enterprise Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0
~ /ConvergedDatabase/Developers/nodeapp2 ➜ master ?1
```

Genere la imagen del contenedor con el siguiente comando:

```
podman build --tag nodeapp-container-oracle .
```



```
Terminal
File Edit View Search Terminal Help
└─▶ ~/ConvergedDatabase/Developers/nodeapp2 └─▶ master ?1
podman build --tag nodeapp-container-oracle .
```

Seleccione el repositorio docker.io usando la flecha hacia abajo de su teclado:

```
› podman build --tag nodeapp-docker-oracle .
STEP 1: FROM node:14
? Please select an image:
  container-registry.oracle.com/node:14
  ▶ docker.io/library/node:14
```

Tras unos minutos, la imagen estará lista en el repositorio local, se puede comprobar con:

```
podman image ls
```

```
STEP 25: COMMIT nodeapp-container-oracle
--> 0fee7a3956d
Successfully tagged localhost/nodeapp-container-oracle:latest
0fee7a3956dcc9d33b903701a921e94495a68923924f45670c9e9037822c83fd
› podman image ls
REPOSITORY          TAG      IMAGE ID   CREATED    SIZE
localhost/nodeapp-container-oracle  latest   0fee7a3956dc  58 seconds ago  1.11 GB
docker.io/library/node       14      9cb3f042a684  3 weeks ago   975 MB
└─▶ ~/ConvergedDatabase/Developers/nodeapp2 └─▶ master ?1
```

Esta imagen se basa en la imagen oficial de Node y contiene además el código de la aplicación de pedidos así como el driver de SODA.

Se puede arrancar con el siguiente comando:

```
podman run -it -d --name app2 -p 3000:3000 nodeapp-container-oracle
```

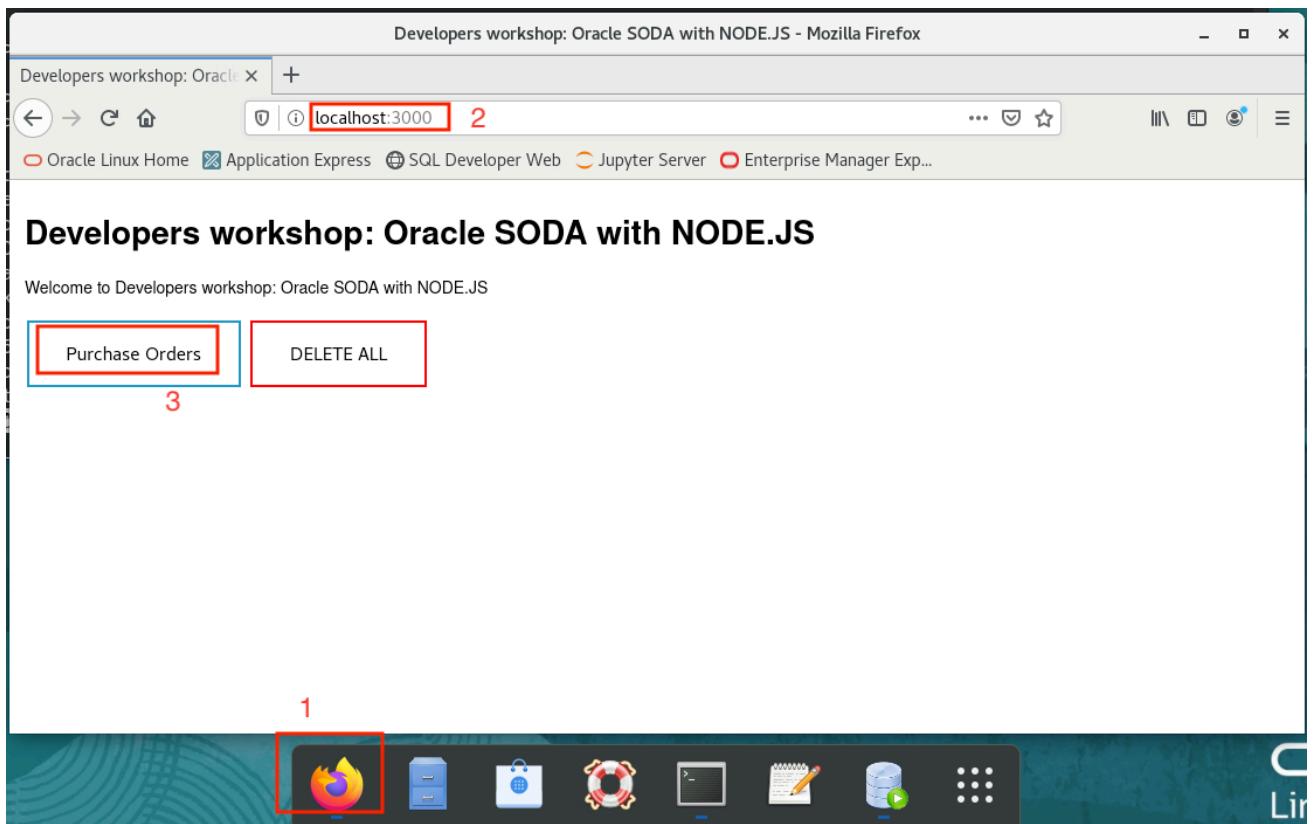
```
› podman run -it -d --name app2 -p 3000:3000 nodeapp-container-oracle
a2b0da7110b5c9e56b5c83783c467a1e406825e818ce24e8d759f9e1951db5f5
› podman ps
CONTAINER ID  IMAGE          COMMAND      CREATED     STATUS      PORTS
 NAMES
a2b0da7110b5  localhost/nodeapp-container-oracle:latest  node ./bin/www  4 seconds ago  Up 4 seconds ago  0.0.0.0:3000->3000
/tcp app2
```

Una vez iniciada, está accesible en el puerto 3000 por medio del navegador.

Arranque Firefox desde el icono del dock (1) e introduzca <http://localhost:3000> en la barra de direcciones (2).

Una vez se cargue la pantalla, haga click en el botón ‘Purchase Orders’(3) para que se inicialice la conexión a la base de datos y se lea la colección SODA de documentos JSON si es necesario.





En la pantalla verá toda la lista de pedidos migrados desde App1.

Date	Customer Name	Customer Last Name	Address	City	Price	Status	Detail
2007-07-30T22:00:00.000000Z	duane	richards	Camino Real 5	Villanueva del Pardillo	5963	6	Detail
2007-11-18T16:00:00.000000Z	wendell	rhodes	Calle Higuera 8	Las Rozas de Madrid	3921	4	Detail
2011-09-06T11:00:00.000000Z	randy	bennett	Av. de Madrid 20	Villanueva del Pardillo	5041	5	Detail

Puede también crear un nuevo pedido haciendo click en el botón Create PO.



En la pantalla de creación de la PO (Purchase Order), por comodidad se ha incluido un botón de **Default Values** (1) que rellena el formulario con valores por defecto. En la parte inferior está el botón **Create** (2) que envía el pedido a la colección SODA en la base de datos.

The screenshot shows a web-based application titled 'Create PO'. At the top left is a 'Default values' button, which is highlighted with a red box and labeled '1'. Below it, there are two customer entries: 'Customer name: Ramiro' and 'Customer name: Sanchez'. Under 'Address', it says 'Address: Calle del Fresno 21'. Under 'City', it says 'City: Mostoles'. A 'Total price:' field contains '310'. The form is divided into sections for 'Item line 1' and 'Item line 2'. 'Item line 1' includes 'Item name: Monitor', 'Item quantity: 1', and 'Item price: 300'. 'Item line 2' includes 'Item name: HDMI cable', 'Item quantity: 1', and 'Item price: 10'. At the bottom right is a 'Create' button, which is highlighted with a red box and labeled '2'.

Una vez creado, se informa del identificador único en la colección que almacena el pedido:

The screenshot shows a table titled 'Purchase orders'. It displays a single row of data with the following columns: Date, Customer Name, Customer Last Name, Address, City, Price, Status, and Detail. The data in the table is:

Date	Customer Name	Customer Last Name	Address	City	Price	Status	Detail
2007-07-30T22:00:00.000000Z	duane	richards	Camino Real 5	Villanueva del Pardillo	5963	6	Detail

At the top left of the table area, there is a message: 'New PO created with unique ID: 69EBC4D1128A4F25BF1F4798733C64C9', with the ID part highlighted by a red box. There is also a 'Create PO' button at the top left of the table area.



Este identificador se puede usar para localizar el pedido en la colección SODA desde la base de datos. En SQL*Developer, conectado a PDB2 ejecute la siguiente consulta usando el identificador único que recibió en su navegador:

```
select j.JSON_DOCUMENT.ORDER_ID,
       j.JSON_DOCUMENT.CUSTOMER_NAME,
       j.JSON_DOCUMENT.CUSTOMER_SURNAME,
       j.JSON_DOCUMENT.TOWN.string() AS TOWN
  from PURCHASE_ORDERS j
 where ID='id único del navegador';
```

The screenshot shows the Oracle SQL Developer interface. On the left, the 'Connections' sidebar lists 'Oracle Connections' with 'PDB1' and 'PDB2' selected. Under 'PDB2', the 'Tables (Filtered)' section shows 'PURCHASE_ORDERS' with columns: ID, CREATED_ON, LAST_MODIFIED, VERSION, and JSON_DOCUMENT. Below this are sections for Views, Indexes, Packages, Procedures, Functions, Operators, Queues, and Queues Tables. The central workspace shows a query in the 'Worksheet' tab:

