Base de Dados

É muito usual a utilização de um SGBD para armazenamento de dados, por serem sistemas ideais para o suporte persistente da informação.
Em vez de escrever dados num ficheiro, a sua aplicação distribuída pode beneficiar dos serviços de um **SGBD** como o Postgres ou MySQL.

Apontamentos sobre a criação de uma BD típica

Inicialização de uma BD em MySQL

0- instalação so SDGB em Ubuntu/Debian

```
$ sudo apt install mysql-server
```

1- entrar como *administrador MySQL*

```
$ mysql --user=root mysql
```

A password tinha sido determinada aquando da instalação... (Ubuntu/Debian: pacote mysql-server)

2- criar uma BD nova (aqui designada *tmpdb*):

```
CREATE DATABASE tmpdb;
```

3- criar um perfil de utilizador para esta BD com plenos privilégios:

```
GRANT ALL PRIVILEGES ON tmpdb.* TO tmp@'%' IDENTIFIED BY 'tmp_pass';
```

Este comando dá todos os privilégios sobre todas as tabelas desta BD.
O beneficiário é um utilizador *tmp* que poderá aceder de qualquer origem remota (*%*) com a password indicada.
Opcionalmente, podemos acrescentar "`WITH GRANT OPTION`" delegando poderes de passagem de privilégios a outros utilizadores.

4- `logout`: sair do MySQL (da sessão de administrador)

5- aceder à nova BD com o utilizador *tmp*:

```
$ mysql -h localhost --user=tmp --password=tmp_pass tmpdb
```

Sintaxe alternativa: consultar a *man page* de mysql.

6- comandos SQL para criar tabelas e para inserir, modificar ou consultar os dados
https://dev.mysql.com/doc/refman/8.0/en/sql-statements.html

Nota: há outras abordagens para criar uma BD. A sequência de passos acima é uma das formas possíveis.

Inicialização de uma BD em PostgreSQL

0- instalação so SDGB em Ubuntu/Debian

```
$ sudo apt install postgresql
```

1- Entrar no sistema (Linux) como o utilizador de Postgres
Por exemplo:

```
$ sudo su - postgres
[sudo] password for username:
postgres@maquina:~$
```

2- Criar um utilizador comum (sem previlégios especiais)

```
$ createuser --no-superuser --no-createdb --no-createrole user1 ;
```

3- Criar uma BD, especificando o utilizador anterior como o *owner*:

```
$ createdb -O user1 bd1;
```

4- Definir a password para aquele utilizador aceder à sua BD (ainda com o user linux *postgres*):

```
$ psql bd1
# alter user user1 with password 'umaPass';
# sair com Ctrl+D
... e sair também da sessão linux com o utilizador postgres: Ctrl+D
```

5- Tentar um acesso remoto à BD, já com o utilizador certo:

```
$ psql bd1 -U user1 -h localhost
# ... e inserir a password
```

6- Comandos SQL para criar tabelas e para inserir, modificar ou consultar os dados:
https://www.postgresql.org/docs/12/sql.html

Nota: há outras abordagens para criar uma BD. A sequência de passos acima é uma das formas possíveis.

Drivers JDBC

https://jdbc.postgresql.org/download.html
http://dev.mysql.com/downloads/connector/

(para os exercícios desta aula, **já recebe** o driver no *pack* inicial)

Erros frequentes

Ocorreu um erro `"java.lang.ClassNotFoundException: org.postgresql.Driver" ?`

```
Atualize a classpath (ou garanta que o jar do driver está nas libraries do projeto NetBeans). Veja um exemplo abaixo.
```

Problema `The column name name not found ?`
é possível que exista menção a um campo "name"... que devia ser nome (ou vice-versa). Confirme listando os nomes da tabela, e compare com os nomes de coluna que a aplicação está a usar. Se necessário, ajuste.

Problemas de permissões?

