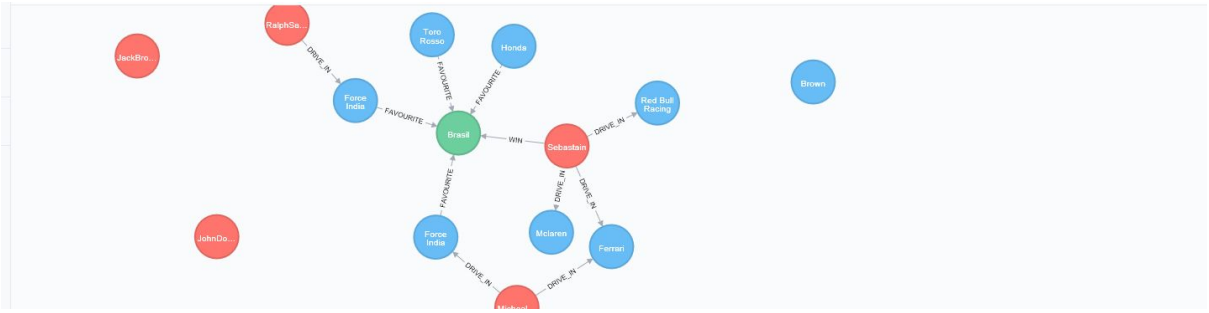
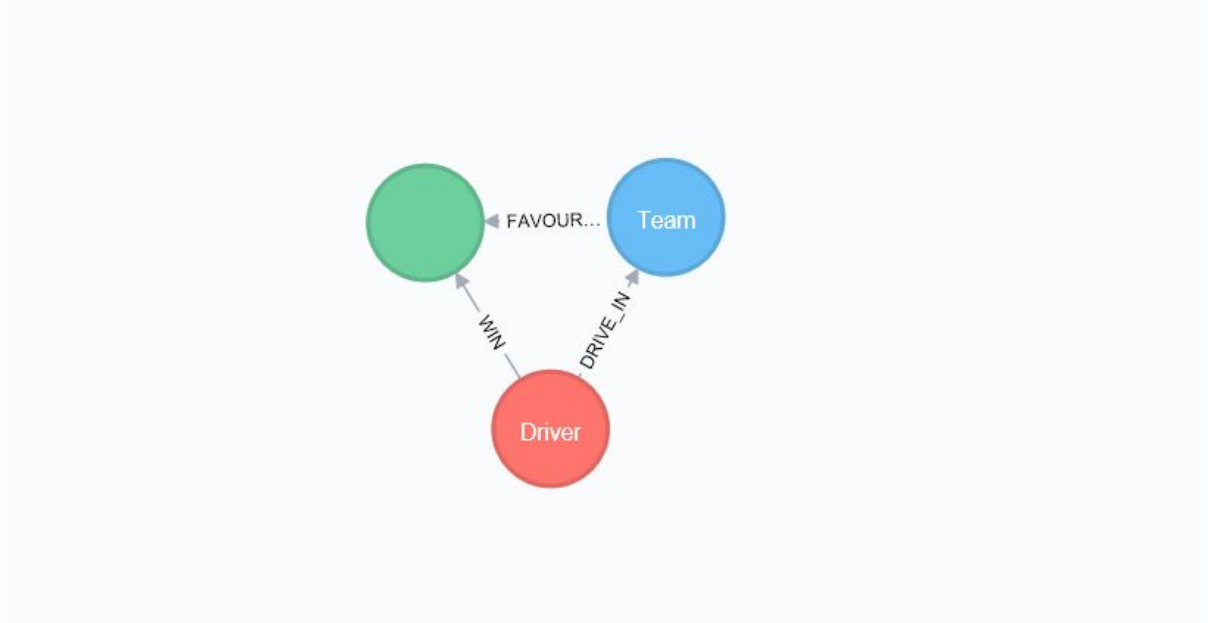
GRAFOWE BAZY DANYCH - NEO4J

1. Moja baza danych to baza kierowców Formuły 1 , zawodów GranPrix i zespołów Formuły 1. W moim projekcie wyróżniłem 3 rodzaje relacji :

- DRIVE_IN - relacja pomiędzy kierowcą a zespołem, oznaczająca, że kierowca jeździł dla danego zespołu
- WIN - relacja pomiędzy kierowcą a GranPrix, oznaczająca, że kierowca wygrał kiedykolwiek te zawody
- FAVOURITE - relacja pomiędzy zespołem a GranPrix, oznaczająca, iż jest to jeden z ulubionych torów danego zespołu, tam ten zespół zazwyczaj dobrze sobie radzi



Wynik działania funkcji “call db.schema()”