```
select j.JSON_DOCUMENT.ORDER_ID,
       j.JSON_DOCUMENT.CUSTOMER_NAME,
       j.JSON_DOCUMENT.CUSTOMER_SURNAME,
       j.JSON_DOCUMENT.TOWN.string() AS TOWN
  from PURCHASE_ORDERS j
 where ID='69EBC4D1128A4F25BF1F4798733C64C9';
```

Below the worksheet is the 'Script Output' tab, which displays the results of the query execution:

ORDER_ID	CUSTOMER_NAME	CUSTOMER_SURNAME	TOWN
1 2085	"Ramiro"	"Sanchez"	Mostoles

Compruebe que el nuevo pedido está en el listado de la página web de App2:

24/2/2022

Ramiro

Sanchez

Calle del Fresno
21

Mostoles

310

6

[Detail](#)

Enhorabuena, ha completado esta sección del workshop.

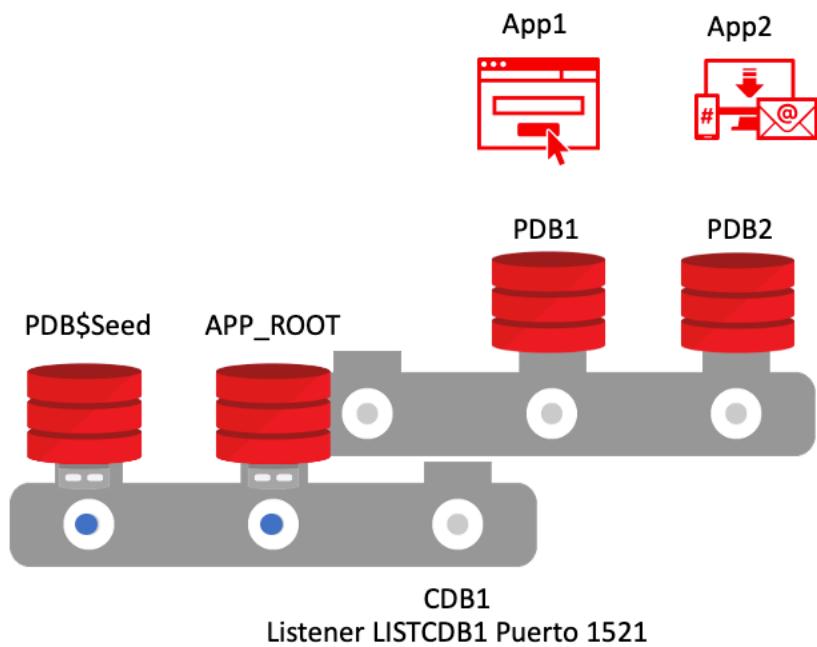
Además de SODA, Oracle dispone de **Oracle Database API for MongoDB** que permite a las aplicaciones interaccionar con colecciones de documentos JSON usando los comandos de MongoDB. Este componente traduce *MongoDB wire protocol* a sentencias que son ejecutadas en la base de datos Oracle, de forma que las aplicaciones pueden seguir usando los driver, frameworks y herramientas que ya conocen, pero el motor de almacenamiento y ejecución es Oracle.

Más información en: <https://docs.oracle.com/en/database/oracle/mongodb-api/mgapi/overview-oracle-database-api-mongodb.html>



Consulta combinada de la información de App1 y App2

En este momento están conviviendo la aplicación heredada App1 y la nueva App2 usando sus respectivos almacenes de información en dos PDB: PDB1 y PDB2 respectivamente que están bajo el control del Application Container APP_ROOT.



En esta última actividad se va a realizar desde APP_ROOT una consulta de información que se encuentra en PDB1 y PDB2. El resultado de esta consulta se va a formalizar en forma de una vista que se va a publicar en un API REST para su consumo desde otras aplicaciones que la necesiten.

Para llevar a cabo esta tarea es necesario crear en el APP_ROOT un esquema con unas tablas similares a las que se van a consultar en PDB1 y PDB2, pero que en APP_ROOT estarán vacías.

En PDB1 y PDB2 no se realiza ningún tipo de modificación en sus esquemas, de modo que no se interfiere con las aplicaciones y es completamente transparente para ellas.

En primer lugar se crea un usuario soe en el app_root desde el que trabajar.

