| |
|---|
| **Objectius de la pràctica:**<br>- Conèixer com accedir a una base de dades amb Java |
| **Instruccions:**<br>- Responeu a l'espai de cada pregunta, si ho feu amb diapositives enganxeu la diapositiva en aquest mateix espai.<br>- Es valorarà la presentació i els comentaris al codi |
| **Criteris d'avaluació:**<br>- Cada exercici té la mateixa puntuació<br>- Les metodologies de treball pròpies, organització personal i participació valen un 10% |
| **Entrega:**<br>- Un arxiu .zip anomenat: **PRx.y-NomCognom.zip**<br>● PRx.y correspon al codi de la pràctica, per exemple PR1.1<br>● NomCognom correspon al nom i primer cognom de cada participant<br>- L'arxiu **.zip** conte:<br>● Aquest document emplenat en format **.pdf** anomenat **memoria.pdf**<br>● Els arxius necessaris per fer anar la pràctica |

**Materials:**

- Teoria sobre SQLite
- Enllaços a tutorials de Moodle.
- Necessiteu una eina per programar en JAVA.
- Feu servir Google per buscar els tutorials que us serveixin millor.
- Repositori amb exemples d'SQLite: https://github.com/optimisme/DAM-JavaExempleSQLite
- Repositori amb esquelet en blanc de projecte (opcional): https://github.com/jpala4-ieti/DAM2-Esquelet-Basic-Projecte-Java-Maven

**Objectiu del programa:** Desenvolupar una aplicació Java que gestioni una base de dades de personatges i faccions basada en el videojoc "For Honor". Utilitza JDBC i SQLite.

Pots crear les classes que creguis necessàries per organitzar bé l'aplicació.

# Exercici 0 - Aplicació “Main.java”

**Inicialització de la Base de Dades:**

- Verifica si la base de dades existeix.
- En cas contrari, inicialitza la base de dades amb les taules següents:
    - Taula Facció:
        - id: Clau primària, tipus enter.
        - nom: Cadena de text (varchar) de màxim 15 caràcters.
        - resum: Cadena de text (varchar) de màxim 500 caràcters.
    - Taula Personatge:
        - id: Clau primària, tipus enter.
        - nom: Cadena de text (varchar) de màxim 15 caràcters.
        - atac: Número real.
        - defensa: Número real.
        - idFaccio: Clau forània que enllaça amb l’id de la taula Facció.

**Població de la Base de Dades:**

- Cerca informació sobre les faccions de “For Honor” (inclòs el portal Fandom).
- Crea almenys 3 faccions diferents (Cavallers, Vikings, i Samurais) i afegeix 3 personatges per cada facció.

Enllaços útils: For Honor - Wikipedia i fandom

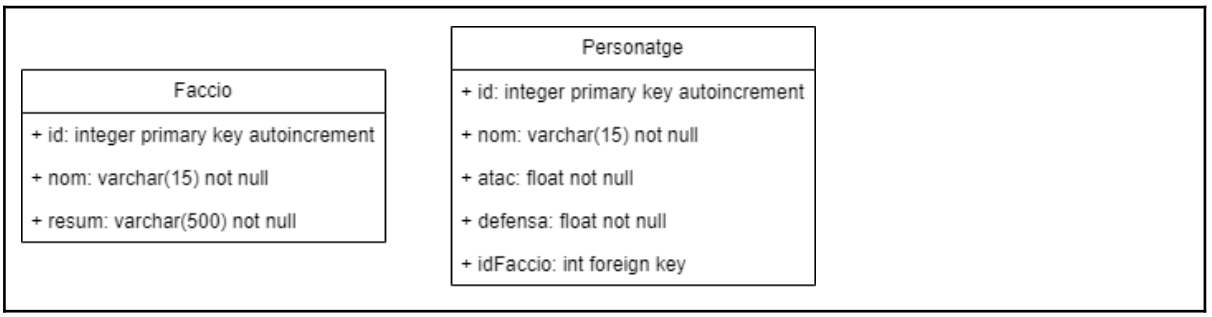
**Funcionalitats del Programa:**

- Mostrar una taula: Permet l’usuari seleccionar quina taula vol visualitzar.
- Mostrar personatges per facció: Llista els personatges pertanyents a una facció específica. Indica el nom de la facció no només l’id.
- Mostrar el millor atacant per facció: Indica quin personatge té l’atac més alt dins d’una facció seleccionada.
- Mostrar el millor defensor per facció: Indica quin personatge té la defensa més alta dins d’una facció seleccionada.
- Sortir: Tanca l’aplicació.

