UNIVERSIDAD DE GRANADA

DECSAI

Diseño y Desarrollo de Sistemas de Información

Grado en Ingeniería Informática

Seminario 1 – Acceso a bases de datos

©I. J. Blanco, F. J. Cabrerizo, C. Cruz, J.A. García, M. J. Martín, M.J. Rodríguez, D. Sánchez

Departamento de Ciencias de la Computación e Inteligencia Artificial
http://decsai.ugr.es

Contenidos

- Conexión a SGBD desde un lenguaje de programación de propósito general
  - Componentes
  - ODBC Driver Manager. Ejemplos
  - JDBC Driver Manager. Ejemplos
- Control de transacciones en Oracle
  - Concepto de transacción
  - Comienzo y final de una transacción
  - Ejemplo de uso
  - Savepoints
- Trabajo a realizar:
  - Elección de herramientas
  - Implementación
  - Entrega
  - Evaluación

Conexión a una BD desde un lenguaje de programación

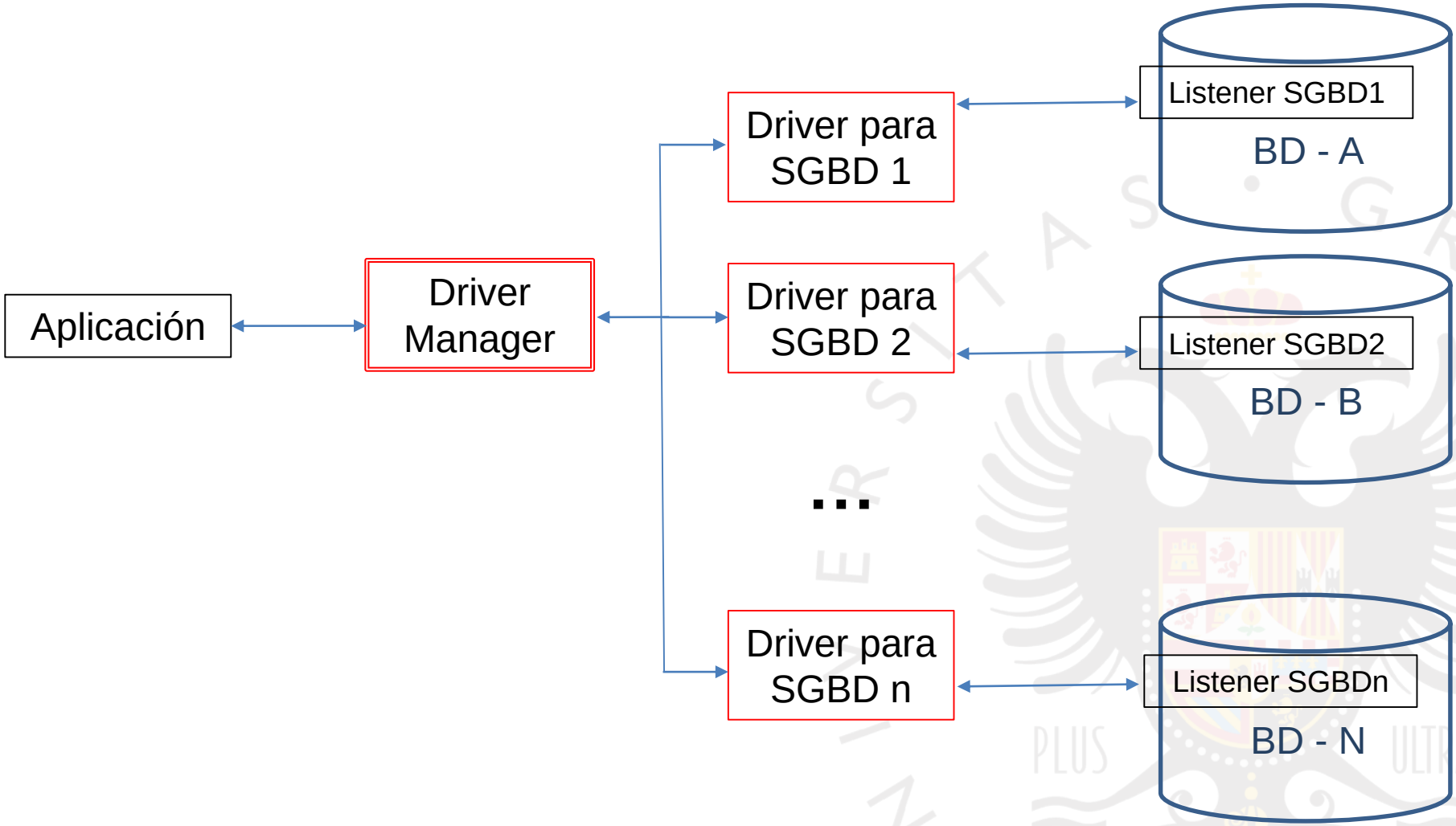
- Para poder gestionar la información almacenada en una base de datos desde un programa que estemos implementando necesitamos poder acceder desde éste a la Base de Datos.