```
$ sql sys/Oracle_4U@localhost:1521/app_root as sysdba

SQLcl: Release 21.2 Production on Tue Jan 04 17:59:41 2022
Copyright (c) 1982, 2022, Oracle. All rights reserved.

Connected to:
Oracle Database 21c Enterprise Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

SQL> create tablespace TBS_SOE datafile size 100M;
```



```
Tablespace TBS_SOE created.

SQL> create user SOE identified by soe default tablespace TBS_SOE temporary tablespace TEMP;

User SOE created.

SQL> alter user SOE quota unlimited on TBS_SOE;

User SOE altered.

SQL> grant connect, resource to SOE;

Grant succeeded.

SQL> grant create view to soe;

Grant succeeded.

SQL> grant graph_developer to soe;

Grant succeeded.

SQL> grant soda_app to soe;

Grant succeeded.
```

Ahora se cambia al usuario soe para crear las tablas y las vistas. Para este ejercicio se usará:

- De PDB1 la tabla CUSTOMERS
- De PDB2 la colección PURCHASE_ORDERS

Por ello se van a crear estos dos mismos objetos en el esquema SOE del APP_ROOT pero vacíos.

En primer lugar la tabla CUSTOMERS.

```
SQL> conn soe/soe@localhost:1521/app_root

Connected.

SQL> CREATE TABLE "CUSTOMERS"
  (  "CUSTOMER_ID" NUMBER(12,0) ,
  "CUST_FIRST_NAME" VARCHAR2(40 BYTE),
  "CUST_LAST_NAME" VARCHAR2(40 BYTE),
  "NLS_LANGUAGE" VARCHAR2(3 BYTE),
  "NLS_TERRITORY" VARCHAR2(30 BYTE),
  "CREDIT_LIMIT" NUMBER(9,2),
  "CUST_EMAIL" VARCHAR2(100 BYTE),
  "ACCOUNT_MGR_ID" NUMBER(12,0),
  "CUSTOMER_SINCE" DATE,
  "CUSTOMER_CLASS" VARCHAR2(40 BYTE),
  "SUGGESTIONS" VARCHAR2(40 BYTE),
  "DOB" DATE,
  "MAILSHOT" VARCHAR2(1 BYTE),
  "PARTNER_MAILSHOT" VARCHAR2(1 BYTE),
  "PREFERRED_ADDRESS" NUMBER(12,0),
  "PREFERRED_CARD" NUMBER(12,0)
```



```

);

Table "CUSTOMERS" created.

SQL> select count(*) from CUSTOMERS;

COUNT(*)
_____
0

```

Aunque la tabla está vacía, si se añade la cláusula **containers**, se puede ver que entonces la consulta tiene en cuenta los registros de las tablas del mismo nombre en las PDBs dentro del application container.

```

SQL> select count(*) from containers(CUSTOMERS);

COUNT(*)
_____
100

```

Ahora la colección SODA PURCHASE_ORDERS

```

SQL> DECLARE
      collection  SODA_Collection_T;
    BEGIN
      collection := DBMS_SODA.create_collection('PURCHASE_ORDERS');
    END;
/

PL/SQL procedure successfully completed.

SQL> select count(1) from PURCHASE_ORDERS;

COUNT(1)
_____
0

SQL> select count(1) from containers(PURCHASE_ORDERS);

COUNT(1)
_____
149

```

Para mayor comodidad se usarán unas vistas.

```

SQL> create or replace view V_CUSTOMERS as select * from containers(CUSTOMERS);

View V_CUSTOMERS created.

SQL> create or replace view V_PURCHASE_ORDERS AS select * from
containers(PURCHASE_ORDERS);

View V_PURCHASE_ORDERS created.