Nota: En cas d’empat en atac o defensa, mostrar només un dels personatges.

**Nom i Cognom:** Joel Berzal Álamo

Enganxa en aquest quadre el diagrama de la base de dades.




Enganxa una captura de pantalla que mostri l'organització del teu programa (packages, classes) i explica les principals decisions de disseny que has pres.

L'estructura del meu programa està conformada per dues classes: *UtilsSQLite* i *Main*. La primera classe té la mateixa estructura que l'exemple d'SQLite proporcionat al principi d'aquest document, tal com es pot veure en aquestes dues captures:

```java
package com.project;

import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

public class UtilsSQLite {

    public static Connection connect (String filePath) {
        Connection conn = null;

        try {
            String url = "jdbc:sqlite:" + filePath;
            conn = DriverManager.getConnection(url);
            if (conn != null) {
                DatabaseMetaData meta = conn.getMetaData();
                System.out.println("BBDD driver: " + meta.getDriverName());
            }
            System.out.println(x:"BBDD SQLite connectada");
        } catch (SQLException e) { e.printStackTrace(); }

        return conn;
    }

    public static void disconnect (Connection conn ) {
        try {
            if (conn != null) {
                conn.close();
                System.out.println(x:"DDBB SQLite desconnectada");
            }
        } catch (SQLException ex) { System.out.println(ex.getMessage()); }
    }

```

```java
    public static ArrayList<String> listTables (Connection conn) {
        ArrayList<String> list = new ArrayList<>();
        try {
            ResultSet rs = conn.getMetaData().getTables(catalog:null, schemaPattern:null, tableNamePattern:null, types:null);
            while (rs.next()) {
                list.add(rs.getString(columnLabel:"TABLE_NAME"));
            }
        } catch (SQLException ex) { System.out.println(ex.getMessage()); }
        return list;
    }

    public static int queryUpdate (Connection conn, String sql) {
        int result = 0;
        try {
            Statement stmt = conn.createStatement();
            result = stmt.executeUpdate(sql);
        } catch (SQLException e) { e.printStackTrace(); }
        return result;
    }

    public static ResultSet querySelect (Connection conn, String sql) {
        ResultSet rs = null;
        try {
            Statement stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
        } catch (SQLException e) { e.printStackTrace(); }
        return rs;
    }
}
```

Al contrari, la segona classe té diverses modificacions malgrat fer servir com a base l'exemple d'SQLite. Aquestes modificacions consisteixen en els següents aspectes clau:

