WorldCupMetrics

D'Antuono Francesco Paolo 0522501767
Miron Roberto Andrei 0512110581
Link repository: https://github.com/CpDant/WorldCupMetrics

PANORAMICA

Il progetto proposto risulta essere una web-application che consente di operare su un database, che contiene informazioni sulle coppe del mondo giocate fino ad oggi. Oltre ad applicare le classiche operazioni CRUD, si vorranno effettuare analisi specifiche su particolari dati ed estrarre informazioni interessanti, per fare ciò si effettueranno delle query che combineranno varie informazioni di più tabelle. Inoltre si vorrà permettere di visualizzare le informazioni principali dei vincitori su una mappa interattiva.

OBIETTIVI

1. Operazioni CRUD
2. Visualizzazione su una mappa interattiva delle informazioni riguardanti i vincitori dei mondiali e chi ha vinto più mondiali.
3. Uso di query più particolari per fare delle analisi più specifiche sui dati che abbiamo a disposizione.

TECNOLOGIE

Il progetto userà le seguenti tecnologie:

- **MongoDB**: Un DBMS non relazionale orientato ai documenti. Utilizza documenti in un formato JSON esteso e più dinamico chiamato BSON, che facilita e accelera l'integrazione dei dati in alcune implementazioni. Per agevolare l'inserimento e la creazione del database a partire dal dataset preprocessato, è stato utilizzato anche MongoDB Compass, uno strumento con interfaccia grafica che permette di eseguire query, aggregazioni e analisi dei dati in modo visuale. Il database è stato costruito su cloud sfruttando il servizio MongoDB Atlas.
- **Python**: Un linguaggio di programmazione versatile e leggibile, utilizzato per sviluppo web, automazione, analisi dei dati e molto altro.
- **Flask**: Un micro framework web per Python, leggero e flessibile, ideale per creare API e applicazioni web semplici.
- **React**: Una libreria JavaScript per la costruzione di interfacce utente dinamiche e componenti riutilizzabili, sviluppata da Facebook.

Le tecnologie appena elencate, sono state scelte poiché risultavano essere le più utili per il tipo di progetto che si doveva affrontare e per aumentare le conoscenze di linguaggi e framework con l'approccio "learn-by-doing".

Infatti, questo tipo di tecnologie sono state studiate proprio per essere utilizzate insieme e fornire un pattern comune di progettazione e realizzazione di componenti per la gestione di db NoSQL.

DATASET SCELTO E PRE-PROCESSING DEI DATI

Il dataset utilizzato sarà:
https://www.kaggle.com/datasets/keremkarayaz/2022-world-cup-datasets?select=world_cups.csv.
In particolare, sono state estrapolati da esso i due dataset che riguardano le informazioni generali sui mondiali di calcio (world_cups.csv) e le informazioni sulle partite giocate nei mondiali di calcio (world_cup_matches.csv)

Il seguente dataset non presentava le ultime informazioni che riguardano il mondiale di calcio del 2022 in Qatar. Nel dettaglio vediamo le tabelle come sono state aggiornate:

- **world_cups.csv**: In questa tabella sono stati aggiunti i campi relativi al vincitore, al paese ospitante, al secondo classificato, al terzo classificato, all'anno specifico nel quale si è giocato, il numero di gol totali segnati nel mondiale, il numero di squadre qualificate al mondiale e il numero totale di partite giocate.
- **world_cup_matches.csv**: In questa tabella sono stati aggiunti i campi relativi all'anno in cui è stata giocata la partita, che identifica la coppa del mondo nella quale è stata giocata, la data nella quale è stata giocata la partita specifica, la fase della competizione nella quale si è giocata la partita, la squadra di casa e di trasferta (utili solo per ordinare le squadre e capire il risultato), i gol segnati dalla squadra in casa e in trasferta e un valore booleano che ci consente di capire se quella partita è stata giocata dal paese ospitante.

Tutto ciò è stato fatto per garantire una correttezza dei dati e per evitare di lavorare su dati non aggiornati per le analisi che sono state effettuate.

DIZIONARIO DEI DATI

Vediamo nel dettaglio la descrizione di ogni campo dei nostro dataset.