- Para ello, los SGBD actuales proporcionan mecanismos para la recepción de sentencias de su lenguaje, así como la transmisión de resultados en forma de datos (ej. Tablas) y/o excepciones generadas por errores o por violación de restricciones.

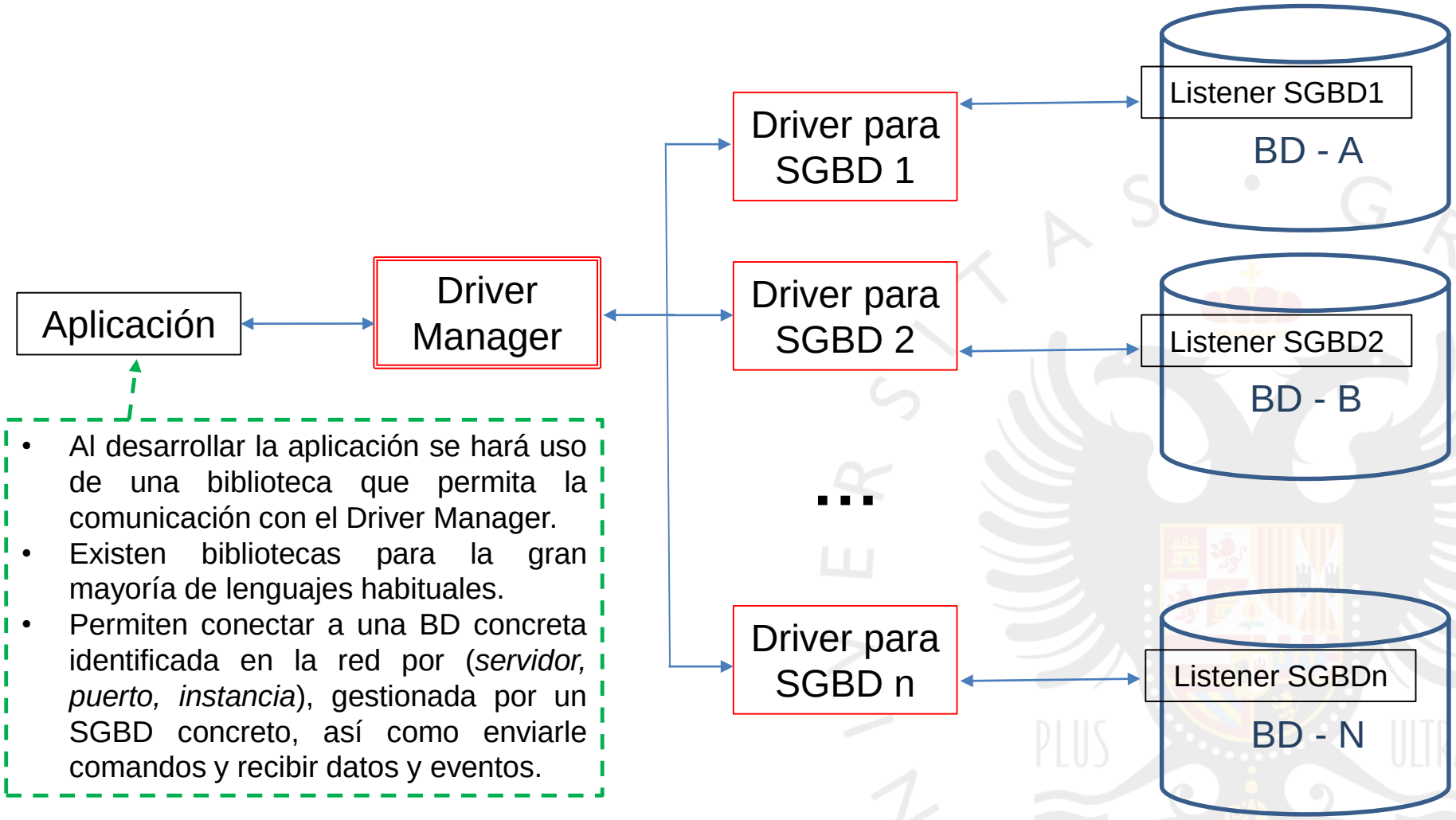
- Por su parte, los lenguajes de programación deben disponer de bibliotecas con procedimientos y funciones, o clases y métodos si son OO, para utilizar los mecanismos proporcionados por el SGBD.

- Por lo general, hoy en día es posible conectarse a cualquier SGBD desde cualquier lenguaje de programación de propósito general, así como desde lenguajes específicos para el desarrollo de SI.