2. Funkcje do tworzenia relacji i obiektów w grafie :

```
public String addDriver(String name, int birthDate, String nationality){
1.
2.          return graphDatabase.runCypher(String.format("CREATE (d:Driver{ name : \"%s\",
    nat : \"%s\" , year : \"%d\" })",
3.                  name, nationality, birthDate));
```

```
4.
5.      }
6.
7.      public String addTeam(String name, int year, String founder){
8.
9.          return graphDatabase.runCypher(String .format("CREATE(t : Team{ name : \"%s\",
    year : \"%d\"," +
10.             "founder : \"s\"})", name, year, founder));
11.     }
12.
13.     public String addRace(String country, int year){
14.
15.         return graphDatabase.runCypher(String.format("CREATE(g:GranPrix{ country :
    \"%s\", year : \"%d\"}) ", country, year));
16.     }
17.
18.     public String addWINrelation(String driver, String race){
19.
20.         return graphDatabase.runCypher(String.format( "MATCH (d : Driver), (r:GranPrix)" +
21.             "where d.name = \"%s\" AND r.country = \"%s\" " +
22.             "CREATE (d)-[re:WIN]->(r) return re ", driver, race));
23.     }
24.
25.     public String addDRIVE_INrelation(String driver, String name){
26.
27.         return graphDatabase.runCypher(String.format("MATCH (d : Driver), (t:Team)" +
28.             "where d.name = \"%s\" AND t.name = \"%s\" " +
29.             "CREATE (d)-[re:DRIVE_IN]->(t) return re  ", driver, name));
30.     }
31.
32.     public String addFAVOURITErelation(String team, String race){
33.
34.         return graphDatabase.runCypher(String.format("MATCH (t : Team), (g:GranPrix)" +
35.             "where t.name = \"%s\" AND g.country = \"%s\" " +
36.             "CREATE (t)-[re:FAVOURITE]->(g) return re ", team, race));
37.
38.     }
```

3. Prosty generator danych :

```
1.      public void generate(){
2.
3.
4.          String names[] = new String[] {"John", "David" , "Michael", "Ralph", "Jason", "Jack"};
5.          String surnames[] = new String[] {"Brown", "Doster", "Foster", "Sailer", "Martin"};
6.          String countries[] = new String [] {"Germany", "Great Britain", "Australia", "Finland",
    "Brasil"};
7.          String races[] = new String[] {"Germany", "Great Britain", "Australia", "Brasil",
    "Finland", "Hungary", "Canada", "France"};
8.          String teams[] = new String[] {"Ferrari", "McLaren", "Mercedes", "Force India",
    "Honda", "Red Bull", "Renault", "BMW"};
9.          for(int i=0;i<15;i++){
10.             this.addDriver(names[i%6] + surnames[i%5], 1980 + i, countries[i%5]  );
```

```
    }
    for(int i =0; i< 8;i++){
        this.addTeam(teams[i], 1950 + i * 5, names[(4 + i)%6] + surnames[(3+ i) % 5]);
    }
    for(int i=0;i<8;i++){
        this.addRace(races[i], 1950 + i * 4);
    }
    for(int i=0;i<15;i++ ){

        this.addWINrelation(names[i%6] + surnames[i%5], races[i%8]);
        if(i > 7) {
            this.addWINrelation(names[i % 6] + surnames[i % 5], races[(i + 1) % 8]);
        }
        if(i > 12){
            this.addWINrelation(names[i % 6] + surnames[i % 5], races[(i + 2) % 8]);
        }
    }

    for(int i =0;i< 15 ;i++){
        this.addDRIVE_INrelation(names[i%6] + surnames[i%5], teams[i%8]);
        this.addDRIVE_INrelation(names[i%6] + surnames[i%5], teams[(1+i)%8]);
        this.addDRIVE_INrelation(names[i%6] + surnames[i%5], teams[(2+i)%8]);
    }

    for(int i=0;i<20;i++){
        this.addFAVOURITErelation(teams[i%8], races[(1 + i)%8]);
        this.addFAVOURITErelation(teams[i%8], races[(2 + i)%8]);
    }

}
```