- `A connection error has occurred: FATAL:  password authentication failed for user "login"`
  - veja a configuração... e note que a password não é de utilizador no sistema, mas antes a definida para esta BD para acesso remoto.
- `org.postgresql.util.PSQLException: ERROR: permission denied for table personl99999`
  - o owner da tabela é mesmo o utilizador q está a executar os comandos? Poderia resolver-se com atribuição de privilégios (GRANT SELECT...).
  - num cenário mais realista, poderia querer configurar origens (IP, rede) autorizadas a aceder a cada BD no SGBD
    - /etc/postgresql/11/main/pg_hba.conf
      host sdist all 127.0.0.1/32 password

Contacto com BD existente (i)

- cada aluno tem já uma área no servidor `alunos.di.uevora.pt`
- cada aluno tem uma BD Postgres cujo nome é o respetivo login lxxxxx, com password "teste" (note que a password será pedida apenas em acesso remoto).

1- testar o acesso à máquina (se necessário, definir a password com o docente)

```
$ ssh -l lxxxxx alunos.di.uevora.pt
```

2- Aceder localmente (em alunos.di) à BD sdist

```
$ psql sdist -U sdist -h localhost
# a password é sd2021
```

3- Listar as relações existentes:

```
sdist=> \dt

          List of relations
 Schema |     Name     | Type  | Owner
--------+--------------+-------+-------
 public | personl99999 | table | sdist
(1 row)
```

3.1- no seu PC será preciso criar a tabela

```
sdist> CREATE TABLE personl99999 (id integer,  name varchar(128), birth timestamp without time zone);
```

4- Ver a estrutura da tabela (campos e respectivos tipos):

```
sdist=> \d personl99999

 Table "public.personl99999"
 Column |            Type             | Modifiers
--------+-----------------------------+-----------
 id     | integer                     |
 name   | character varying(128)       |
 birth  | timestamp without time zone |
```

5- Listar os tuplos (registos) na tabela *personl99999*:

```
sdist=> select * from personl99999 ;
```

Sintaxe para o comando SELECT: http://www.postgresql.org/docs/8.2/static/queries.html

6- Inserir um registo com o nome do aluno
Documentação com a Sintaxe: http://www.postgresql.org/docs/8.2/static/dml-insert.html

7- Listar o novo conteúdo da tabela.

8- Terminar a ligação à BD: Ctrl+D

9- Aceder à própria BD do aluno em `alunos.di`:
Sintaxe curta:

```
# assumindo que está na sessão de SSH no servidor
$ psql lxxxxx          (onde xxxxx é o nº de aluno)
```

10- Definir uma password para acesso remoto (já depois de aceder localmente à BD com psql - note a *prompt*)
Inicialmente, a sua password é "teste".

```
  # alter user lxxxxx with password 'novaPass';
```

11- terminar ligação: Ctrl+D

12- Testar um acesso remoto (ainda com o cliente postgres psql)

```
$ psql lxxxxx -U lxxxxx -h localhost
# ... inserir a nova password para a BD - que não é a mesma do aluno no sistema Linux... é específica deste perfil e para esta BD.
```

Nota Importante: se tiver conetividade até ao porto do SGBD (5432), pode executar os clientes (psql ou outro) no seu próprio computador, com o cuidado de trocar "`localhost`" por "`alunos.di.uevora.pt`", e desse modo dispensa a sessão de SSH. Dentro da rede UÉ deve conseguir; de casa não funciona... devido aos filtros de rede exterior-interior para portos não autorizados.

Acesso à BD desde a Aplicação

Os SGBDs mais comuns permitem o acesso remoto através de interfaces **JDBC** - *Java Database Connectivity*. Isto permite-lhe construir, modificar e consultar uma base de dados usando um conjunto de classes Java (que fazem parte do driver JDBC do SGBD) a partir da sua aplicação.

O que é preciso?