- Una forma muy habitual de establecer la conexión desde un lenguaje de propósito general es usando estos componentes:

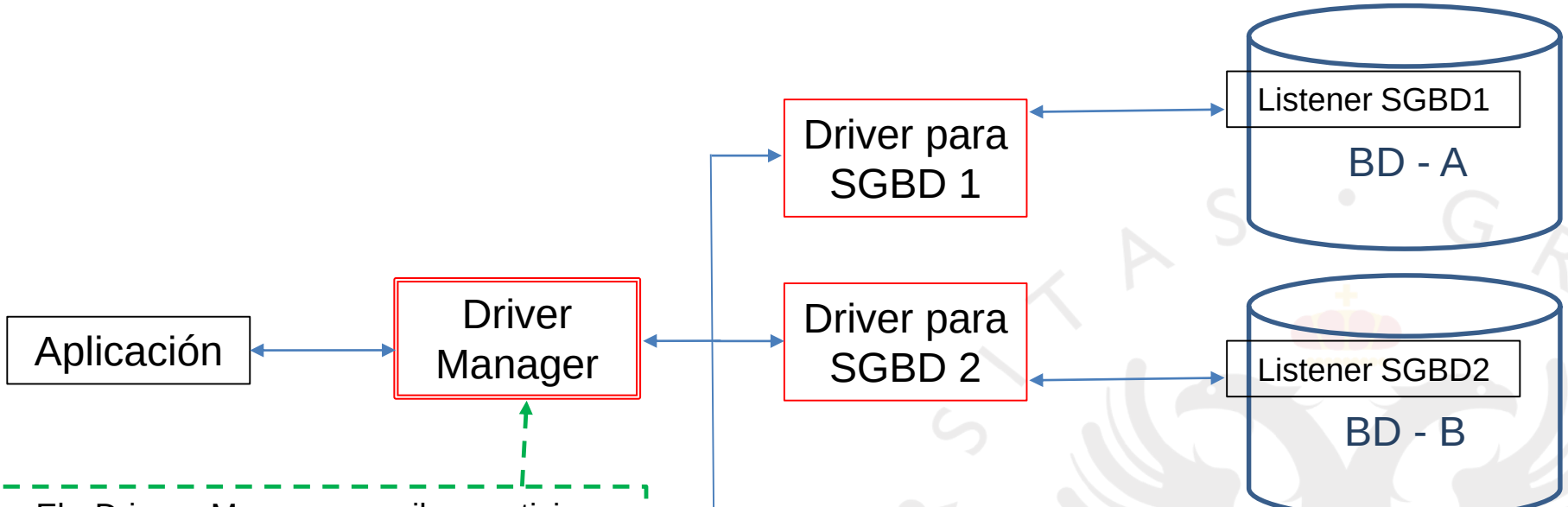


- Una forma muy habitual de establecer la conexión desde un lenguaje de propósito general es usando estos componentes:



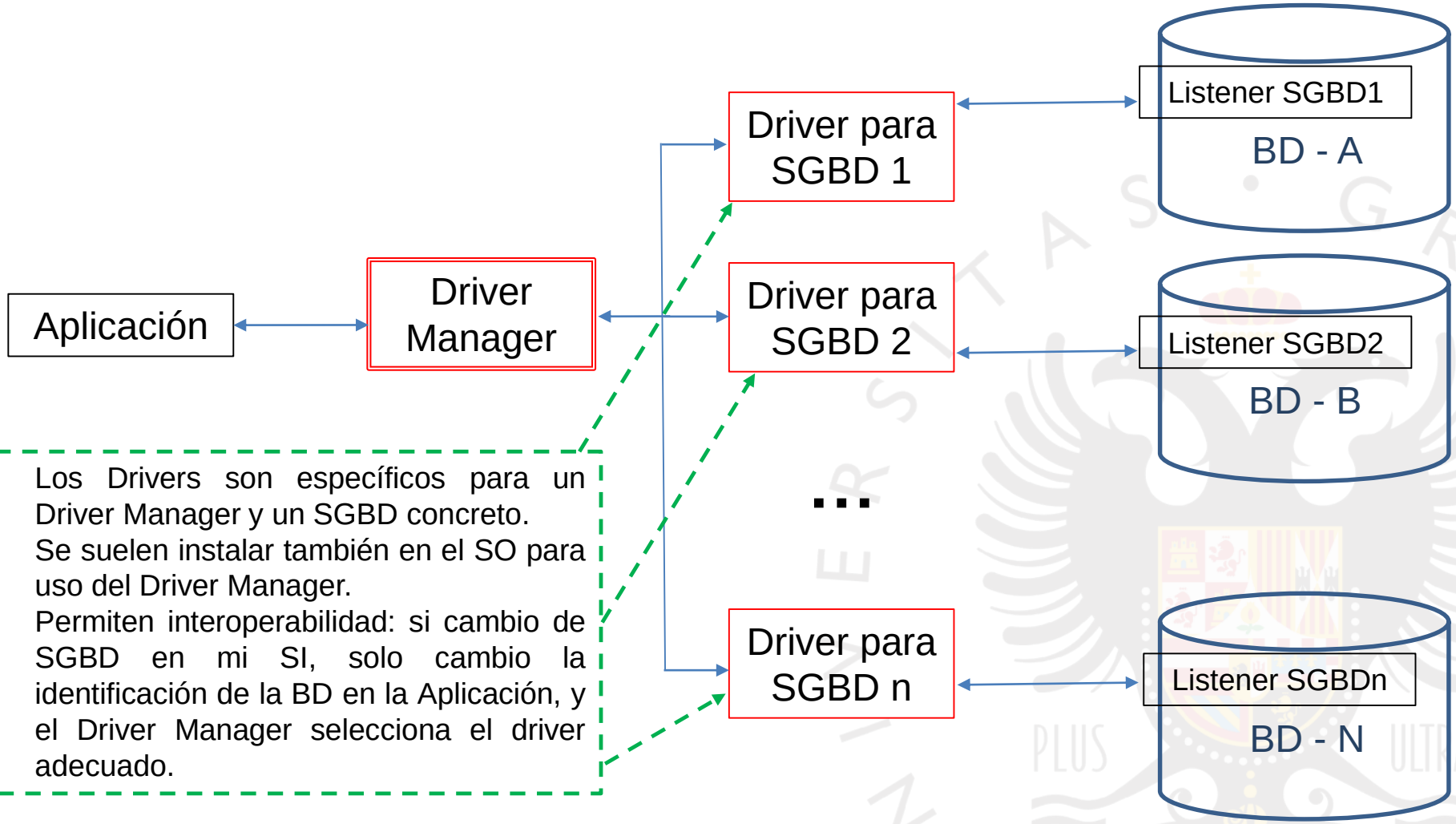
- Al desarrollar la aplicación se hará uso de una biblioteca que permita la comunicación con el Driver Manager.
- Existen bibliotecas para la gran mayoría de lenguajes habituales.
- Permiten conectar a una BD concreta identificada en la red por (*servidor, puerto, instancia*), gestionada por un SGBD concreto, así como enviarle comandos y recibir datos y eventos.

