UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Progetto di Basi di Dati II

MotoGP championship simulator

0522501917 - Giovanni Manfredi

Anno Accademico 2024/2025

Indice

1 Introduzione

Il presente documento illustra il progetto universitario sviluppato per il corso "**Basi di Dati II**", tenutosi nell'**anno accademico 2024/2025** presso l'*Università degli Studi di Salerno*.

1.1 Prefazione

La **MotoGP** rappresenta la massima espressione del motociclismo sportivo su pista. Le sue origini risalgono al 1949, anno in cui la **FIM** (Fédération Internationale de Motocyclisme) istituì il primo Campionato Mondiale. Nel corso dei decenni, questa disciplina ha vissuto una profonda evoluzione, sia sul piano tecnologico sia in termini di visibilità globale. Con l'introduzione dei motori a quattro tempi nel 2002 e il costante adeguamento dei regolamenti tecnici e sportivi, la MotoGP si è affermata come uno spettacolo adrenalinico seguito da milioni di appassionati in tutto il mondo.

La recente digitalizzazione e la crescente disponibilità di dati dettagliati hanno aperto nuove frontiere per l'analisi delle performance, abilitando la creazione di applicazioni interattive e simulatori basati su tali informazioni. Di conseguenza, l'interesse per la MotoGP trascende il singolo evento sportivo, estendendosi a strumenti che ne consentono una fruizione più coinvolgente e personalizzata, come simulatori di gara, giochi strategici o piattaforme di data analysis.

1.2 Obiettivo

L'obiettivo del progetto è sviluppare una **piattaforma web interattiva** che consenta agli utenti di **creare e gestire campionati personalizzati** di MotoGP. Il sistema permette di simulare gare realistiche, basandosi su statistiche storiche dettagliate, offrendo la possibilità di organizzare nuovi campionati mondiali in cui far competere i piloti preferiti sui circuiti selezionati. I risultati delle gare vengono generati automaticamente tramite un **algoritmo predittivo avanzato**, basato su tecniche di *Machine Learning*, che garantisce simulazioni accurate e coinvolgenti.

1.3 La sfida

Una delle principali sfide affrontate nel progetto è stata la gestione e l'elaborazione di dataset storici eterogenei, che coprono un arco temporale molto ampio (dal 1949 al 2022). Inoltre, è stata necessaria la progettazione e l'integrazione di un'infrastruttura tecnologica completa, composta da un database NoSQL (in questo caso **MongoDB**) per la persistenza dei dati, **API REST** sviluppate con Flask per la logica di business e un'interfaccia utente moderna e responsive realizzata con **React** e **Tailwind CSS**.

Il modello predittivo L'utilizzo di modelli predittivi, pur opzionale, rappresenta un elemento distintivo del progetto, poiché conferisce alla simulazione un elevato grado di realismo e personalizzazione, posizionando la piattaforma come uno strumento con triplice valenza: didattica, tecnica e ludica.

1.4 Specifica P.E.A.S.

Di seguito viene presentata la specifica P.E.A.S. (Performance, Environment, Actuators, Sensors) relativa al sistema realizzato.

Performance	Predizione dei risultati di una gara all'interno di un campionato personalizzato, costituito da piloti e circuiti selezionati dall'utente.
Environment	Ambiente rappresentato dalla web-app MotoGP championship simulator.
Actuators	Dispositivi di output, quali il monitor, utilizzati per visualizzare i risultati delle predizioni.
Sensors	Dispositivi di input, come il puntatore del mouse o il touch screen, tramite i quali l'utente seleziona piloti e circuiti.

Table 1: Specifica P.E.A.S. del simulatore

1.4.1 Proprietà dell'ambiente

L'ambiente della simulazione presenta le seguenti caratteristiche principali:

1. **Completamente osservabile**: l'agente (algoritmo predittivo) dispone di tutte le informazioni necessarie — quali statistiche, dati dei piloti, dati dei tracciati e risultati — per effettuare la simulazione.

2. **Deterministico**: a parità di input, il sistema produce lo stesso output simulato, risultando quindi deterministico.

3. **Episodico**: ogni simulazione di gara è considerata come un episodio indipendente.

4. **Statico**: l'ambiente rimane invariato durante l'interazione dell'utente con il sistema e durante l'elaborazione della simulazione da parte del modello.

5. **Discreto**: le azioni dell'agente e gli eventi (come l'aggiunta di nuovi campionati, la simulazione di una gara o l'aggiornamento della classifica) sono distinti e ben definiti.

6. **Singolo agente**: il sistema comprende un unico agente decisionale, rappresentato dal simulatore, senza la presenza di agenti indipendenti che influenzino direttamente l'ambiente.

2 Descrizione del sistema

Il sistema è stato progettato con l'obiettivo di fornire una **piattaforma web interattiva** per la simulazione e la gestione personalizzata di campionati MotoGP, facilitando l'organizzazione delle gare e la consultazione dei risultati.