Tabella World_cups.csv

Colonna	Descrizione
Year	Anno in cui si è svolta la coppa del mondo
Host Country	Paese che ha ospitato il torneo
Winner	Paese vincitore del torneo
Runners-Up	Paese che si è classificato secondo
Third	Paese che si è classificato terzo
Fourth	Paese che si è classificato quarto
Goals Scored	Numero totale di reti segnate nel torneo
Qualified Teams	Numero totale di squadre che hanno partecipato al torneo
Matches Played	Numero totale di partite disputate nel torneo

Tabella World_cup_matches.csv

Colonna	Descrizione
ID	Identificatore univoco della partita
Year	Anno in cui si è svolta la Coppa del Mondo
Date	Data della partita
Stage	Fase del torneo
Home Team	Paese che gioca in casa
Home Goals	Reti segnate dalla squadra in casa
Away Goals	Reti segnate dalla squadra in trasferta
Away Team	Paese che gioca in trasferta
Host Team	Numero totale di partite disputate nel torneo

CONFIGURAZIONE DEL PROGETTO

In questa sezione verranno mostrati i passaggi da replicare, per riuscire a far funzionare il sistema implementato sulle proprie macchine:

1. Clona il repository:

   git clone https://github.com/CpDant/WorldCupMetrics.git

2. Caricare il dataset su MongoDBCompass per permettere l'accesso al database (assicurarsi che il nome che viene associato alla collection sia uguale a quello associato in app.py).
3. Installa e attiva il virtual environment di Python:

   python -m venv venv

4. Installa le dipendenze per il backend:

   pip install -r requirements.txt

5. Installa le dipendenze per il frontend ed effettua il build del progetto:

   cd main/frontend

   npm install

   npm run build

6. Effettuare il run del progetto con il seguente comando (dalla cartella main):

   flask run

QUERY IMPLEMENTATE PER CRUD E ANALISI SUI DATI

In questa sezione vi è una descrizione dettagliata delle query implementate.

Operazioni CRUD

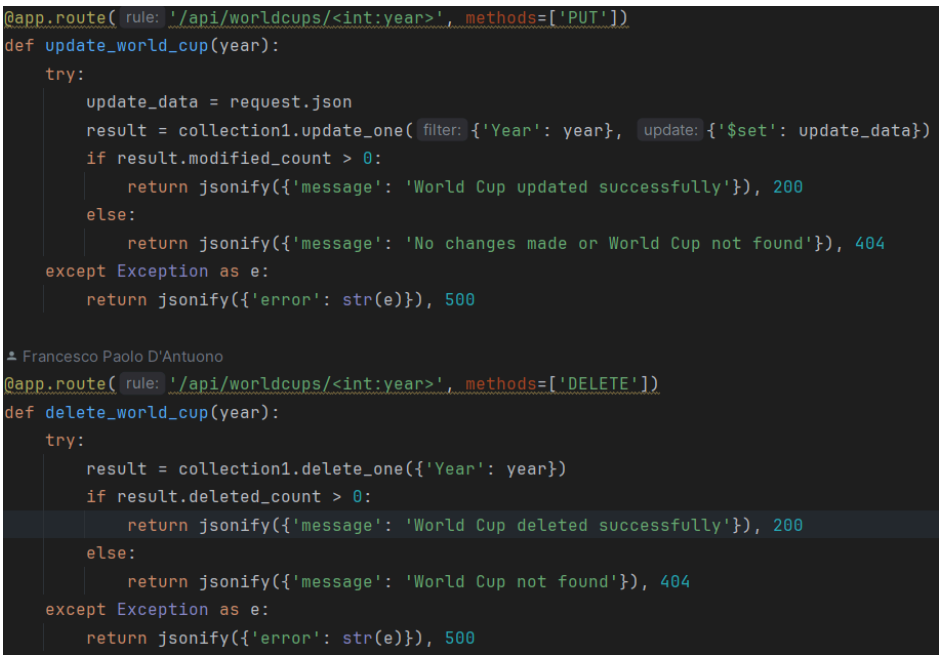
Le operazioni CRUD sono state implementate a scopo didattico su una singola collection (world_cups.csv).