- Una forma muy habitual de establecer la conexión desde un lenguaje de propósito general es usando estos componentes:



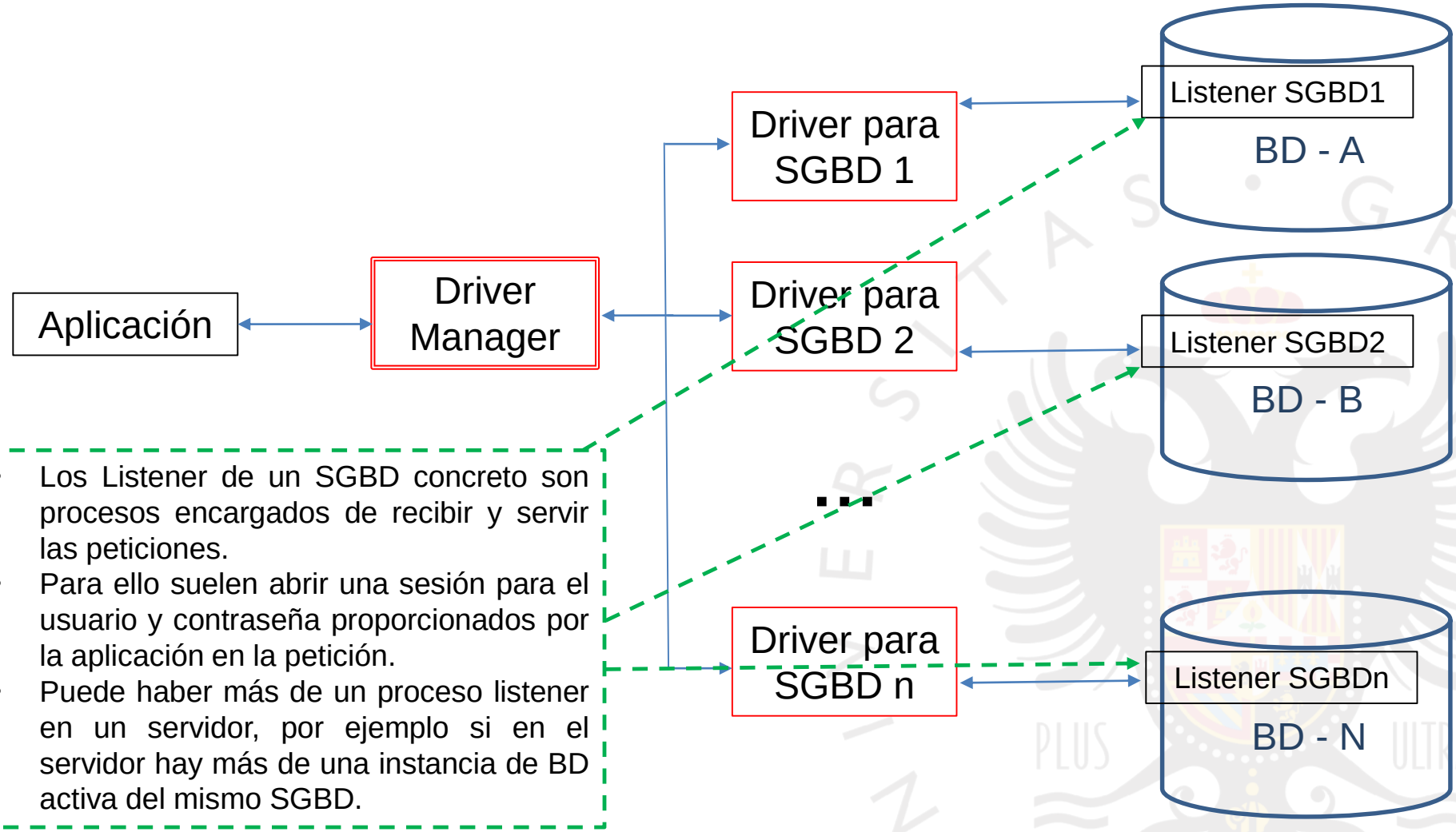
- El Driver Manager recibe peticiones desde la aplicación y las redirige a la BD indicada usando el driver propio del SGBD de dicha BD. Devuelve datos y excepciones.
- Entre los Driver Manager más usados para bases de datos relacionales están ODBC y JDBC. Se alojan en el SO.
- Los SGBD proporcionan drivers para éstos y otros Driver Managers.