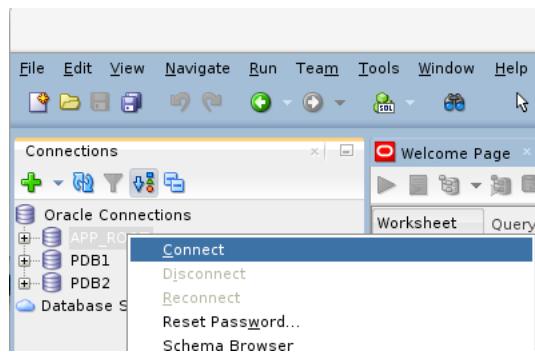
```



Exposición de datos en API REST con ORDS.

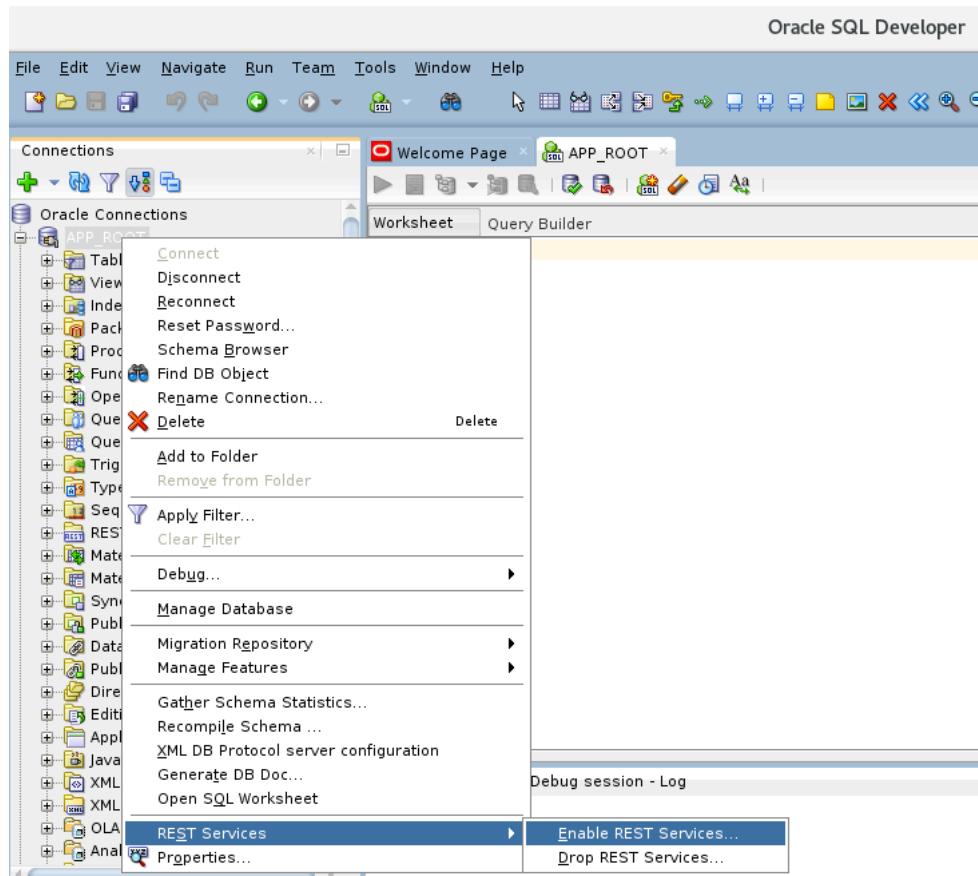
Para concluir esta sesión, se va a exponer esta información en un api REST usando ORDS (Oracle Rest Data Services), servicio que viene incluido sin coste con la Base de Datos Oracle. Este producto puede descargarse en <https://www.oracle.com/es/database/technologies/appdev/rest.html> y también encontrará en este sitio web la documentación.

En primer lugar debe activarse el esquema para ORDS. Esto se puede realizar de forma sencilla en SQL*Developer. Conecte con el usuario SOE de APP_ROOT:

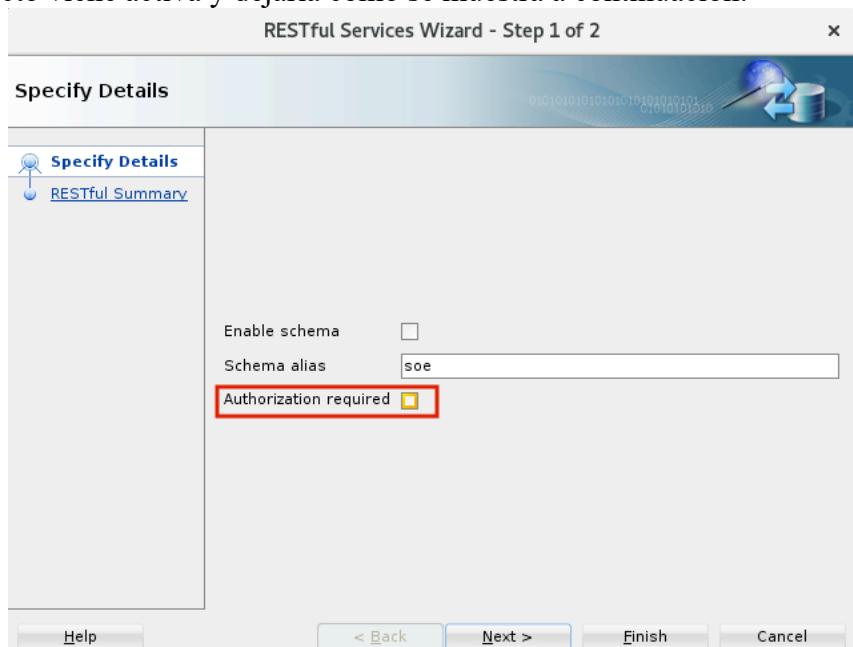


Haga botón derecho sobre la conexión, escoja **REST Services** y **Enable REST Services** como se muestra en la siguiente imagen.



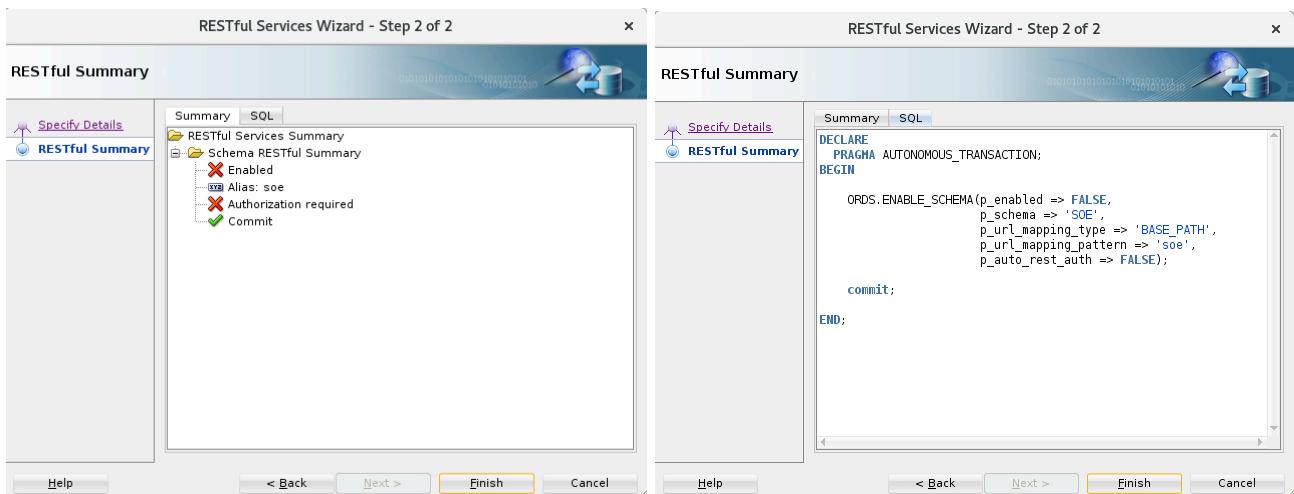


Para simplificar la actividad no se va a activar la autorización, entonces debe deseleccionar la check box que por defecto viene activa y dejarla como se muestra a continuación:

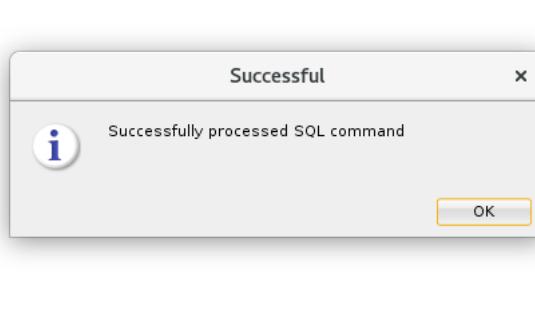


El alias del esquema se usa para construir la URL de acceso a los objetos que se publiquen desde el mismo, tal y como verá en los siguientes pasos.