Il progetto prevede lo sviluppo di un'applicazione web full-stack, progettata per offrire un'interfaccia grafica moderna e intuitiva, attraverso la quale l'utente può accedere a tutte le funzionalità del sistema in modo semplice ed efficace. La piattaforma si basa sull'impiego del dataset pubblico "**MotoGP World Championship (1949–2022)**", disponibile su Kaggle, con l'obiettivo di abilitare la creazione di campionati personalizzati partendo da dati storici reali, e di generare risultati verosimili mediante l'uso di algoritmi predittivi.

L'architettura del sistema adotta un modello client/server: il backend, sviluppato in **Flask**, si occupa della logica applicativa e della gestione dei dati, mentre il frontend, realizzato con **React** e **Tailwind CSS**, cura la presentazione e l'esperienza utente.

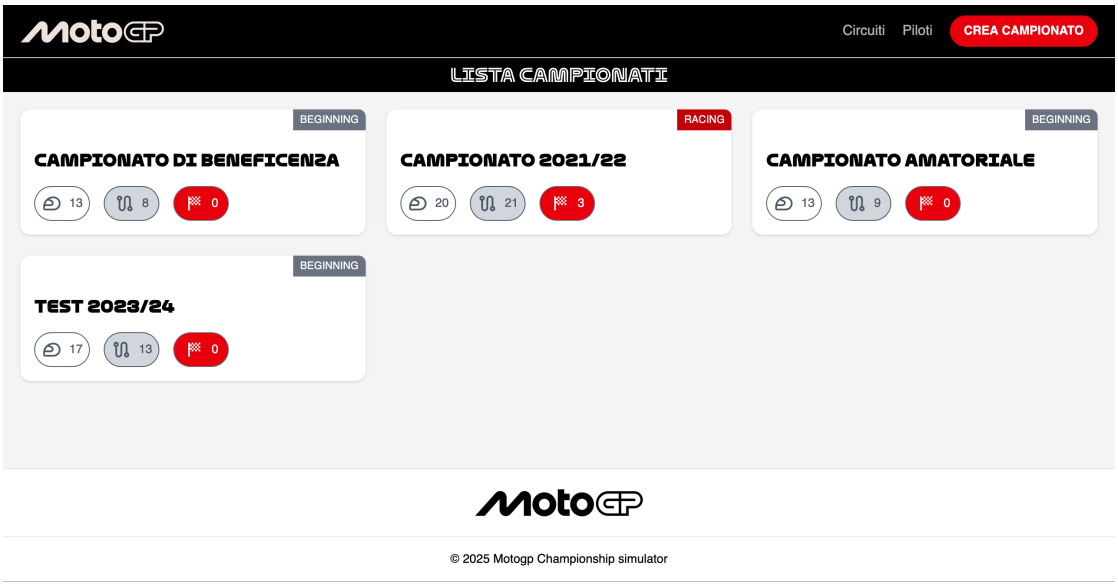


Figure 1: Homepage di MotoGP championship simulator

Le principali funzionalità offerte dalla piattaforma includono:

1. **Creazione**, **modifica** ed **eliminazione** di campionati personalizzati;
2. **Gestione** dei piloti associati a ciascun campionato;
3. **Simulazione** dei risultati di gara mediante un algoritmo predittivo basato su dati storici;
4. **Aggiornamento automatico** della classifica piloti in base ai risultati delle singole gare;
5. **Visualizzazione** dettagliata di piloti, circuiti e campionati;
6. **Rimozione** di gare già disputate con conseguente ricalcolo della classifica.

2.1 Le operazioni CRUD

Il sistema implementa in modo completo le **operazioni CRUD** (*Create, Read, Update, Delete*) sulle principali entità gestite. In particolare:

- **Creazione** di nuovi campionati personalizzati;
- **Lettura** dei dati relativi a piloti, circuiti e campionati;
- **Aggiornamento** delle informazioni relative a gare, piloti o composizione dei campionati;
- **Eliminazione** di campionati o gare già disputate, con conseguente ricalcolo della classifica.