- Afegida la lògica del menú d'opcions que he hagut d'implementar en les pràctiques anteriors.
- Creació de 4 mètodes que permeten gestionar les opcions disponibles en el menú (mostrar taules, mostrar personatges, mostrar personatges per facció, mostrar el millor atacant d'una facció i mostrar el millor defensor d'una facció).
- Canviada la lògica del mètode *initDataBase* perquè en lloc de crear la taula *warehouses*, creï les taules *faccio* i *personatge*.

```java
package com.project;

import java.io.File;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Scanner;

public class Main {

    static Scanner in = new Scanner(System.in); // System.in és global

    Run | Debug
    public static void main(String[] args) throws SQLException {

        String basePath = System.getProperty(key:"user.dir") + "/data/";
        String filePath = basePath + "database.db";
        ResultSet rs = null;

        // Si no hi ha l'arxiu creat, el crea i li posa dades
        File fDatabase = new File(filePath);
        if (!fDatabase.exists()) { initDatabase(filePath); }

        // Connectar (crea la BBDD si no existeix)
        Connection conn = UtilsSQLite.connect(filePath);

        // Llistar les taules
        ArrayList<String> taules = UtilsSQLite.listTables(conn);
        System.out.println(taules);

        boolean running = true;

        while (running) {

            System.out.println(x:"Escull una opcio:\n\n0) Mostrar taula Faccio\n1) Mostrar taula Personatge\n2) Mostrar personatges per faccio\n
```

```
            try {

                int opcio = Integer.valueOf(llegirLinia(text:"Opcio: "));

                switch (opcio) {
                    case 0: mostrarTaulaFaccion(conn, rs);
                            break;
                    case 1: mostrarTaulaPersonatge(conn, rs);
                            break;
                    case 2: mostrarPersonatges(conn, rs, idFaccio:1);
                            mostrarPersonatges(conn, rs, idFaccio:2);
                            mostrarPersonatges(conn, rs, idFaccio:3);
                            break;
                    case 3: mostrarPersonatgesOrdenats(conn, rs, idFaccio:1, ordre:"atac");
                            mostrarPersonatgesOrdenats(conn, rs, idFaccio:2, ordre:"atac");
                            mostrarPersonatgesOrdenats(conn, rs, idFaccio:3, ordre:"atac");
                            break;
                    case 4: mostrarPersonatgesOrdenats(conn, rs, idFaccio:1, ordre:"defensa");
                            mostrarPersonatgesOrdenats(conn, rs, idFaccio:2, ordre:"defensa");
                            mostrarPersonatgesOrdenats(conn, rs, idFaccio:3, ordre:"defensa");
                            break;
                    case 100: running = false;
                            break;
                    default: System.out.println(x:"\nOpcio fora del rang!");
                            break;
                }

                System.out.println(x:"\n**********************************************************************************************************

            } catch (Exception e) {
                System.out.println(x:"\nOpcio no numerica!\n\n**********************************************************************************************************
            }
        }

        // Desconnectar
        UtilsSQLite.disconnect(conn);
    }

    static String llegirLinia (String text) {
        System.out.print(text);
        return in.nextLine();
    }

    static void mostrarTaulaFaccion(Connection conn, ResultSet rs) {
        try {
            rs = UtilsSQLite.querySelect(conn, sql:"SELECT * FROM faccio;");
            while (rs.next()) {
                System.out.println("\nId: " + rs.getInt(columnLabel:"id")
                                   + "; Nom: " + rs.getString(columnLabel:"nom")
                                   + "; Resum: " + rs.getString(columnLabel:"resum"));
            }
        } catch (SQLException e) { e.printStackTrace(); }
    }

    static void mostrarTaulaPersonatge(Connection conn, ResultSet rs) {
        try {
            rs = UtilsSQLite.querySelect(conn, sql:"SELECT * FROM personatge;");
            System.out.println(x:"");
            while (rs.next()) {
                System.out.println("Id: " + rs.getInt(columnLabel:"id")
                                   + "; Nom: " + rs.getString(columnLabel:"nom")
                                   + "; Atac: " + rs.getString(columnLabel:"atac")
                                   + "; Defensa: " + rs.getString(columnLabel:"defensa")
                                   + "; Id Faccio: " + rs.getString(columnLabel:"idFaccio"));
            }
        } catch (SQLException e) { e.printStackTrace(); }
    }

    static void mostrarPersonatges(Connection conn, ResultSet rs, int idFaccio) {
        try {
            rs = UtilsSQLite.querySelect(conn, "SELECT * FROM personatge WHERE idFaccio = " + idFaccio + ";");
            if (idFaccio == 1) { System.out.println(x:"\nPersonatges de la faccio Cavallers:\n"); }
            else if (idFaccio == 2) { System.out.println(x:"\nPersonatges de la faccio Vikings:\n"); }
            else { System.out.println(x:"\nPersonatges de la faccio Samurais:\n"); }
            while (rs.next()) {
                System.out.println("Id: " + rs.getInt(columnLabel:"id")
                                   + "; Nom: " + rs.getString(columnLabel:"nom")
                                   + "; Atac: " + rs.getString(columnLabel:"atac")
                                   + "; Defensa: " + rs.getString(columnLabel:"defensa")
                                   + "; Id Faccio: " + rs.getString(columnLabel:"idFaccio"));
            }
        } catch (SQLException e) { e.printStackTrace(); }
    }