4. Funkcje do pobierania wszystkich relacji dla danego węzła:

```
public String getAllRealationsForNode(String name, int type){

    if(type == 0) return getAllRelationsForDriver(name);
    if(type == 1) return getAllRealationsForTeam(name);
    return  getAllRelationsForRace(name);

}
```

```
public String getAllRelationsForDriver(String driver){

    return graphDatabase.runCypher(
            String.format("MATCH (d:Driver {name : \"%s\"})-[*1] - (other:GranPrix) return distinct other.country ", driver)
    ) +
            graphDatabase.runCypher(
                    String.format("MATCH (d:Driver {name : \"%s\"})-[*1] - (other:Team) return distinct other.name ", driver) ) ;
```

```
8.
9.      }
10.
11.     public String getAllRealationsForTeam(String team){
12.
13.         return graphDatabase.runCypher(
14.                 String.format("MATCH (t:Team {name : \"%s\"})-[*1] -(other:Driver) return
    distinct other.name  " , team)
15.         ) +
16.          graphDatabase.runCypher(
17.                 String.format("MATCH (t:Team {name : \"%s\"})-[*1] -(other:GranPrix) return
    distinct other.country  " , team)
18.         );
19.
20.     }
21.
22.     public String getAllRelationsForRace(String race){
23.
24.         return graphDatabase.runCypher(
25.                 String.format("MATCH (t:GranPrix {country : \"%s\"})-[*1] -(other) return
    distinct other.name ", race)
26.         );
27.     }
```

Przykładowy test :

```
1.        System.out.println(getAllRealationsForTeam("Ferrari"));
2.          System.out.println();
3.          // test , wypisanie wszytskich relacji
4.          String[] nodes = new String[]{"DavidFoster", "JohnDoster", "JohnBrown",
    "MichaelMartin", "MichaelSailer",
5.                                  "MichaelFoster", "Sebastain Vettel", "Australia", "Hungary", "Great
    Britain"};
6.          for(int i=0;i<7;i++) {
7.              if (getAllRelationsForDriver(nodes[i]).contains("Ferrari")) System.out.println("OK" +
    i);
8.          }
9.
10.
11.         for(int i=0;i<3;i++){
12.             if(getAllRelationsForRace(nodes[7+i]).contains("Ferrari")) System.out.println("OK"
    + (7+i));
13.         }
```

Działanie :

```
+--------------------+
| other.name         |
+--------------------+
| "DavidFoster"      |
| "JohnDoster"       |
| "JohnBrown"        |
| "MichaelMartin"    |
| "MichaelSailer"    |
| "MichaelFoster"    |
| "Sebastain Vettel" |
+--------------------+
7 rows
+-----------------+
| other.country   |
+-----------------+
| "Australia"     |
| "Hungary"       |
| "Great Britain" |
+-----------------+
3 rows


OK0
OK1
OK2
OK3
OK4
OK5
OK6
OK7
OK8
OK9
```

5. Funkcje zwracające najkrótsze ścieżki między węzłami :

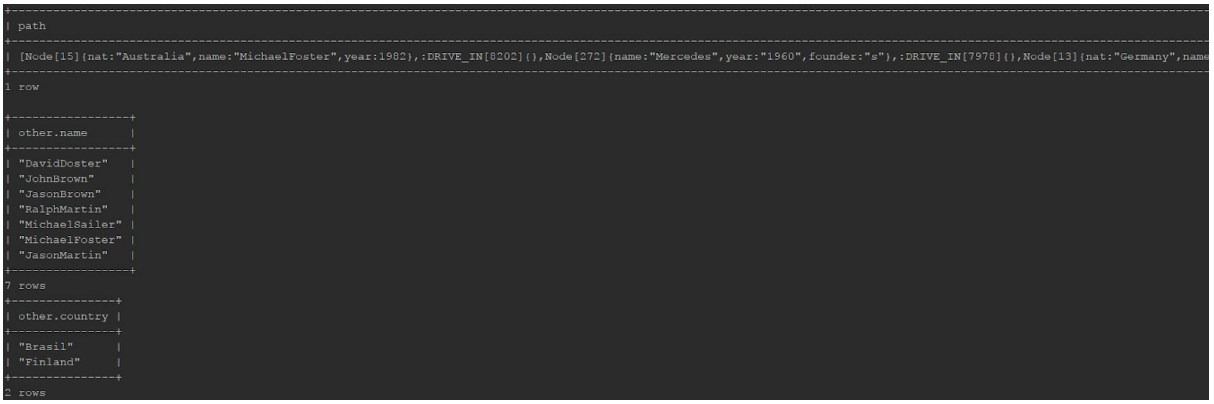
```
1.    public String findShortestPathDrivers(String driver1, String driver2) {
2.
3.
4.        return graphDatabase.runCypher(
5.              String.format("MATCH" +
6.                    "     path = shortestpath((d1:Driver) -[*]- (d2:Driver)) " +
```

```
                "     where d1.name = \"%s\" and d2.name = \"%s\" return path limit 1" , driver1, driver2)
        );

    }

    public String findShortestPathRaces(String race1, String race2){

        return graphDatabase.runCypher(String.format("MATCH" +
                "     path = shortestpath((r1:GranPrix) -[*]- (r2:GranPrix)) " +
                "     where r1.country = \"%s\" and r2.country = \"%s\" return path limit 1" , race1, race2));
    }

    public String findShortestPathTeams(String team1, String team2){

        return graphDatabase.runCypher(String.format("MATCH" +
                "     path = shortestpath((t1:Team) -[*]- (t2:Team)) " +
                "     where t1.name = \"%s\" and t2.name = \"%s\" return path limit 1" , team1, team2));
    }

    public String findShortestPathDriver_Team(String driver, String team){

        return graphDatabase.runCypher(String.format("MATCH path = shortestpath((d1:Driver)-[*]-(t1:Team))" +
                "where d1.name = \"s\" and t1.name = \"s\" return path limit 1", driver, team));
    }

    public String findShortestPathDriver_GranPrix(String driver, String race){

        return graphDatabase.runCypher(String.format("MATCH path = shortestpath((d1:Driver)-[*]-(g1:GranPrix))" +
                "where d1.name = \"s\" and g1.country = \"s\" return path limit 1", driver, race));
    }


    public String findShortestPathTeam_GranPrix(String team, String race){

        return graphDatabase.runCypher(String.format("MATCH path = shortestpath((t1:Team)-[*]-(g1:GranPrix))" +
                "where t1.name = \"s\" and g1.country = \"s\" return path limit 1", team, race));

    }
```

