Ejercicio 1

Pregunta 1:

Primero creo la base datos bdfactura:

```
walterrg@Lenovo:~/Documents/CC-3S2$ sudo -i -u postgres
[sudo] password for walterrg:
postgres@Lenovo:~$ psql
psql (14.8 (Ubuntu 14.8-0ubuntu0.22.04.1))
Type "help" for help.

postgres=# create database bdfactura
postgres-# ;
CREATE DATABASE
postgres=#
```

Se crea la tabla factura:

```
bdfactura=# create table factura(
bdfactura(# nombre varchar(100),
bdfactura(# valor float8);
CREATE TABLE
bdfactura=#
```

Creo la clase FacturaDao, esta tiene el método conectar, el cual permite conectarse con la base de datos creada bdfactura y retorna la conexión:

```
public class FacturaDao {
  private final String url = "jdbc:postgresql://localhost/bdfactura";
  private final String user = "postgres";
  private final String password = "1234";

  public Connection conectar() {
    Connection conn = null;
    try {
      conn = DriverManager.getConnection(url, user, password);
      System.out.println("Conexión existosa");
    } catch (SQLException e) {
      System.out.println(e.getMessage());
    }
    return conn;
  }
```

Creo el método guardar para insertar valores en la tabla factura:

```
public void guardar(String cliente, int valor) {
  Connection conn = conectar();
  try {
    Statement stmt = conn.createStatement();
    String consultaInsert =
            String.format("INSERT INTO factura (nombre, valor) VALUES ('%s',
%d)", cliente, valor);
    stmt.executeUpdate(consultaInsert);
    stmt.close();
    conn.close();
  } catch (SQLException e) {
    throw new RuntimeException(e);
  }
}
```

Luego creo el método todo que retorna una lista de todas las facturas:

```
public List<Factura> todo() {
  List<Factura> facturas = new ArrayList<>();
  Connection conn = conectar();
  try {
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT * FROM factura;");
    while (rs.next()) {
      String nombre = rs.getString("nombre");
      int valor = rs.getInt("valor");
      facturas.add(new Factura(nombre, valor));
    }

    rs.close();
    stmt.close();
    conn.close();
  } catch (SQLException e) {
    throw new RuntimeException(e);
  }
  return facturas;
}
```

Finalmente creo el método todosConAlMenos que retorna una lista de facturas con un valor mayor o igual que el valor del argumento pasado al método:

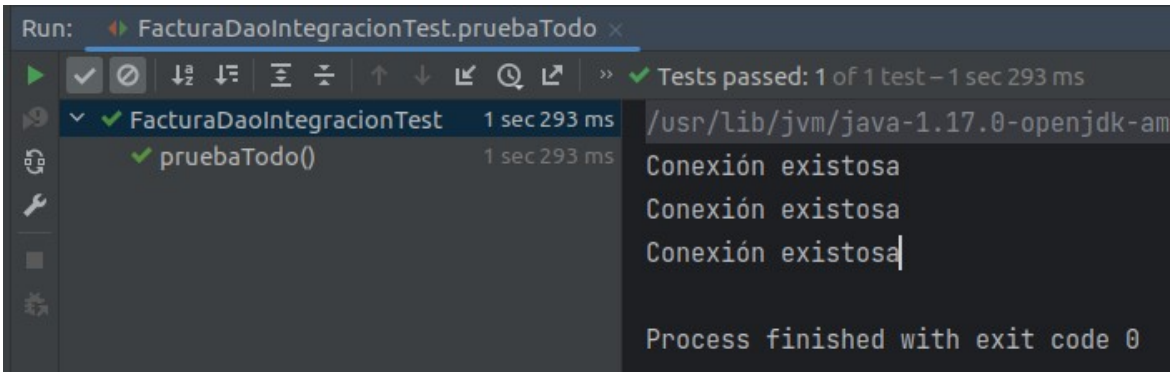
```
public List<Factura> todosConAlMenos(int minVal) {
  List<Factura> facturas = new ArrayList<>();
  Connection conn = conectar();
  try {
    Statement stmt = conn.createStatement();
    ResultSet rs =
        stmt.executeQuery(String.format("SELECT * FROM factura WHERE valor
>= %d;", minVal));
    while (rs.next()) {
      String nombre = rs.getString("nombre");
      int valor = rs.getInt("valor");
      facturas.add(new Factura(nombre, valor));
    }

    rs.close();
    stmt.close();
    conn.close();
  } catch (SQLException e) {
    throw new RuntimeException(e);
  }
  return facturas;
}
```