- Una forma muy habitual de establecer la conexión desde un lenguaje de propósito general es usando estos componentes:



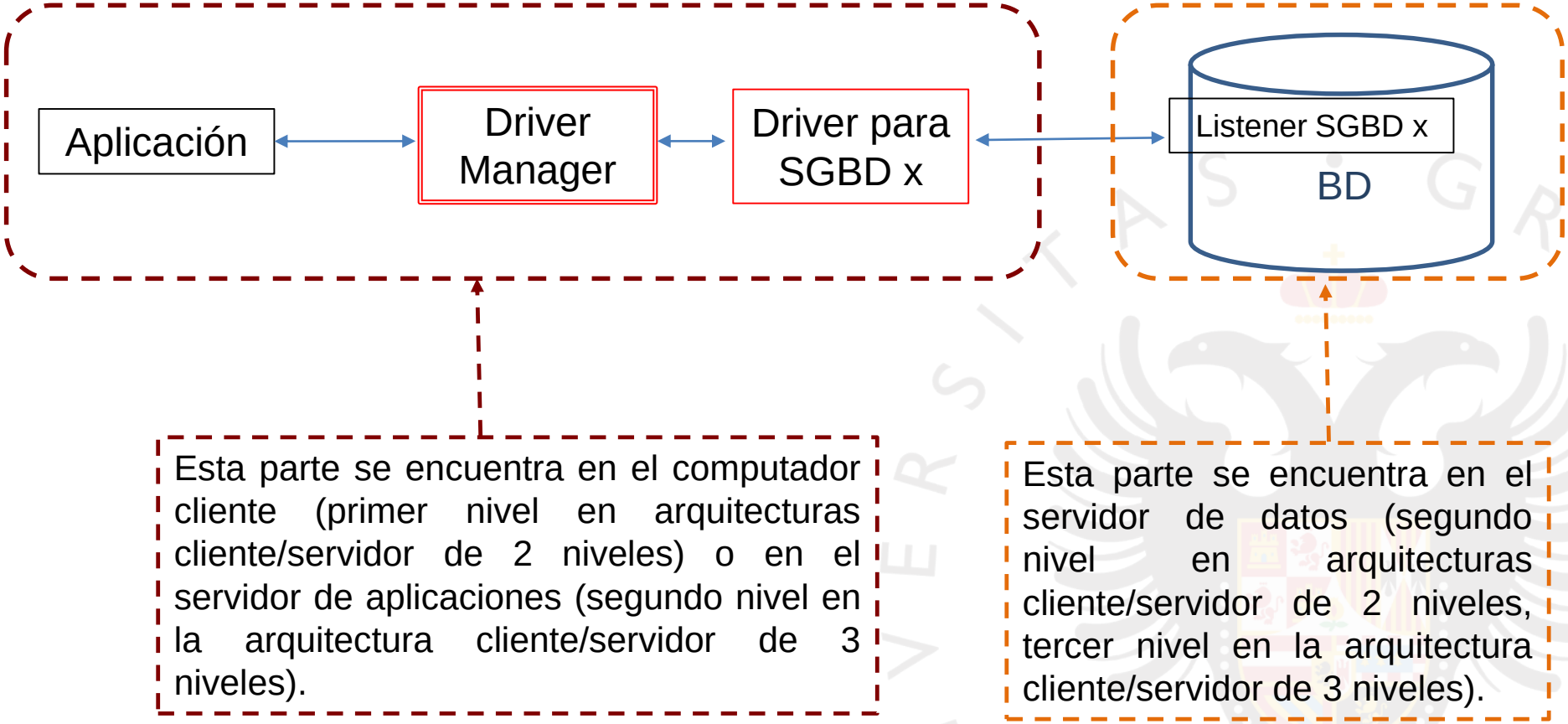
- Los Drivers son específicos para un Driver Manager y un SGBD concreto.
- Se suelen instalar también en el SO para uso del Driver Manager.
- Permiten interoperabilidad: si cambio de SGBD en mi SI, solo cambio la identificación de la BD en la Aplicación, y el Driver Manager selecciona el driver adecuado.

- Una forma muy habitual de establecer la conexión desde un lenguaje de propósito general es usando estos componentes:



- Los Listener de un SGBD concreto son procesos encargados de recibir y servir las peticiones.
- Para ello suelen abrir una sesión para el usuario y contraseña proporcionados por la aplicación en la petición.
- Puede haber más de un proceso listener en un servidor, por ejemplo si en el servidor hay más de una instancia de BD activa del mismo SGBD.