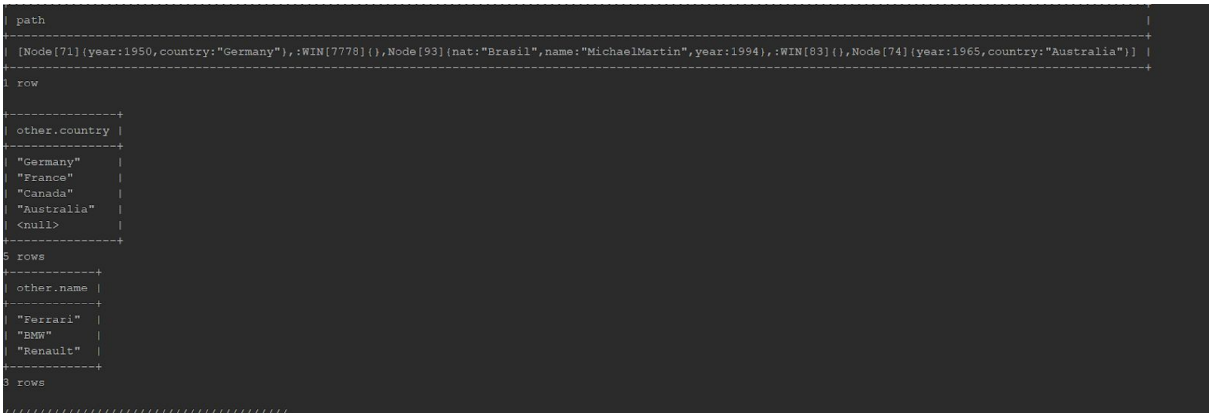
Testy :
Poprawność funkcji weryfikuję poprzez wypisanie wszystkich relacji węzłów, które są na znalezionej najkrótszej ścieżce.
Przykładowy test :

```
1.      // test1
2.      System.out.println(findShortestPathDrivers("MichaelFoster", "JohnBrown"));
3.      //Mercedes jest łącznkiem
4.      System.out.println(getAllRealationsForTeam("Mercedes"));
5.      System.out.println("////////////////////////////////////");
6.      // test2
7.      //Michael Martin jest łącznikiem
8.      System.out.println(findShortestPathRaces("Germany", "Australia"));
9.      System.out.println(getAllRelationsForDriver("MichaelMartin"));
10.     System.out.println("////////////////////////////////////////");
11.     //test3
12.     //Michael Sailer jest łącznikiem
13.     System.out.println(findShortestPathTeams("Mercedes", "Ferrari"));
14.     System.out.println(getAllRelationsForDriver("MichaelSailer"));
15.     System.out.println("////////////////////////////////////////");
```

Działanie tego testu :



Obaj kierowcy znajdują się w relacji z Mercedesem .



Oba wyścigi są w relacji z MichaelemMartinem.

```
////////////////////////////////////////
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
| path
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
| [Node[2]{founder:"DavidJones",name:"Mercedes",year:1950},:DRIVE_IN[9232]{},Node[232]{nat:"Finland",name:"MichaelSailer",year:"1988"},:DRIVE_IN[9128]{},Node[0]{year:1950,n
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
1 row

+-----------------+
| other.country   |
+-----------------+
| "Great Britain" |
| "Germany"       |
| "Belgium"       |
| "Singapur"      |
| <null>          |
+-----------------+
5 rows
+--------------+
| other.name   |
+--------------+
| "Ferrari"    |
| "Mercedes"   |
| "McLaren"    |
| "Toro Rosso" |
| "Sauber"     |
+--------------+
```

Oba zespoły są w relacji z MichaelemSailerem