Pregunta 2

Creo la clase FacturaDaoIntegracionTest para realizar las pruebas.
Primero creo la prueba pruebaTodo, la cual crea un objeto FacturaDao, utiliza este objeto para insertar dos filas en la tabla factura usando el método guardar y luego ejecuta el método todo para obtener una lista con las facturas de la tabla.
Finalmente compara las facturas devueltas con las facturas ingresadas:

```
public class FacturaDaoIntegracionTest {
    @Test
    public void pruebaTodo() {
        FacturaDao dao = new FacturaDao();
        dao.guardar("Luis",1000);
        dao.guardar("Carlos",2000);
        List<Factura> facturas = dao.todo();
        assertThat(facturas.get(0)).isEqualTo(new Factura("Luis", 1000));
        assertThat(facturas.get(1)).isEqualTo(new Factura("Carlos", 2000));
    }
}
```

Ejecutando la pruebase obtiene que pasa la prueba:



Pregunta 3

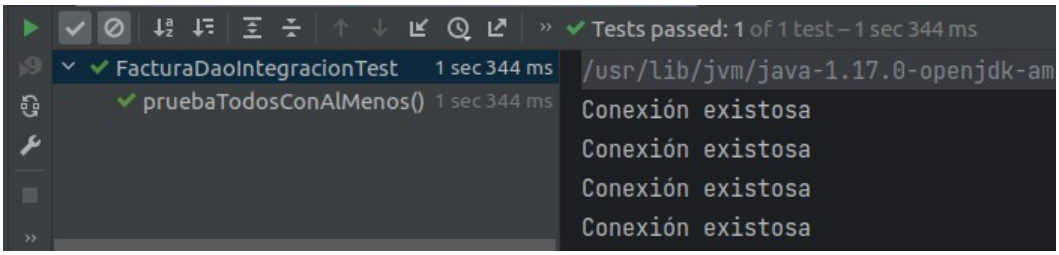
Añado los métodos openConnectionAndCleanup paraque se ejecute antes de cada prueba. Este método crea una conexión con la base de datos y borra todas las filas de la tabla factura. También creo el método closeConnection, el cual se ejecuta luego de cada prueba y cierra la conexión:

```
public class FacturaDaoIntegracionTest {
  FacturaDao dao;
  Connection conn;

  @BeforeEach
  public void openConnectionAndCleanup() {
    dao = new FacturaDao();
    conn = dao.conectar();
    try {
      Statement stmt = conn.createStatement();
      stmt.executeUpdate("DELETE FROM factura;");
      stmt.close();
    } catch (SQLException e) {
      throw new RuntimeException(e);
    }
  }

  @AfterEach
  public void closeConnection() {
    try {
      conn.close();
    } catch (SQLException e) {
      throw new RuntimeException(e);
    }
  }
```

También creo la prueba pruebaTodosConAlMenos,la cual crea 3 filas con valores 49, 50 y 51. Luego llamo al método todosConAlMenos y le paso el valor 50 para que me retorne una lista con las facturas con valor mayor o igual a 50. Finalmente comprueba que la lista retornada solo contenga las facturas con valor mayor o igual a 50.

```
@Test
public void pruebaTodosConAlMenos() {
  dao.guardar("Luis", 49);
  dao.guardar("Carlos", 50);
  dao.guardar("Juan", 51);
  List<Factura> facturas = dao.todosConAlMenos(50);
  assertThat(facturas).doesNotContain(new Factura("Luis", 49));
  assertThat(facturas).contains(new Factura("Carlos", 50));
  assertThat(facturas).contains(new Factura("Juan", 51));
}
```

Ejecutando se obtiene que las pruebas pasan:



Pregunta 4

Creo la clase SQLIntegrationTestBase, la cual declara facturaDao con acceso protegido e implementa los métodos de limpieza de la tabla factura (openConnectionAndCleanup) y el cierre de la conexion (closeConnection) que se ejecutan antes y despueś de cada prueba respectivamente.