Se muestra un resumen, con la posibilidad de ver en la pestaña SQL el comando que se va a ejecutar:



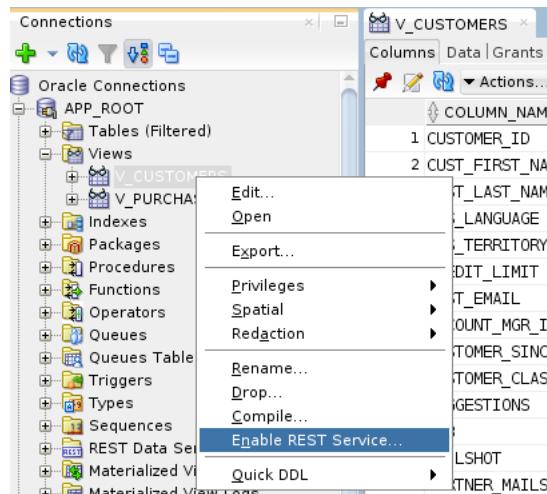


Haga click en el botón Finish para terminar la activación del esquema en ORDS.



Una vez activo el esquema, la manera más sencilla de publicar un objeto en el API es usando el método automático. Consulte la documentación para publicar desde consultas SQL a llamadas a procedimientos PL/SQL con parámetros.

Despliegue el nodo de las vistas (Views) y haga botón derecho sobre V_CUSTOMERS, seleccione la opción **Enable REST Service**.



En el diálogo que se muestra, haga click en Enable object y al igual que en el caso anterior, por simplicidad, desactive la autorización, tal y como se muestra a continuación:





El alias del objeto que está publicando (que por defecto es el nombre del objeto) se usará para construir la URL de acceso al mismo.

Haga click en **Next** para ver un resumen de las opciones escogidas, con la posibilidad de inspeccionar el SQL que va asociado a esta operación en la pestaña **SQL**:

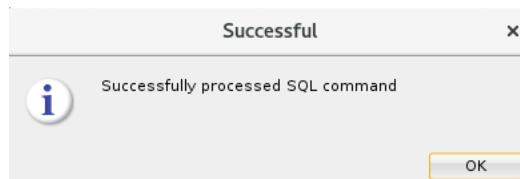
```