Vediamo come sono state implementate le operazioni CRUD.

```python
@app.route( rule: '/api/worldcups', methods=['POST'])
def create_world_cup():
    try:
        cup_data = request.json
        collection1.insert_one(cup_data)
        return jsonify({'message': 'World Cup added successfully'}), 201
    except Exception as e:
        return jsonify({'error': str(e)}), 400

Francesco Paolo D'Antuono
@app.route( rule: '/api/worldcups', methods=['GET'])
def read_world_cups():
    try:
        cups = list(collection1.find( *args: {}, {'_id': 0}).sort([('Year', pymongo.ASCENDING)]))
        return jsonify(cups), 200
    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

In questa immagine, vengono mostrate le operazioni di create, che permette di creare una nuova istanza di coppa del mondo nel db, e di read, che consente di leggere tutte le coppe del mondo presenti in quella specifica collection e visualizzarle in una tabella. nella pagina specifica della web application.



```python
@app.route( rule: '/api/worldcups/<int:year>', methods=['PUT'])
def update_world_cup(year):
    try:
        update_data = request.json
        result = collection1.update_one( filter: {'Year': year}, update: {'$set': update_data})
        if result.modified_count > 0:
            return jsonify({'message': 'World Cup updated successfully'}), 200
        else:
            return jsonify({'message': 'No changes made or World Cup not found'}), 404
    except Exception as e:
        return jsonify({'error': str(e)}), 500

Francesco Paolo D'Antuono
@app.route( rule: '/api/worldcups/<int:year>', methods=['DELETE'])
def delete_world_cup(year):
    try:
        result = collection1.delete_one({'Year': year})
        if result.deleted_count > 0:
            return jsonify({'message': 'World Cup deleted successfully'}), 200
        else:
            return jsonify({'message': 'World Cup not found'}), 404
    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

In quest'altra immagine, invece, vengono mostrate le operazioni di update, che consente di aggiornare i campi inerenti ad una specifica coppa del mondo, e delete, che consente di eliminare una specifica coppa del mondo dal db.

Paesi vincitori della world cup e numero di world cup vinte

Questa query è stata realizzata per riuscire ad avere informazioni sui paesi vincitori della world cup e il numero di world cup che ogni paese ha vinto. Questi dati verranno visualizzati nel sito in una mappa interattiva, dove ogni paese che ha vinto la coppa del mondo è colorato con una gradazione di verde distinta. La gradazione diversa di colore serve ad indicare chi ha vinto più world cup (verde scuro indica il numero più grande, cioé 5, mentre verde chiaro indica il numero più basso, cioé 1).

Nella mappa vi è anche la possibilità di visualizzare il numero di world cup vinte andando a cliccare sul paese specifico. Questa query ci ha permesso di identificare il Brasile come paese più titolato al mondo. Vediamo come è stata realizzata con la seguente immagine.

```python
@app.route( rule: '/api/worldcupwins', methods=['GET'])
def get_all_world_cup_wins():
    try:
        pipeline = [
            {
                "$group": {
                    "_id": "$Winner",
                    "wins": {"$sum": 1}
                }
            },
            {
                "$project": {
                    "_id": 0,
                    "country": "$_id",
                    "wins": 1
                }
            }
        ]
        results = list(collection1.aggregate(pipeline))
        return jsonify(results), 200
    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

La query effettua il raggruppamento dei dati sul campo ‘Winner’ e conta il numero di vittorie di world cup. Dopo aver effettuato questo calcolo con il comando “$group”, si è proceduto a rimodellare l’output della query con “$project” per fornire le informazioni richieste nel formato corretto.

Numero di partite giocate per anno in ogni world cup

Questa query è stata realizzata per visualizzare l’andamento del numero di partite nel corso degli anni. Dunque quello che si è ricavato è il numero di partite giocate per ogni anno e il risultato viene visualizzato nel sistema con un line plot che ci permette di visualizzare correttamente le informazioni come cambiano col passare del tempo.

Dai risultati ottenuti, si può dedurre che, col passare del tempo, la tendenza del numero di partite è crescente, quindi più si avanza con gli anni e più il numero di partite crescerà. Ciò porta più spettacolarità e introiti, ma può portare anche ad un incremento di infortuni e quindi prima o poi l'aumento del numero di partite diventerà un problema serio da affrontare. Infatti, è già stato annunciato che nel 2026 il numero di partite della coppa del mondo che si disputerà in America-Messico-Canada avrà 104 partite.

Vediamo nel dettaglio come è stata realizzata la query.