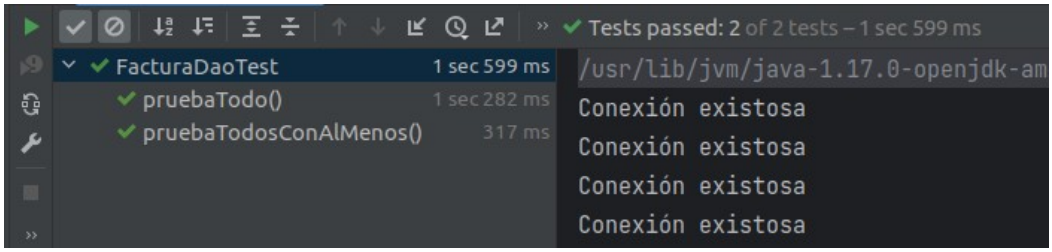
```
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;

public class SQLIntegrationTestBase {
    protected FacturaDao facturaDao;
    Connection conn;
    @BeforeEach
    public void openConnectionAndCleanup() {
        facturaDao = new FacturaDao();
        conn = facturaDao.conectar();
        try {
            Statement stmt = conn.createStatement();
            stmt.executeUpdate("DELETE FROM factura;");
            stmt.close();
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }

    @AfterEach
    public void closeConnection() {
        try {
            conn.close();
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }
}
```

Luego creo la clase FacturaDaoTest, la cual extiende la clase SQLIntegrationTestBase y por lo tanto puede acceder al objeto facturaDao de la clase padre. Este clase se encarga de probar los métodos todo y todosConAlMenos de la clase FacturaDao, los métodos de limpieza de la tabla y cierre de la conexión los hereda de la clase SQLIntegrationTestBase.

```
import static org.assertj.core.api.Assertions.*;

import java.util.List;
import org.junit.jupiter.api.Test;

public class FacturaDaoTest extends SQLIntegrationTestBase {
  @Test
  public void pruebaTodo() {
    facturaDao.guardar("Luis", 1000);
    facturaDao.guardar("Carlos", 2000);
    List<Factura> facturas = facturaDao.todo();
    assertThat(facturas.get(0)).isEqualTo(new Factura("Luis", 1000));
    assertThat(facturas.get(1)).isEqualTo(new Factura("Carlos", 2000));
```

```
  }

  @Test
  public void pruebaTodosConAlMenos() {
    facturaDao.guardar("Luis", 49);
    facturaDao.guardar("Carlos", 50);
    facturaDao.guardar("Juan", 51);
    List<Factura> facturas = facturaDao.todosConAlMenos(50);
    assertThat(facturas).doesNotContain(new Factura("Luis", 49));
    assertThat(facturas).contains(new Factura("Carlos", 50));
    assertThat(facturas).contains(new Factura("Juan", 51));
  }
}
```

Ejecutando las pruebas en esta clase se observa que las pruebas pasan, por lo que el funcionamiento de las pruebas no se ha alterado al utilizar separar la funcionalidad de la clase anterior en dos clases:



Pregunta 5

Se creó el método conectar en la clase FacturaDao, el cual realiza la apertura y confirmación de la conexión:

```
public Connection conectar() {
  Connection conn = null;
  try {
    conn = DriverManager.getConnection(url, user, password);
    System.out.println("Conexión existosa");
  } catch (SQLException e) {
    System.out.println(e.getMessage());
  }
  return conn;
}
```

Adicionalmente, se creó el método eliminarDatos en la clase FacturaDao, el cual realiza la limpieza de la tabla.

```
public void eliminarDatos() {
  Connection conn = conectar();
  conn = conectar();
  try {
    Statement stmt = conn.createStatement();
    stmt.executeUpdate("DELETE FROM factura;");
    stmt.close();
  } catch (SQLException e) {
    throw new RuntimeException(e);
  }
}
```