1- driver: a classpath deve incluir um jar ou zip com o conjunto de classes Java que implementam a conectividade com o SGBD.
Exemplo: `postgresql.jar` ou `mysql-connector-java.jar`

2- inicialização da ligação à BD (exemplo em PostgreSQL)

```
       java.sql.Connection con= null;
       java.sql.Statement stmt=

try {
           Class.forName ("org.postgresql.Driver");
           // url = "jdbc:postgresql://host:port/database",
           con = DriverManager.getConnection("jdbc:postgresql://"+PG_HOST+":5432/"+PG_DB,
                                             USER,
                                             PWD);

           stmt = con.createStatement();

       }
       catch (Exception e) {
           e.printStackTrace();
           System.err.println("Problems setting the connection");

       }
```

3- exemplo de inserção (ou outro comando SQL de modificação)

```
        // update/insert

        try {
            stmt.executeUpdate("insert into personl99999 values(1,'O.MeuNome','"+new java.util.Date()+"')");
            // o objeto java.util.Date será convertido para String com toString(). Se não for aceite pelo Postgres, use um DateFormat.
        }
        catch (Exception e) {
            e.printStackTrace();
            System.err.println("Problems on insert...");
       }
```

```
// atenção às restrições, como chaves primárias: ao inserir (acima) o identificador 1 poderia levar a um erro, dependendo de haver uma chave naquele campo e o valor estar repetido.
```

4- exemplo de consulta

```
        try {

            ResultSet rs = stmt.executeQuery("SELECT id,name,birth,extract(hour from birth) as hh from personl99999 order by birth");

            while (rs.next()) {
                int id= rs.getInt("id");
                String name = rs.getString("name");
                java.sql.Timestamp birth= rs.getTimestamp("birth");
                java.sql.Time time= rs.getTime("birth");

                System.out.println("Id: "+id+"  name: "+name+
                                   "  birth: "+birth+ "   time: "+time);
            }
            rs.close(); // muito importante depois da consulta!
        }
        catch (Exception e) {
            e.printStackTrace();
            System.err.println("Problems retrieving data from db...");
        }
```

5 - finalizar a ligação à BD

```
        // importante: fechar a ligacao 'a BD

        try {
            stmt.close();
            con.close();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
```

Pode consultar a API das classes referidas (ResultSet, Statement) e outras no pacote java.sql

---

Exercício 1

Obtenha o código inicial, versão em projeto NetBeans.

Confirme que nas bibliotecas (*Libraries*) do projeto se encontra o JAR com o driver de Postgres (poderá ter de acertar o path do jar, dentro da pasta do projeto).
Estude o código fonte fornecido...

Altere o código SQL do insert para que o segundo campo corresponda ao seu nome.

Compile e execute a aplicação.
Passe argumentos para aceder à BD sdist com o utilizador sdist e a password antes indicada.

??
Ocorreu um erro "`java.lang.ClassNotFoundException: org.postgresql.Driver`" ?
Atualize a `classpath` (ou garanta que o jar do driver está nas libraries do projeto NetBeans).
Volte a executar.

```
$ java -classpath build/classes:resources/postgresql.jar sd.TesteAcessoBD
```

O **resultado** (output de execução) deve ser uma consulta à tabela, após a inserção do seu nome (e outros registos para os seus colegas).

caso tenha problemas com o driver de Postgres, veja no link (acima) a versão ideal para o seu Java e Postgres (exemplo: este)

Confirme o conteúdo da tabela, ligando-se diretamente à base de dados Postgres em `alunos.di.uevora.pt`, com o comando linux:

```
# assumindo que já está ligado no servidor, em sessão SSH:
$ psql sdist -U sdist -h localhost
(em seguida pede-se uma password: sd2021)
E pode dar o comando SQL:
 select * from person99999;
```

Exercício 2

Implemente uma aplicação (classe **sd.Avaliacao**) com um menu de linha de comandos, que permita realizar as seguintes operações:
- registar um aluno (*nº, nome, BI, telefone*)
- registar uma disciplina (*código, nome, descrição*)
- inscrever um aluno numa disciplina
- listar os alunos
- listar disciplinas
- listar inscrições
- listar os alunos inscritos na disciplina com o código ***X***
- listar disciplinas a que o aluno ***N*** está inscrito

Use a sua BD como repositório persistente. (Note que **não é** a BD `sdist`, mas a sua BD `l4xxxx`)

Outros exemplos: Azure
https://docs.microsoft.com/en-us/azure/sql-database/sql-database-connect-query-java
---
Note: esta é a abordagem de ligação explicitamente controlada pelo programador... há outras de ter persistência dos dados em BDs, com maior dinamismo e menos intervenção direta do programador... (adiante)