```python
@app.route( rule: '/api/matchesplayed', methods=['GET'])
def get_matches_played_per_year():
    try:
        pipeline = [
            {
                "$project": {
                    "_id": 0,
                    "Year": 1,
                    "Matches Played": 1
                }
            },
            {
                "$sort": {"Year": 1}
            }
        ]
        results = list(collection1.aggregate(pipeline))
        return jsonify(results), 200
    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

Qui, vengono ordinati i risultati per anno con il comando “$sort”, per poi formattare con “$project” l’output con le informazioni richieste.

Tasso win/lose di ogni nazione per tutte le world cup

In questa query si vuole analizzare il tasso di vittorie, pareggi e sconfitte di un paese per tutte le coppe del mondo disputate. Ciò è stato realizzato per capire quali nazioni hanno vinto più partite e quali ne hanno perse di più.

Nella web app i risultati, non appena viene scelta la nazione specifica, vengono visualizzati attraverso un grafico a torta, che mostra la percentuale di vittorie, pareggi e sconfitte, rispettivamente colorate in verde, grigio e rosso.

Vediamo come è stata realizzata la query nelle seguenti immagini.

```python
@app.route( rule: '/api/teamstats', methods=['GET'])
def get_team_stats():
    try:
        pipeline = [
            {
                "$project": {
                    "home_team": "$Home Team",
                    "away_team": "$Away Team",
                    "home_goals": "$Home Goals",
                    "away_goals": "$Away Goals"
                }
            },
            {
                "$facet": {
                    "home_stats": [
                        {
                            "$group": {
                                "_id": "$home_team",
                                "wins": {
                                    "$sum": {
                                        "$cond": [{"$gt": ["$home_goals", "$away_goals"]}, 1, 0]
                                    }
                                },
                                "draws": {
                                    "$sum": {
                                        "$cond": [{"$eq": ["$home_goals", "$away_goals"]}, 1, 0]
                                    }
                                },
                                "losses": {
                                    "$sum": {
                                        "$cond": [{"$lt": ["$home_goals", "$away_goals"]}, 1, 0]
                                    }
                                },
                                "total_games": {"$sum": 1}
                            }
                        }
                    ],
                    "away_stats": [
                        {
                            "$group": {
                                "_id": "$away_team",
                                "wins": {
                                    "$sum": {
                                        "$cond": [{"$gt": ["$away_goals", "$home_goals"]}, 1, 0]
                                    }
                                },
                                "draws": {
                                    "$sum": {
                                        "$cond": [{"$eq": ["$away_goals", "$home_goals"]}, 1, 0]
                                    }
                                },
                                "losses": {
                                    "$sum": {
                                        "$cond": [{"$lt": ["$away_goals", "$home_goals"]}, 1, 0]
                                    }
                                },
                                "total_games": {"$sum": 1}
                            }
                        }
```

```
                    }
                ]
            }
        },
        {
            "$project": {
                "all_stats": {
                    "$concatArrays": ["$home_stats", "$away_stats"]
                }
            }
        },
        {
            "$unwind": "$all_stats"
        },
        {
            "$group": {
                "_id": "$all_stats._id",
                "wins": {"$sum": "$all_stats.wins"},
                "draws": {"$sum": "$all_stats.draws"},
                "losses": {"$sum": "$all_stats.losses"},
                "total_games": {"$sum": "$all_stats.total_games"}
            }
        },
        {
            "$project": {
                "_id": 0,
                "team": "$_id",
                "wins": 1,
                "draws": 1,
```

```
                "losses": 1,
                "total_games": 1,
                "win_percentage": {"$multiply": [{"$divide": ["$wins", "$total_games"]}, 100]},
                "draw_percentage": {"$multiply": [{"$divide": ["$draws", "$total_games"]}, 100]},
                "lose_percentage": {"$multiply": [{"$divide": ["$losses", "$total_games"]}, 100]}
            }
        }
    ]
    results = list(collection2.aggregate(pipeline))
    return jsonify(results), 200
except Exception as e:
    return jsonify({'error': str(e)}), 500
```