Usando este método se simplifica el método openConnectionAndCleanup:

```
@BeforeEach
public void openConnectionAndCleanup() {
    facturaDao = new FacturaDao();
    conn = facturaDao.conectar();
    facturaDao.eliminarDatos();
}
```

Ejercicio 2

Método 1: Usando un Dockerfile

Primero se crea el Dockerfile:

```
FROM alpine:latest
LABEL maintainer="Cesar Lara Avila<checha@claraa.io>"
LABEL description="Este Dockerfile de ejemplo instala NGINX"
RUN apk add --update nginx && \
rm -rf /var/cache/apk/* && \
mkdir -p /tmp/nginx/
COPY files/nginx.conf /etc/nginx/nginx.conf
COPY files/default.conf /etc/nginx/conf.d/default.conf ADD
files/html.tar.gz /usr/share/nginx/
EXPOSE 80/tcp
ENTRYPOINT ["nginx"]
CMD ["-g", "daemon off;"]
```

En este Dockerfile se usa como imagen base alpine para la creación del contenedor.
Se crea la carpeta files y dentro se coloca los archivs nginx.conf,default.conf y html.tar.gz.

Usando este dockerfile se crea la imagen dockerfile-example:

```
walterrg@Lenovo:~/Documents/CC-3S2/PracticaCalificada5/Pregunta2$ docker build -t dockerfile-example .
[+] Building 30.9s (10/10) FINISHED
 => [internal] load build definition from Dockerfile                       0.1s
 => => transferring dockerfile: 455B                                       0.0s
 => [internal] load .dockerignore                                          0.0s
 => => transferring context: 2B                                            0.0s
 => [internal] load metadata for docker.io/library/alpine:latest           1.4s
 => [1/5] FROM docker.io/library/alpine:latest@sha256:82d1e9d7ed48a7523bd  8.0s
 => => resolve docker.io/library/alpine:latest@sha256:82d1e9d7ed48a7523bd  0.1s
 => => sha256:82d1e9d7ed48a7523bdebc18cf6290bdb97b82302a8 1.64kB / 1.64kB  0.0s
 => => sha256:25fad2a32ad1f6f510e528448ae1ec69a28ef81916a004d 528B / 528B  0.0s
 => => sha256:c1aabb73d2339c5ebaa3681de2e9d9c18d57485045a 1.47kB / 1.47kB  0.0s
 => => sha256:31e352740f534f9ad170f75378a84fe453d6156e407 3.40MB / 3.40MB  7.6s
 => => extracting sha256:31e352740f534f9ad170f75378a84fe453d6156e40700b88  0.1s
 => [internal] load build context                                          0.0s
 => => transferring context: 59.40kB                                       0.0s
 => [2/5] RUN apk add --update nginx && rm -rf /var/cache/apk/* && mkdir  20.6s
 => [3/5] COPY files/nginx.conf /etc/nginx/nginx.conf                      0.2s
 => [4/5] COPY files/default.conf /etc/nginx/conf.d/default.conf           0.1s
 => [5/5] ADD files/html.tar.gz /usr/share/nginx/                          0.1s
 => exporting to image                                                     0.2s
 => => exporting layers                                                    0.2s
 => => writing image sha256:9e46f923f3c8b8980b3807b241bf0b2cd56b3a72de7d6  0.0s
 => => naming to docker.io/library/dockerfile-example                      0.0s
walterrg@Lenovo:~/Documents/CC-3S2/PracticaCalificada5/Pregunta2$
```

Se verifica que la imagen se haya creado:

```
walterrg@Lenovo:~/Documents/CC-3S2/PracticaCalificada5/Pregunta2$ docker images
REPOSITORY           TAG       IMAGE ID       CREATED         SIZE
dockerfile-example   latest    9e46f923f3c8   2 minutes ago   8.95MB
```