- Una forma muy habitual de establecer la conexión desde un lenguaje de propósito general es usando estos componentes:



- También puede estar todo centralizado en una misma máquina, p.e. vuestro portátil.

- ODBC (Open DataBase Connectivity) implementa una API con comandos para la comunicación con BD relacionales, así como con algunas otras fuentes de datos (ficheros Excel, etc.).

- Desarrollado por Microsoft e integrado en Windows. También existen versiones libres para otros SO:

  - Linux/UNIX: unixODBC
    https://www.easysoft.com/developer/interfaces/odbc/linux.html
  - Linux/UNIX y MacOS: iODBC
    http://www.iodbc.org/dataspace/doc/iodbc/wiki/iodbcWiki/WelcomeVisitors

- Existen drivers ODBC para prácticamente cualquier SGBD que admita conexiones basadas en arquitecturas cliente/servidor.

- Existen bibliotecas para usar ODBC desde prácticamente cualquier lenguaje de programación.

- C: mediante la biblioteca sql.h

https://www.easysoft.com/developer/languages/c/odbc_tutorial.html
https://docs.microsoft.com/en-us/sql/odbc/reference/develop-app/header-files?view=sql-server-ver15

- C++: mediante la biblioteca sqlapi.h

https://www.sqlapi.com/
https://www.sqlapi.com/Examples/step1.cpp

- Python: utilizando la biblioteca pyodbc

https://www.devart.com/odbc/oracle/docs/python.htm

- Ruby: utilizando los paquetes dbi, dbd-odbc, y ruby-odbc

https://www.cdata.com/kb/tech/oracledb-odbc-ruby.rst

- Podéis encontrar muchos más ejemplos de bibliotecas y código en la Web para éstos y otros lenguajes, así como distintos SO y SGBD .

- JDBC (Java DataBase Connectivity) es una API específica del lenguaje Java para la comunicación con BD relacionales y otras fuentes de datos en formato tabular.

- Independiente de la plataforma, utilizable en Windows, Linux/UNIX, MacOS …

- Proporciona conexión tanto mediante los componentes que hemos visto (type 4 Driver en JDBC) como mediante otras arquitecturas de componentes (type 1, 2 y 3).

  https://en.wikipedia.org/wiki/JDBC_driver

- Las bibliotecas java.sql y javax.sql proporcionan la API mediante una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos.

  https://docs.oracle.com/javase/8/docs/api/java/sql/package-summary.html

- https://mkyong.com/jdbc/connect-to-oracle-db-via-jdbc-driver-java/
- https://www.javatpoint.com/example-to-connect-to-the-oracle-database
- https://gist.github.com/jujogracu/3063672
- Podéis encontrar muchos más ejemplos en la Web.

- Una transacción es una unidad lógica atómica de procesamiento que contiene una o más sentencias SQL.

- El que sea atómica significa que el SGBD debe garantizar que o se aplican a los datos todas las sentencias SQL que conforman la transacción, o ninguna.

- Ejemplo típico: transacción bancaria. Supongamos que tenemos las siguientes tablas que guardan el saldo actual y los movimientos realizados sobre cuentas bancarias:

Cuenta	Saldo
1	100
2	80

Cuenta	Id-Movimiento	Cantidad

- Para pasar una cantidad X de la cuenta 1 a la 2 hay que ejecutar cuatro sentencias de SQL que forman una única transacción:
  - Restar X al saldo de la cuenta 1 (update)
  - Sumar X al saldo de la cuenta 2 (update)
  - Reflejar el movimiento en cuenta 1 (insert)
  - Reflejar el movimiento en cuenta 2 (insert)
- Deben aplicarse todas o ninguna.

Cuenta	Saldo
1	100
2	80

Cuenta	Id-Movimiento	Cantidad

- Restar X al saldo de la cuenta 1 (update)

Cuenta	Saldo
1	100-X
2	80

Cuenta	Id-Movimiento	Cantidad

- No se ha completado la transacción: la cuenta 1 ha perdido X unidades que deben ir a alguna parte. La BD está inconsistente. Además, el movimiento no se ha reflejado en la tabla de movimientos.

- Restar X al saldo de la cuenta 1 (update)

Cuenta	Saldo
1	100-X
2	80

Cuenta	Id-Movimiento	Cantidad

- Sumar X al saldo de la cuenta 2 (update)

Cuenta	Saldo
1	100-X
2	80+X

Cuenta	Id-Movimiento	Cantidad

- No se ha completado la transacción: la BD está consistente en cuanto a los saldos, pero falta la información de los movimientos, que no se han reflejado todavía.

- Restar X al saldo de la cuenta 1 (update)

Cuenta	Saldo
1	100-X
2	80

Cuenta	Id-Movimiento	Cantidad

- Sumar X al saldo de la cuenta 2

Cuenta	Saldo
1	100-X
2	80+X

- No se ha completado la transacción: la BD está consistente en cuanto a los saldos, pero falta la información del segundo movimiento, que no se ha reflejado todavía.