In questa query, innanzitutto vengono estratti i campi dalla collection “world_cup_matches.csv” utili per la query che sono Home Team, Away Team, Home Goals, Away Goals, e li rinomina in home_team, away_team, home_goals e away_goals rispettivamente. Poi viene effettuata un’operazione di “$facet” dove vengono create due pipeline separate, home_stats e away_stats, per raggruppare i risultati rispettivamente con “$group” in base alla squadra di casa (home_team) e alla squadra in trasferta (away_team) calcolando per entrambi vittorie, pareggi, sconfitte e il totale delle partite giocate.

Vengono poi concatenate le due statistiche utilizzando “$concatArrays” e il risultato viene salvato nell’array all_stats del quale verrà effettuato il “$unwind” per distendere le informazioni su un documento. Viene effettuato un ulteriore raggruppamento per sommare il numero di vittorie, pareggi, sconfitte e numero di partite giocate aggregando i dati sia per le partite in casa che in trasferta. Infine, viene formattato l’output con la “$project” finale per avere le informazioni richieste nella quale viene effettuato il calcolo della percentuale, in base ai dati che sono stati calcolati in precedenza.

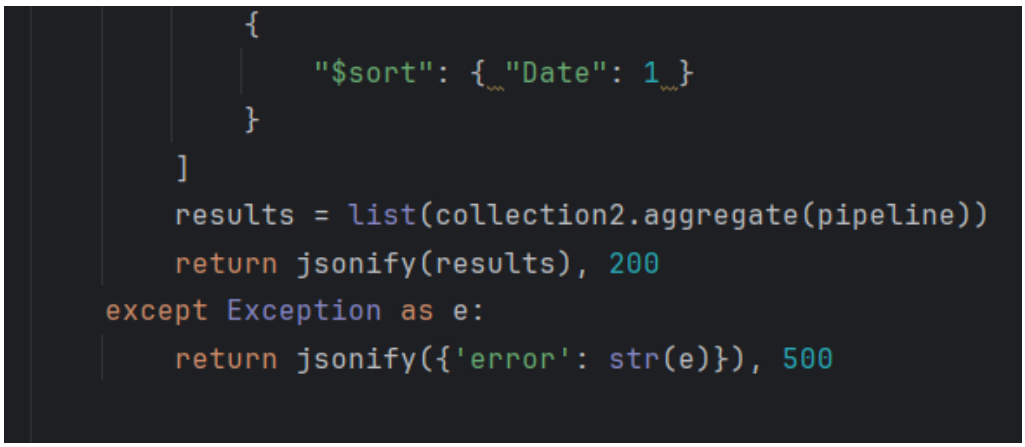
Scontri diretti tra due nazioni

Questa query permette di prelevare, dopo aver inserito le due nazioni nel sistema dalla web application, gli scontri diretti tra le due nazioni, cioè gli scontri avvenuti nel corso degli anni tra due nazioni specifiche. nella web application dunque basterà inserire le due nazioni e di conseguenza i risultati verranno mostrati in una tabella. Questa query è stata realizzata per dare una cronologia delle partite tra due nazioni e capire chi è in “vantaggio” negli scontri diretti.