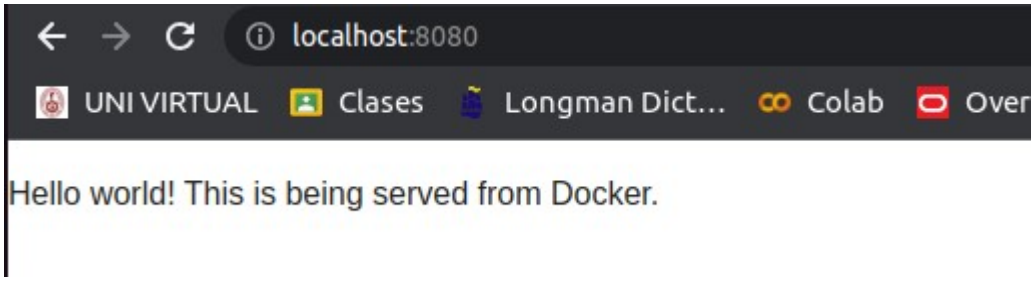
Se crea un contenedor de nombre dockerfile-example a partir de la imagen creada en modo dettach para que se ejecute en segundo plano y no se cierre, además se expone el puerto 80 en el puerto 8080 del host de docker.

```
walterrg@Lenovo:~/Documents/CC-3S2/PracticaCalificada5/Pregunta2$ docker run -d --name dockerfile-example -p 8080:80 dockerfile
-example
003d9058957f84476c73eceb83d8903a2b5504634717e99811615f62da8dd439
walterrg@Lenovo:~/Documents/CC-3S2/PracticaCalificada5/Pregunta2$
```

Se comprueba que el contenedor se haya creado y se encuentre en ejecución:

```
walterrg@Lenovo:~/Documents/CC-3S2/PracticaCalificada5/Pregunta2$ docker container ls
CONTAINER ID   IMAGE               COMMAND                  CREATED             STATUS             PORTS
               NAMES
003d9058957f   dockerfile-example   "nginx -g 'daemon of…"   About a minute ago   Up About a minute   0.0.0.0:8080->80/tcp, :::
8080->80/tcp   dockerfile-example
walterrg@Lenovo:~/Documents/CC-3S2/PracticaCalificada5/Pregunta2$
```

Abriendo el navegador y llendo a localhost:8080 se obtiene:



Se detiene el contenedor:

```
walterrg@Lenovo:~/Documents/CC-3S2/PracticaCalificada5/Pregunta2$ docker container stop dockerfile-example
dockerfile-example
```

Método 2

Primero se descarga la imagen de alpine desde docker-hub:

```
walterrg@Lenovo:~/Documents/CC-3S2/PracticaCalificada5/Pregunta2$ docker image pull alpine:latest
latest: Pulling from library/alpine
31e352740f53: Pull complete
Digest: sha256:82d1e9d7ed48a7523bdebc18cf6290bdb97b82302a8a9c27d4fe885949ea94d1
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
walterrg@Lenovo:~/Documents/CC-3S2/PracticaCalificada5/Pregunta2$
```

Luego se ejecuta el contenedor y al ingresar a este se agrega el paquete nginx:

```
walterrg@Lenovo:~/Documents/CC-3S2/PracticaCalificada5/Pregunta
2$ docker container run -it --name alpine-test alpine /bin/sh
/ # apk update
fetch https://dl-cdn.alpinelinux.org/alpine/v3.18/main/x86_64/A
PKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.18/community/x86
_64/APKINDEX.tar.gz
v3.18.2-275-g694c92e73de [https://dl-cdn.alpinelinux.org/alpine
/v3.18/main]
v3.18.2-274-ga199b869491 [https://dl-cdn.alpinelinux.org/alpine
/v3.18/community]
OK: 20062 distinct packages available
/ # apk upgrade
OK: 7 MiB in 15 packages
/ # apk add --update nginx
(1/2) Installing pcre (8.45-r3)
(2/2) Installing nginx (1.24.0-r6)
Executing nginx-1.24.0-r6.pre-install
Executing nginx-1.24.0-r6.post-install
Executing busybox-1.36.1-r0.trigger
OK: 9 MiB in 17 packages
/ # rm -rf /var/cache/apk/*
/ # mkdir -p /tmp/nginx/
/ # exit
walterrg@Lenovo:~/Documents/CC-3S2/PracticaCalificada5/Pregunta
```