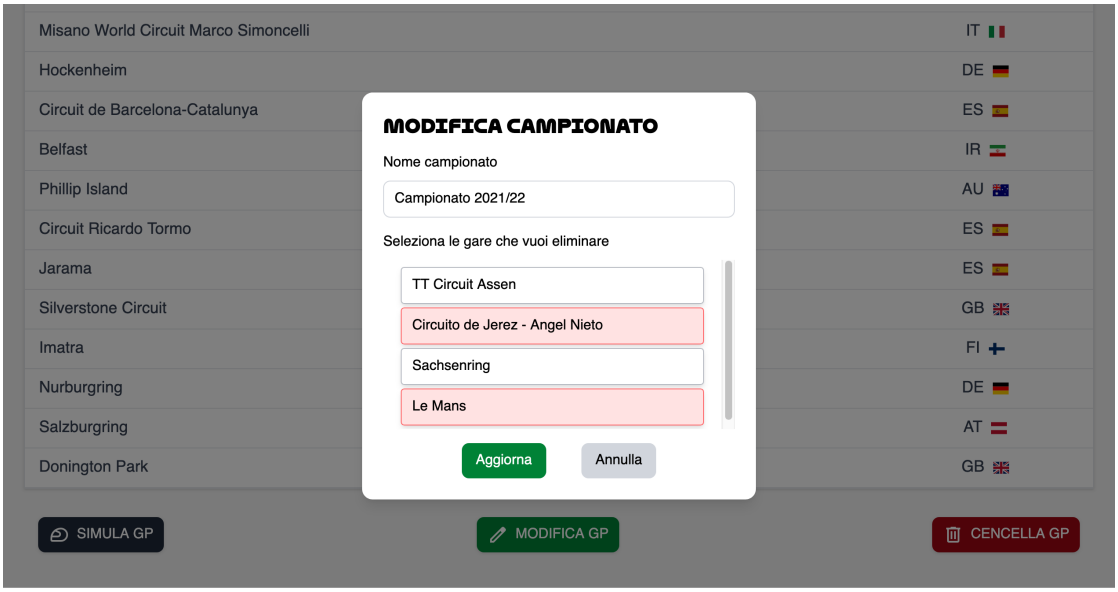


Figure 2: Cancellazione di un nuovo campionato

2.2 Stack tecnologico

Il sistema è stato sviluppato utilizzando un moderno stack tecnologico full-stack, che integra strumenti e librerie efficienti sia per il frontend che per il backend. In particolare:

- **Frontend**: *React* con *Vite* per il bundling e *Tailwind CSS* per la gestione dello stile;
- **Backend**: *Flask*, framework Python leggero e flessibile per lo sviluppo delle API REST;
- **Database**: *MongoDB*, database NoSQL orientato ai documenti, ideale per la gestione dinamica dei dati;
- **Machine Learning**: *Scikit-learn*, libreria Python utilizzata per l'implementazione dell'algoritmo predittivo.

3 Il dataset

Come già anticipato, per la realizzazione della piattaforma è stato utilizzato il dataset pubblico "**MotoGP World Championship (1949–2022)**", disponibile su Kaggle. Esso include dati relativi a gare, piloti, campionati, team e circuiti, coprendo un arco temporale di oltre 70 anni.

3.1 Struttura originale del dataset

Il dataset utilizzato per l'alimentazione del sistema è composto da sei file principali in formato CSV, ognuno dei quali contiene informazioni storiche e statistiche legate al Campionato Mondiale MotoGP.

Di seguito una descrizione sintetica di ciascun file:

- `constructure-world-championship.csv`: contiene i costruttori vincitori del titolo mondiale per ciascuna stagione e classe;
- `grand-prix-events-held.csv`: riporta il numero di edizioni disputate per ciascun circuito e paese, evidenziando i tracciati con maggiore storicità;
- `grand-prix-race-winners.csv`: elenca i vincitori di ogni gara, specificando circuito, classe, costruttore, nazionalità e pilota per ogni stagione;
- `riders-finishing-positions.csv`: raccoglie le posizioni finali ottenute da ciascun pilota nel corso della carriera, dai primi sei piazzamenti;
- `riders-info.csv`: fornisce una panoramica complessiva sui piloti più titolati, riportando dati su vittorie, podi, pole position, giri veloci e titoli mondiali;
- `same-nation-podium-lockouts.csv`: documenta le gare in cui tutti e tre i piloti sul podio appartenevano alla stessa nazione.

Pulizia e normalizzazione dei dati

L'elaborazione dei dati è stata effettuata a partire da tre file di input principali:

- `riders-finishing-positions.csv`;
- `riders-info.csv`;
- `grand-prix-events-held.csv`.

Attraverso uno script Python appositamente sviluppato, i dataset sono stati sottoposti a un processo di **pulizia** e **normalizzazione**, che ha previsto le seguenti operazioni principali:

- **rinominazione** delle colonne per garantirne uniformità e chiarezza;
- standardizzazione della **formattazione** dei nomi dei piloti, invertendo ordine e capitalizzazione;
- **rimozione** di colonne ridondanti o non rilevanti ai fini dell'analisi;
- **integrazione** dei dati provenienti da fonti diverse per ottenere profili pilota completi e coerenti.