- Reflejar el movimiento en cuenta 1 (insert)

Cuenta	Saldo
1	100-X
2	80+X

Cuenta	Id-Movimiento	Cantidad
1	M1	-X

- Restar X al saldo de la cuenta 1 (update)

Cuenta	Saldo
1	100-X
2	80

Cuenta	Id-Movimiento	Cantidad

- Sumar X al saldo de la cuenta 2

Cuenta	Saldo
1	100-X
2	80+X

- Reflejar el movimiento en cuenta 1 (insert)

Cuenta	Saldo
1	100-X
2	80+X

Cuenta	Id-Movimiento	Cantidad
1	M1	-X

- Reflejar el movimiento en cuenta 2 (insert)

Cuenta	Saldo
1	100-X
2	80+X

Cuenta	Id-Movimiento	Cantidad
1	M1	-X
2	M2	+X

- Se ha completado la transacción: la BD está consistente en cuanto a los saldos, y todos los movimientos han sido reflejados. La operación lógica “Transacción de X desde cuenta 1 a cuenta 2” se ha terminado

- Una transacción comienza al ejecutar cualquier operación DML (insert, update, delete). El comienzo de una transacción produce el bloqueo de la tabla afectada hasta que termine la transacción.

- Una transacción termina, causando el desbloqueo de la tabla, cuando:

  - se hace una llamada a los comandos COMMIT o ROLLBACK, o
  - se ejecuta una sentencia DDL, o
  - el usuario se desconecta, o
  - el proceso actual termina de forma anormal.

- Nota: En ocasiones, el Driver Manager está configurado para realizar un commit automático (autocommit) después de cada operación del DML desde el lenguaje de programación. En esos casos, para poder realizar nosotros el control de transacciones desde la aplicación mediante Commit y Rollback, tenemos que desactivar esa opción previamente en nuestro código de aplicación.

- El comando COMMIT indica el final de la transacción y hace que los cambios en los datos se hagan "permanentes", en el sentido de que dichos cambios serán visibles en las consultas de otros usuarios.

- Antes de aplicar COMMIT los cambios son visibles para nosotros al hacer un SELECT, pero no para el resto de usuarios.

- El comando ROLLBACK deshace todas las operaciones realizadas desde el comienzo de la transacción activa actual.

- Cuando se aplica el comando COMMIT ya no se puede aplicar el comando ROLLBACK sobre la transacción, ya que ésta deja de estar activa.

Ejemplo de uso: transacción bancaria.

- Comienza la transacción con el primer update, que resta X a la cuenta 1.
- Se hace el segundo update, que suma X a la cuenta 2. Aquí se ve la importancia del bloqueo de tablas (otro usuario podría intentar a la vez transferir el dinero de la cuenta 2 a otra, sin que se haya terminado la transacción que asegura que ese saldo es correcto, esto hay que evitarlo).
  - Si nuestra aplicación en este punto da opción de cancelar la transacción, y el usuario la ejecuta, podemos simplemente ordenar un ROLLBACK, que deshará los dos update anteriores y desbloqueará las tablas.
  - En cualquier punto de la transacción, si se produce un error (por ejemplo, un corte de red o un apagón en el cliente), Oracle hace un ROLLBACK automático de las operaciones realizadas.
- Si todo ha ido bien y se han hecho las 4 operaciones, hacemos un COMMIT, con lo que la transacción se hace permanente y se desbloquean la tabla de Saldos y la de Movimientos.

- Pueden establecerse puntos intermedios dentro de una transacción de forma que el comando ROLLBACK pueda eliminar todas las operaciones realizadas después de un punto concreto. Esto se realiza mediante el comando SAVEPOINT.

SAVEPOINT <nombre>

- Siempre que no haya terminado la transacción, podemos hacer un ROLLBACK hasta ese punto con el comando

ROLLBACK TO <nombre>

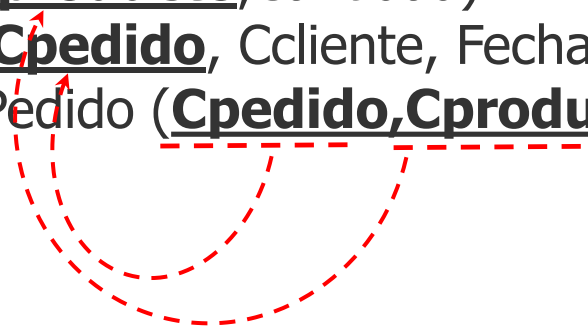
- En el ejemplo de la transferencia bancaria, sería útil si después del segundo update, nuestra aplicación permite al usuario cancelar la transferencia y cambiar la cuenta de destino. Basta con poner un Savepoint tras el primer update, y hacer un Rollback a ese savepoint antes de las dos inserciones en Movimientos.

- OJO: al hacer Rollback a un Savepoint, la transacción sigue activa, pero en un punto anterior, marcado por el Savepoint, al que estaba.