Vediamo nel dettaglio come è stata fatta la query.

```
def get_versus_matches():
    try:
        nation1 = request.args.get('nation1')
        nation2 = request.args.get('nation2')
        if not nation1 or not nation2:
            return jsonify({'error': 'Two nations must be specified'}), 400

        pipeline = [
            {
                "$match": {
                    "$or": [
                        {"$and": [{"Home Team": nation1}, {"Away Team": nation2}]},
                        {"$and": [{"Home Team": nation2}, {"Away Team": nation1}]}
                    ]
                }
            },
            {
                "$project": {
                    "_id": 0,
                    "Date": 1,
                    "Stage": 1,
                    "HomeTeam": "$Home Team",
                    "AwayTeam": "$Away Team",
                    "HomeGoals": "$Home Goals",
                    "AwayGoals": "$Away Goals"
                }
            },
```



```
            {
                "$sort": { "Date": 1 }
            }
        ]
        results = list(collection2.aggregate(pipeline))
        return jsonify(results), 200
    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

In questa query, viene utilizzato il comando “$match” che filtra i documenti in base a una combinazione di due possibili scenari:

- La nazione nation1 gioca in casa e nation2 gioca in trasferta.
- La nazione nation2 gioca in casa e nation1 gioca in trasferta.

Poi è stata effettuata l’operazione di “$project” per riorganizzare l’output, in modo tale da avere le informazioni specifiche che si vogliono ottenere dalla query, ordinando i risultati per il campo (Date) con il comando “$sort” in ordine crescente.

Numero di gol totali di una nazione per ogni anno

In questa query si vuole analizzare il numero di gol segnati da una nazione nel corso delle varie coppe del mondo, per capire se nel corso degli anni c’è stato un incremento o un decremento del numero di gol segnati, che potrebbe indicare un cambiamento di stile di gioco.

Nella web application, i risultati sono stati visualizzati in un istogramma per evidenziare in maniera immediata la distribuzione dei dati e il loro andamento.

Vediamo nel dettaglio la realizzazione di questa query.

```python
@app.route( rule: '/api/goalstats', methods=['GET'])
def get_goal_stats():
    try:
        pipeline = [
            {
                "$group": {
                    "_id": {
                        "year": "$Year",
                        "team": "$Home Team"
                    },
                    "total_goals": {"$sum": "$Home Goals"}
                }
            },
            {
                "$unionWith": {
                    "coll": "world_cup_matches",
                    "pipeline": [
                        {
                            "$group": {
                                "_id": {
                                    "year": "$Year",
                                    "team": "$Away Team"
                                },
                                "total_goals": {"$sum": "$Away Goals"}
                            }
                        }
                    ]
                }
            },
            {
                "$group": {
                    "_id": {
                        "year": "$_id.year",
                        "team": "$_id.team"
                    },
                    "total_goals": {"$sum": "$total_goals"}
                }
            },
            {
                "$project": {
                    "_id": 0,
                    "year": "$_id.year",
                    "team": "$_id.team",
                    "total_goals": 1
                }
            },
            {
                "$sort": {"year": 1}
            }
        ]
        results = list(collection2.aggregate(pipeline))
        return jsonify(results), 200
    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

Come si può notare dalle immagini, il primo comando è “$group” che, in questo caso, raggruppa i documenti in base all'anno (Year) e alla squadra di casa (Home Team) e calcola il totale dei gol segnati dalla squadra di casa (Home Goals). Poi verranno calcolati il totale dei gol segnati della squadra in trasferta con una sotto pipeline e uniti i risultati con la pipeline principale con il comando “$unionWith”. Infine, dopo aver raggruppato ulteriormente i risultati per squadra e per anno, viene formattato l’output con le informazioni richieste dalla query, ordinandole per anno in maniera crescente.

Media gol fatti e subiti dei vincitori delle world cup

In questa query si vuole andare a visualizzare, per ogni vincitore della world cup, la media di gol fatti e subiti in quello specifico torneo, per capire quale nazione ha avuto prestazioni migliori rispetto ad altre. Lo scopo principale della query era quello di unire le informazioni delle due collection differenti.