DECLARE
  PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
  ORDS.ENABLE_OBJECT(p_enabled => TRUE,
    p_schema => 'SOE',
    p_object => 'V_CUSTOMERS',
    p_object_type => 'VIEW',
    p_object_alias => 'v_customers',
    p_auto_rest_auth => FALSE);

  commit;

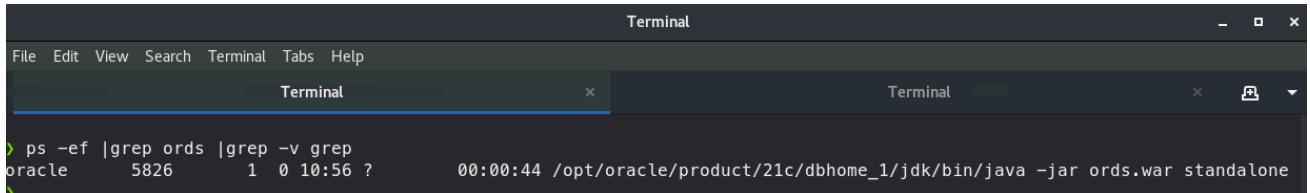
END;
  
```

Haga click en **Finish** para publicar esta vista en ORDS y así publicarla en el API Rest de ORDS.



ORDS se encuentra ya instalado y en ejecución en su máquina de laboratorio. Puede encontrar el proceso ejecutando:

```
ps -ef | grep ords | grep -v grep
```



```
Terminal
File Edit View Search Terminal Tabs Help
Terminal Terminal
> ps -ef |grep ords |grep -v grep
oracle      5826      1  0 10:56 ?          00:00:44 /opt/oracle/product/21c/dbhome_1/jdk/bin/java -jar ords.war standalone
```

Por defecto ORDS está escuchando en el puerto 8080. La forma de acceder a los objetos que se publican en este servicio de la forma que se acaba de mostrar es como sigue:

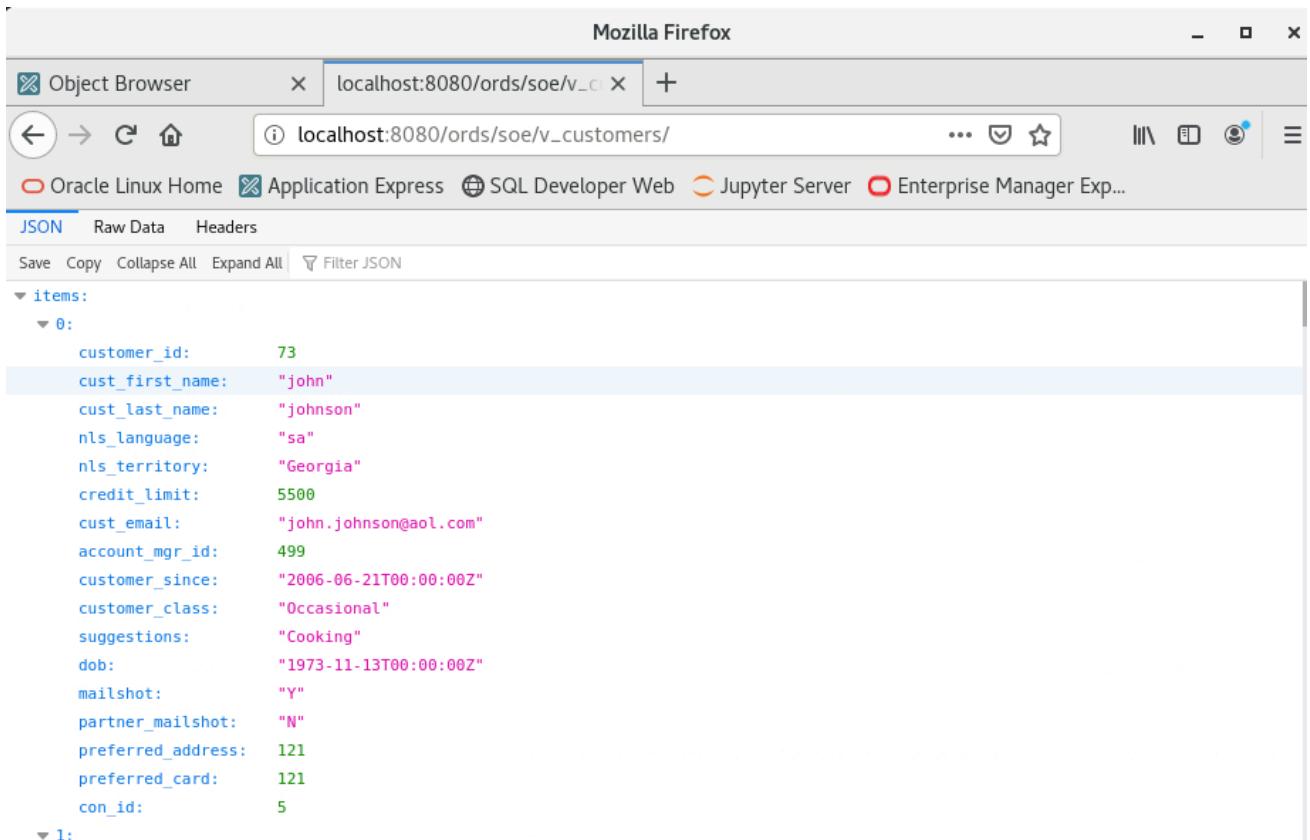
```
http://server\_name\_or\_ip\_address:port/ords/schema\_alias/object\_alias/
```

Para el ejemplo concreto que se acaba de publicar, teniendo en cuenta que los alias del esquema y la vista coinciden con sus nombres, sería como sigue:

```
http://localhost:8080/ords/soe/v\_customers/
```

Esta url se puede usar desde el navegador, ya que en este caso la operación que se va a realizar es un HTTP GET.

Así es cómo se ve el resultado de la consulta desde el navegador cuando se accede a ella:



Mozilla Firefox

Object Browser localhost:8080/ords/soe/v_customers/

localhost:8080/ords/soe/v_customers/

Oracle Linux Home Application Express SQL Developer Web Jupyter Server Enterprise Manager Exp...

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

items:

0:

customer_id:	73
cust_first_name:	"john"
cust_last_name:	"johnson"
nls_language:	"sa"
nls_territory:	"Georgia"
credit_limit:	5500
cust_email:	"john.johnson@aol.com"
account_mgr_id:	499
customer_since:	"2006-06-21T00:00:00Z"
customer_class:	"Occasional"
suggestions:	"Cooking"
dob:	"1973-11-13T00:00:00Z"
mailshot:	"Y"
partner_mailshot:	"N"
preferred_address:	121
preferred_card:	121
con_id:	5

1:



También se puede hacer una petición con una herramienta de línea de comando como `curl` y para que el resultado, que es un documento JSON que encapsula los datos de la vista V_CUSTOMERS sea más legible, se imprime en pantalla con la utilidad `jq`:

```
curl http://localhost:8080/ords/soe/v_customers/ | jq
```

```
> curl http://localhost:8080/ords/soe/v_customers/ | jq
% Total    % Received % Xferd  Average Speed   Time     Time     Time  Current
                                         Dload  Upload   Total   Spent   Left  Speed
100 10729     0 10729     0      0  108k       0 --::-- --::-- --::-- 109k
{
  "items": [
    {
      "customer_id": 73,
      "cust_first_name": "john",
      "cust_last_name": "johnson",
      "nls_language": "sa",
      "nls_territory": "Georgia",
      "credit_limit": 5500,
      "cust_email": "john.johnson@aol.com",
      "account_mgr_id": 499,
      "customer_since": "2006-06-21T00:00:00Z",
      "customer_class": "Occasional",
      "suggestions": "Cooking",
      "dob": "1973-11-13T00:00:00Z",
      "mailshot": "Y",
      "partner_mailshot": "N",
      "preferred_address": 121,
      "preferred_card": 121,
      "con_id": 5
    },
  ]
}
```

¡Enhorabuena!, ha completado todas las actividades de esta jornada.

