Analiza sieci podań

Paweł Klinkowski

Projekt zaliczeniowy z przedmiotu Przetwarzanie danych w chmurach obliczeniowych

1. Cel projektu

Celem projektu było stworzenie proof of concept aplikacji internetowej wykorzystującej grafową bazę danych Neo4j do analizy relacji pomiędzy zawodnikami na podstawie podań w meczu piłkarskim.

2. Zakres funkcjonalny aplikacji

Aplikacja umożliwia:

- Import nowego meczu poprzez przesłanie pliku JSON (StatsBomb)
- Automatyczne utworzenie struktury grafowej:
  - mecz,
  - drużyny,
  - zawodnicy,
  - relacje podań
- Wizualizacje sieci podań osobno dla każdej drużyny
- Filtrowanie danych
- Obliczanie i prezentację
  - liczby podań pomiędzy zawodnikami,
  - średnich pozycji zawodników na boisku,
  - rankingu zawodników z największą liczbą podań

3. Model danych

   Węzły:

   - Match
     - matchId
     - importedAt
   - Team
     - teamId
     - name
   - Player
     - playerId
     - name

   Relacje:

   - (:Match)-[:INVOLVES_TEAM]->(:Team)
   - (:Team)-[:HAS_PLAYER]->(:Player)
   - (:Player)-[:PASSED_TO]->(:Player)

   Relacja PASSED_TO przechowuje dane o podaniu:

   - matchId
   - teamId
   - minute
   - second
   - succesful
   - startX, startY
   - endX, endY

4. Import danych

   Import meczu polega na:

   - wczytaniu pliku JSON pochodzącego z bazy StatsBomb z eventami z meczu
   - Utworzeniu węzła Match
   - Sprawdzeniu istnienia drużyn i zawodników (MERGE)
   - Filtrowaniu tylko eventów typu Pass
   - Zapisaniu każdego podania jako osobnej relacji PASSED_TO
   - Obliczeniu średnich pozycji zawodników na podstawie współrzędnych podań

5. Zapytania grafowe (przykłady)

Sieć podań drużyny w zadanym przedziale minut:

```
const cypher = `
  MATCH (a:Player)-[p:PASSED_TO {matchId:$matchId}]->(b:Player)
  WHERE p.minute >= $fromMin AND p.minute < $toMin
    AND ($teamId IS NULL OR p.teamId = $teamId)
    AND ($successfulOnly = false OR p.successful = true)
  RETURN a.playerId AS fromId, a.name AS from,
         b.playerId AS toId,   b.name AS to,
         count(p) AS count
  ORDER BY count DESC
`;
```

Średnia pozycja zawodnika:

```
`
MATCH (a:Player)-[p:PASSED_TO {matchId:$matchId}]->()
WHERE p.minute >= $fromMin AND p.minute < $toMin
  AND ($teamId IS NULL OR p.teamId = $teamId)
  AND p.startX IS NOT NULL AND p.startY IS NOT NULL
RETURN a.playerId AS playerId, a.name AS name,
       avg(p.startX) AS avgX, avg(p.startY) AS avgY,
       count(p) AS passesMade
ORDER BY passesMade DESC
`,
```

Ranking podań zawodników:

```
`
MATCH (a:Player)-[p:PASSED_TO {matchId:$matchId}]->()
WHERE p.minute >= $fromMin AND p.minute < $toMin
  AND p.teamId = $teamId
RETURN a.playerId AS playerId,
       a.name AS name,
       count(p) AS attempts,
       sum(CASE WHEN p.successful = true THEN 1 ELSE 0 END) AS completed
ORDER BY attempts DESC
LIMIT $limit
`
```

6. Wizualizacja

Do wizualizacji wykorzystano bibliotekę react-force-graph-2d. Węzły reprezentują zawodników, a krawędzie liczbę podań między nimi. Grubość krawędzi jest proporcjonalna do liczby podań. Pozycja węzłów jest ustalana na podstawie średniej pozycji podań zawodnika, co pozwala na realistyczne odwzorowanie ustawienia drużyny na boisku w danym przedziale czasu.

7. Technologie
    - Backend
        - Node.js
        - Express.js
        - Neo4j AuraDB
    - Frontend
        - React (Vite)
        - react-force-graph-2d
        - HTML5/CSS
8. Możliwe rozszerzenia

    Projekt można poprawić i rozszerzyć na wiele sposobów. Na pewno przydałaby się poprawa frontendu tak, aby grafy były bardziej przejrzyste. Można również przeanalizować inne aspekty meczu takie jak strzały, dryblingi, pressingi i straty piłki, na co pozwala baza StatsBomb. Dobrym pomysłem byłoby również porównywanie wielu meczów.