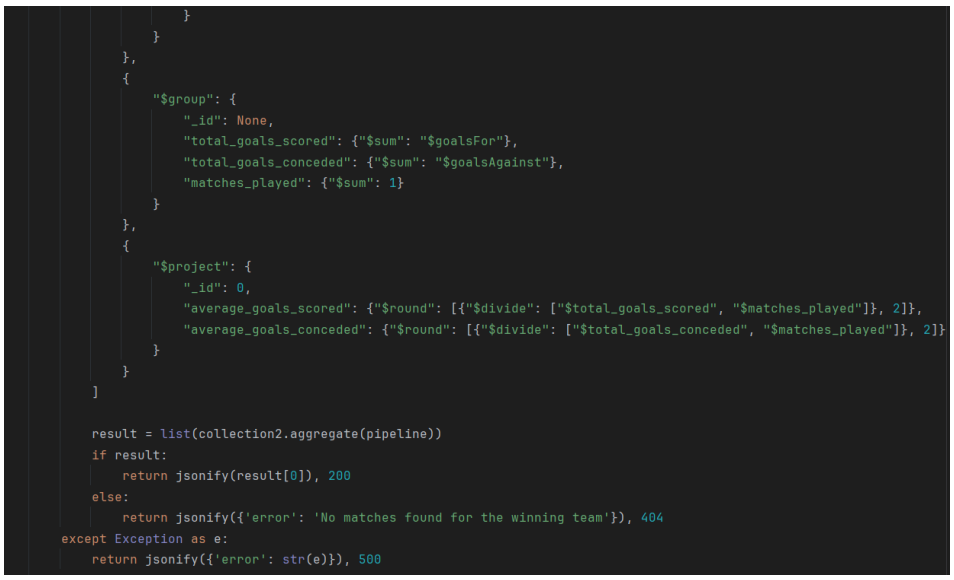
Nella web application, dopo aver inserito l’anno specifico che identifica la world cup, viene mostrato un grafico a torta, dove il colore verde identifica la media dei gol fatti e in rosso la media dei gol subiti.

Vediamo nel dettaglio la query.

```
@app.route( rule: '/api/winnerstats', methods=['GET'])
def get_winner_stats():
    try:
        year = request.args.get('year')
        if not year:
            return jsonify({'error': 'Year must be specified'}), 400

        year = int(year)
        world_cup = collection1.find_one({"Year": year})
        if not world_cup:
            return jsonify({'error': 'No World Cup data found for the specified year'}), 404

        winner = world_cup['Winner']
        pipeline = [
            {
                "$match": {
                    "Year": year,
                    "$or": [
                        {"Home Team": winner},
                        {"Away Team": winner}
                    ]
                }
            },
            {
                "$project": {
                    "goalsFor": {
                        "$cond": [{"$eq": ["$Home Team", winner]}, "$Home Goals", "$Away Goals"]
                    },
                    "goalsAgainst": {
                        "$cond": [{"$eq": ["$Home Team", winner]}, "$Away Goals", "$Home Goals"]
```



```
                    }
                }
            },
            {
                "$group": {
                    "_id": None,
                    "total_goals_scored": {"$sum": "$goalsFor"},
                    "total_goals_conceded": {"$sum": "$goalsAgainst"},
                    "matches_played": {"$sum": 1}
                }
            },
            {
                "$project": {
                    "_id": 0,
                    "average_goals_scored": {"$round": [{"$divide": ["$total_goals_scored", "$matches_played"]}, 2]},
                    "average_goals_conceded": {"$round": [{"$divide": ["$total_goals_conceded", "$matches_played"]}, 2]}
                }
            }
        ]

        result = list(collection2.aggregate(pipeline))
        if result:
            return jsonify(result[0]), 200
        else:
            return jsonify({'error': 'No matches found for the winning team'}), 404
    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

In questa query, dopo aver preso dalla richiesta l'anno scelto dall'utente, si va a prelevare il vincitore di quell'anno specifico, che ci servirà come informazione nella pipeline. La prima operazione effettuata è "$match" che ci ha consentito di filtrare i documenti per anno e per la squadra vincitrice. Poi è stata utilizzata "$project" per creare due nuovi campi:

- goalsFor: I gol segnati dalla squadra vincitrice (se la squadra vincitrice è la squadra di casa, usa Home Goals, altrimenti usa Away Goals).
- goalsAgainst: I gol subiti dalla squadra vincitrice (se la squadra vincitrice è la squadra di casa, usa Away Goals, altrimenti usa Home Goals).

Dopo ciò, verrà effettuata l'operazione "$group", che aggrega i dati calcolando il totale dei gol segnati (total_goals_scored), il totale dei gol subiti (total_goals_conceded) e il numero di partite giocate (matches_played). Infine verrà effettuata l'operazione "$project", che riformatta l'output calcolando la media dei gol segnati (average_goals_scored) e la media dei gol subiti (average_goals_conceded), arrotondando ciascuno a due decimali.