## Transfermarkt Stílusú Konzolos Java Alkalmazás Tervezése

**1. Általános Koncepció**

A fő cél egy olyan konzolos alkalmazás létrehozása, amely a Premier League játékosait és csapatait kezeli. Az adatok alapját egy külső **ingyenes futball API** biztosítja (csapatok, játékosnevek, pozíciók), míg a **piaci érték** adatok az alkalmazás saját **SQLite adatbázisában** tárolódnak és a felhasználó által frissíthetőek. A komplexitást a két adatbázis-tábla (teams és players) és a relációs lekérdezések (JOIN) adják.

**2. Technológiai Halom**

| **Réteg** | **Technológia** | **Cél** |
| --- | --- | --- |
| **Programozási nyelv** | Java (11+ ajánlott) | Az alkalmazás logika és a konzolos felület. |
| **API Kliens** | java.net.http.HttpClient | HTTP kérések küldése a külső futball API felé. |
| **JSON Feldolgozás** | **Jackson** (vagy Gson) | JSON válaszok átalakítása Java objektumokká. |
| **Adatbázis** | **SQLite** | Beágyazott, szervermentes adatbázis a perzisztens tároláshoz. |
| **Adatbázis Kezelés** | **JDBC** | Kapcsolódás és SQL parancsok végrehajtása az SQLite-on. |

**3. Adatmodell (Java Osztályok és SQLite Táblák)**

**A. Java Objektum Modell**

* **Player Osztály:**
  + String name
  + int jerseyNumber
  + String position
  + int teamId (Idegen kulcs)
  + int goals, int assists, int yellowCards (Valós statisztikák)
  + double marketValue\_m\_euro (Manuálisan karbantartott érték)
  + **Metódusok:** Getterek/Setterek, toString(), calculatePerformanceScore()
* **Team Osztály:**
  + int id
  + String name
  + double totalMarketValue (Dinamikusan számolt)

**B. SQLite Adatbázis Séma**

**teams Tábla (Csapatok)**

| **Mező neve** | **Típus** | **Kötöttség** | **Leírás** |
| --- | --- | --- | --- |
| team\_id | INTEGER | PRIMARY KEY | Egyedi azonosító. |
| team\_name | TEXT | NOT NULL, UNIQUE | Csapat neve (pl. "Manchester City"). |
| total\_market\_value | REAL | DEFAULT 0.0 | A keret összesített piaci értéke. |

**players Tábla (Játékosok)**

| **Mező neve** | **Típus** | **Kötöttség** | **Leírás** |
| --- | --- | --- | --- |
| player\_id | INTEGER | PRIMARY KEY | Egyedi azonosító. |
| nev | TEXT | NOT NULL | Játékos neve. |
| mezszam | INTEGER | UNIQUE | Játékos mezszáma. |
| pozicio | TEXT |  | (Csatár, Védő, stb.) |
| golok | INTEGER | DEFAULT 0 | Valós statisztika. |
| piaci\_ertek\_m\_euro | REAL | DEFAULT 0.0 | **Transfermarkt érték.** |
| team\_id | INTEGER | FOREIGN KEY | Kapcsolat a teams táblával. |

**4. Az Alkalmazás Logikája (Osztályok és Funkciók)**

**A. MainApp (Fő belépési pont)**

* main() metódus: Példányosítja a PlayerStatsManager-t és elindítja a start() metódust.

**B. DatabaseManager (Adatbázis Kezelő)**

* connect(): Létrehozza a JDBC kapcsolatot az focis\_db.sqlite fájlhoz.
* createTables(): Létrehozza a **teams** és **players** táblákat (ha még nem léteznek).
* insertTeam(Team team) / insertPlayer(Player player): INSERT SQL utasítások.
* updateMarketValue(String name, double newValue): UPDATE SQL utasítás.
* getPlayerStats(): SELECT \* FROM players JOIN teams... (Összes adat lekérdezése).
* queryTopMarketValueTeams(): SQL lekérdezés a legértékesebb csapatok rangsorolásához (**SUM()** és **GROUP BY** használatával).

**C. APIManager (Külső Adatkezelő)**

* fetchTeams(): Lekérdezi a Premier League csapatok listáját.
* fetchPlayersForTeam(int teamId): Lekérdezi az adott csapat játékosait.
* **Adatbetöltési logika:** A sikeres API hívás után a JSON adatok a DatabaseManager megfelelő insert metódusát hívják meg.

**D. PlayerStatsManager (Konzol UI és Fő Logika)**

* start(): A fő menü ciklus megjelenítése és a felhasználói bemenet kezelése.
* **Indítási fázis:**
  1. DatabaseManager.createTables()
  2. APIManager.fetchTeams() (Csapatok API-ból az adatbázisba)
  3. APIManager.fetchPlayers() (Játékosok API-ból az adatbázisba, 0.0 piaci értékkel)
  4. Adatok betöltése az adatbázisból a belső memóriába a menü számára.

**5. Konzolos Menüstruktúra és Funkciók**

1. **Rangsorok és Kimutatások**
   * 1.1. Legértékesebb Csapatok Rangsora (DB: teams tábla és total\_market\_value mező/összesítés)
   * 1.2. Legértékesebb Játékosok Rangsora (Összes játékos, érték alapján csökkenő)
   * 1.3. Legjobb Játékosok Pozíciók Szerint (Top 5 Csatár/Védő/stb. érték alapján)
   * 1.4. Performance Score Rangsor (Java: dinamikus számítás alapján)
2. **Játékos Keresése/Szűrése**
   * 2.1. Keresés név alapján (részleges egyezés)
   * 2.2. Játékosok listázása csapat szerint
3. **Adatok Frissítése (Transfermarkt Funkció)**
   * 3.1. Játékos piaci értékének frissítése (UPDATE a players táblában).
   * 3.2. Játékos statisztikájának frissítése (Gólok, Gólpasszok, Lapok).
   * *Megjegyzés: Frissítés után a programnak újra kell számolnia az érintett csapat total\_market\_value mezőjét!*
4. **$Exit$ / Kilépés**

Ez a terv biztosítja, hogy a projekted ne csak egy egyszerű lista legyen, hanem valódi adatbázis-kezelést (JOIN-okat, INSERT-eket, UPDATE-eket) és hálózati kommunikációt (API) is tartalmazzon, ami nagyban növeli a projekt értékét. Melyik lépéssel kezdenéd a NetBeans-ben?