- ¿Se puede hacer todo esto bien sin control de transacciones?
- Enlaces a ejemplos de uso y más información:
  - https://docs.oracle.com/en/database/oracle/oracle-database/12.2/tdddg/dml-and-transactions.html#GUID-C9CA9BBB-DECD-4935-A790-195424E08C6A
  - https://docs.oracle.com/database/121/CNCPT/transact.htm#CNCPT038
  - https://docs.oracle.com/javase/tutorial/jdbc/basics/transactions.html
  - https://docs.oracle.com/cd/E15357_01/coh.360/e15726/cpp_tx.htm#COHCG5142
  - https://www.oracletutorial.com/python-oracle/transactions/
  - https://developer.oracle.com/dsl/marx-ruby.html
- Y mucho más que podéis encontrar en la Web.

- El trabajo se realizará por subgrupos (los mismos de prácticas).
- Cada subgrupo tendrá que elegir un lenguaje de programación y un SGBD relacional que permita la gestión de transacciones.
- Como lenguaje de programación se elegirá un lenguaje de propósito general y uso común en la actualidad entre Java, C, C++, Python o Ruby. Se puede usar algún otro lenguaje que disponga de librerías de conexión a SGBD relacionales, previa consulta y aceptación por parte del profesor.
- Como SGBD se recomienda usar Oracle (particularmente el servidor de la ETSIIT) del que hemos visto aquí cómo gestiona transacciones. Podéis usar otros SGBD relacionales que gestionen transacciones, previa consulta y aceptación por parte del profesor.
- Se permite que varios grupos elijan el mismo lenguaje y/o SGBD.

- Se debe implementar un SI sencillo que trabaje con las siguientes tablas (todos los dominios son numéricos excepto la Fecha-pedido):
  - Stock (**Cproducto**,Cantidad)
  - Pedido (**Cpedido**, Ccliente, Fecha-pedido)
  - Detalle-Pedido (**Cpedido,Cproducto**,Cantidad)



- En la implementación de la aplicación se pueden utilizar partes de código que se encuentre en cualquier fuente, referenciándola expresamente, excepto código que se haya desarrollado en años anteriores expresamente para esta asignatura. Lo importante es que se sepa explicar detalladamente el código y su funcionamiento.

- La aplicación debe realizar:
  - Conexión a BD mediante ODBC ó JDBC.
  - Mostrar un interfaz muy sencillo con un menú principal que permita las siguientes opciones:
    - Borrado y nueva creación de las tablas e inserción de 10 tuplas predefinidas en el código en la tabla Stock.
    - Dar de alta nuevo pedido
    - Mostrar contenido de las tablas de la BD
    - Salir del programa y cerrar conexión a BD

- El programa deberá obligatoriamente hacer uso de control de transacciones para el alta de nuevos pedidos, comenzando las mismas, estableciendo Savepoints donde convenga y haciendo uso adecuado de los comandos Rollback y Commit.

- Dentro de la opción de Dar de alta nuevo pedido:
  - El interfaz capturará los datos básicos del pedido, que se insertarán en la tabla Pedido con un INSERT.
  - A continuación ofrecerá como opciones "1. Añadir detalle de producto", "2. Eliminar todos los detalles de producto", "3. Cancelar pedido" y "4. Finalizar pedido".
  - La opción 1 debe capturar los datos de un artículo y cantidad, y realizar la inserción correspondiente en la tabla Detalle-Pedido, si hay stock, así como actualizar el stock, quedando en este mismo menú.
  - La opción 2 debe eliminar todos los detalles de pedido que se han insertado en Detalle-Pedido para el pedido actual (pero no el pedido en la tabla Pedidos) y quedar en este mismo menú.
  - La opción 3 debe eliminar el pedido y todos sus detalles y volver al menú principal de la aplicación.
  - La opción 4 debe hacer los cambios permanentes y volver al menú principal de la aplicación.

- Después de las opciones 1, 2 y 3 en este menú, siempre hay que mostrar el contenido de la BD.

- Se entregará a través de Prado un fichero .zip que contendrá:
  - Un documento .pdf donde se indicará muy brevemente:
    - El lenguaje utilizado.
    - Las tareas de instalación realizadas en relación tanto al lenguaje como a los paquetes necesarios para la conexión.
    - En caso de usar partes de código implementado por otros, las fuentes de donde se ha obtenido.
    - El reparto de trabajo en estos aspectos y en la implementación entre las personas del grupo (que en todo caso tendrán que conocer y estudiar el trabajo realizado por el resto de personas del grupo, pudiéndoseles preguntar por ello en la defensa).
  - El código fuente de la aplicación implementada.

- Evaluación:
  - Se recuerda que el trabajo a realizar en este Seminario forma parte de la evaluación de la asignatura, ver Guía Docente.
  - La calificación final se obtendrá como resultado de:
    - La evaluación por el profesor de la entrega, y
    - La defensa grupal del trabajo, donde habrá que responder a preguntas del profesor sobre el código y hacer una demostración de uso de la aplicación implementada, mostrando el adecuado funcionamiento a través de un cliente de consulta a la base de datos (por ejemplo, SQLDeveloper en el caso de haber usado Oracle).