```

```java
    static void mostrarPersonatgesOrdenats(Connection conn, ResultSet rs, int idFaccio, String ordre) {
        try {
            rs = UtilsSQLite.querySelect(conn, "SELECT * FROM personatge WHERE idFaccio = " + idFaccio + " ORDER BY " + ordre + " DESC LIMIT 1;"
            if (ordre == "atac") {
                if (idFaccio == 1) { System.out.println(x:"\nMillor atacant de la faccio Cavallers:"); }
                else if (idFaccio == 2) { System.out.println(x:"\nMillor atacant de la faccio Vikings:"); }
                else { System.out.println(x:"\nMillor atacant de la faccio Samurais:"); }
            }
            else {
                if (idFaccio == 1) { System.out.println(x:"\nMillor defensor de la faccio Cavallers:"); }
                else if (idFaccio == 2) { System.out.println(x:"\nMillor defensor de la faccio Vikings:"); }
                else { System.out.println(x:"\nMillor defensor de la faccio Samurais:"); }
            }
            while (rs.next()) {
                System.out.println("Id: " + rs.getInt(columnLabel:"id")
                                   + "; Nom: " + rs.getString(columnLabel:"nom")
                                   + "; Atac: " + rs.getString(columnLabel:"atac")
                                   + "; Defensa: " + rs.getString(columnLabel:"defensa")
                                   + "; Id Faccio: " + rs.getString(columnLabel:"idFaccio"));
            }
        } catch (SQLException e) { e.printStackTrace(); }
    }

    static void initDatabase (String filePath) {

        // Connectar (crea la BBDD si no existeix)

        Connection conn = UtilsSQLite.connect(filePath);

        // Esborrar la taula Facció (per si existeix)

        UtilsSQLite.queryUpdate(conn, sql:"DROP TABLE IF EXISTS faccio;");

        // Crear una nova taula Facció

                                   + "id INTEGER PRIMARY KEY AUTOINCREMENT,"
                                   + "nom VARCHAR(15) NOT NULL,"
                                   + "resum VARCHAR(500) NOT NULL);");

        // Esborrar la taula Personatge (per si existeix)

        UtilsSQLite.queryUpdate(conn, sql:"DROP TABLE IF EXISTS personatge;");

        // Crear una nova taula Personatge

        UtilsSQLite.queryUpdate(conn, "CREATE TABLE IF NOT EXISTS personatge ("
                                   + "id INTEGER PRIMARY KEY AUTOINCREMENT,"
                                   + "nom VARCHAR(15) NOT NULL,"
                                   + "atac FLOAT NOT NULL,"
                                   + "defensa FLOAT NOT NULL,"
                                   + "idFaccio INT FOREING KEY);");

        // Afegir 3 faccions diferents

        UtilsSQLite.queryUpdate(conn, sql:"INSERT INTO faccio (nom, resum) VALUES (\"Cavallers\", \"Els Cavallers son una de les quatre faccions
        UtilsSQLite.queryUpdate(conn, sql:"INSERT INTO faccio (nom, resum) VALUES (\"Vikings\", \"Els vikings son una de les quatre faccions jug
        UtilsSQLite.queryUpdate(conn, sql:"INSERT INTO faccio (nom, resum) VALUES (\"Samurais\", \"El Samurai de l'Imperi de l'Alba es una de le

        // Afegir 3 personatges per cada facció

        UtilsSQLite.queryUpdate(conn, sql:"INSERT INTO personatge (nom, atac, defensa, idFaccio) VALUES (\"Warden\", 130, 120, 1);");
        UtilsSQLite.queryUpdate(conn, sql:"INSERT INTO personatge (nom, atac, defensa, idFaccio) VALUES (\"Conqueror\", 140, 120, 1);");
        UtilsSQLite.queryUpdate(conn, sql:"INSERT INTO personatge (nom, atac, defensa, idFaccio) VALUES (\"Pacekeeper\", 120, 120, 1);");
        UtilsSQLite.queryUpdate(conn, sql:"INSERT INTO personatge (nom, atac, defensa, idFaccio) VALUES (\"Highlander\", 125, 120, 2);");
        UtilsSQLite.queryUpdate(conn, sql:"INSERT INTO personatge (nom, atac, defensa, idFaccio) VALUES (\"Shaman\", 120, 120, 2);");
        UtilsSQLite.queryUpdate(conn, sql:"INSERT INTO personatge (nom, atac, defensa, idFaccio) VALUES (\"Jormungandr\", 130, 140, 2);");
        UtilsSQLite.queryUpdate(conn, sql:"INSERT INTO personatge (nom, atac, defensa, idFaccio) VALUES (\"Shinobi\", 120, 135, 3);");
        UtilsSQLite.queryUpdate(conn, sql:"INSERT INTO personatge (nom, atac, defensa, idFaccio) VALUES (\"Hitokiri\", 140, 130, 3);");
        UtilsSQLite.queryUpdate(conn, sql:"INSERT INTO personatge (nom, atac, defensa, idFaccio) VALUES (\"Kyoshin\", 120, 120, 3);");

        // Desconnectar
        UtilsSQLite.disconnect(conn);
    }
}
```

URL Repositori GitHub.

https://github.com/joelberzalgithub/AMS2-MP06-PR2.1-BerzalJoel