In particolare, per ciascun pilota è stato calcolato un punteggio sintetico **Score** basato su una formula pesata che attribuisce un diverso peso alle statistiche di vittorie ($\times 4$), secondi posti ($\times 2$), terzi posti ($\times 1$), pole position ($\times 1.5$) e titoli mondiali ($\times 6$). Tale punteggio è stato quindi normalizzato dividendo ogni valore per il massimo osservato nel dataset, ottenendo così un **NormalizedScore** compreso nell'intervallo da 0 a 1, utile per facilitare il confronto e l'inserimento nei modelli predittivi; per quanto riguarda i dati relativi ai circuiti, sono state mantenute solo le informazioni essenziali, semplificando la struttura del dataset.

I file di output risultanti, puliti e pronti per l'integrazione nel backend, sono:

- `rider.csv`, contenente i profili pilota completi con le statistiche aggregate e il punteggio normalizzato;

rider

	_id ObjectId	Name String	First Places Int32	Second Places Double	Third Places
1	ObjectId('684b5ec6d7a7d00…	"Agostini Giacomo"	122	35	2
2	ObjectId('684b5ec6d7a7d00…	"Rossi Valentino"	115	67	53
3	ObjectId('684b5ec6d7a7d00…	"Nieto Angel"	90	35	14
4	ObjectId('684b5ec6d7a7d00…	"Marquez Marc"	85	36	17
5	ObjectId('684b5ec6d7a7d00…	"Hailwood Mike"	76	25	11
6	ObjectId('684b5ec6d7a7d00…	"Lorenzo Jorge"	68	51	33
7	ObjectId('684b5ec6d7a7d00…	"Pedrosa Dani"	54	52	47
8	ObjectId('684b5ec6d7a7d00…	"Doohan Mick"	54	31	10
9	ObjectId('684b5ec6d7a7d00…	"Read Phil"	52	44	25
10	ObjectId('684b5ec6d7a7d00…	"Redman Jim"	45	33	20
11	ObjectId('684b5ec6d7a7d00…	"Stoner Casey"	45	17	27
12	ObjectId('684b5ec6d7a7d00…	"Biaggi Max"	42	41	28
13	ObjectId('684b5ec6d7a7d00…	"Mang Anton"	42	25	17
14	ObjectId('684b5ec6d7a7d00…	"Ubbiali Carlo"	39	20	9
15	ObjectId('684b5ec6d7a7d00…	"Surtees John"	38	4	3

Figure 3: rider.csv

- `track.csv`, contenente le informazioni essenziali sui circuiti e gli eventi.

track

	_id ObjectId	Track String	Country String
1	ObjectId('684b5ec7d7a7d00…	"TT Circuit Assen"	"NL"
2	ObjectId('684b5ec7d7a7d00…	"Automotodrom Brno"	"CZ"
3	ObjectId('684b5ec7d7a7d00…	"Spa-Francorchamps"	"BE"
4	ObjectId('684b5ec7d7a7d00…	"Sachsenring"	"DE"
5	ObjectId('684b5ec7d7a7d00…	"Circuito de Jerez - Ange…	"ES"
6	ObjectId('684b5ec7d7a7d00…	"Autodromo Internazionale…	"IT"
7	ObjectId('684b5ec7d7a7d00…	"Le Mans"	"FR"
8	ObjectId('684b5ec7d7a7d00…	"Isle of Man"	"GB"
9	ObjectId('684b5ec7d7a7d00…	"Monza"	"IT"
10	ObjectId('684b5ec7d7a7d00…	"Misano World Circuit Mar…	"IT"
11	ObjectId('684b5ec7d7a7d00…	"Hockenheim"	"DE"
12	ObjectId('684b5ec7d7a7d00…	"Circuit de Barcelona-Cat…	"ES"
13	ObjectId('684b5ec7d7a7d00…	"Belfast"	"IR"
14	ObjectId('684b5ec7d7a7d00…	"Phillip Island"	"AU"
15	ObjectId('684b5ec7d7a7d00…	"Circuit Ricardo Tormo"	"ES"
16	ObjectId('684b5ec7d7a7d00…	"Jarama"	"ES"

Figure 4: track.csv

4 MongoDb e API

Il servizio è implementato utilizzando **Flask**, un micro-framework Python che espone un'API RESTful per l'interazione con il database MongoDB. La comunicazione tra frontend e backend avviene tramite chiamate HTTP, con il supporto di **CORS** (Cross-Origin Resource Sharing) per consentire richieste da domini differenti, come ad esempio "`http://127.0.0.1:5173`".

4.1 Endpoint API Principali

- `/api/create_championship` (**POST**): crea un nuovo campionato, richiedendo nome, piloti e tracciati.
- `/api/championship_list` (**GET**): restituisce l'elenco di tutti i campionati esistenti.
- `/api/championship/<id>` (**GET**): recupera i dettagli di un campionato specifico tramite il suo ID.
- `/api/championship/<id>` (**PUT**): aggiorna le informazioni di un campionato, permettendo anche l'eliminazione di gare associate.
- `/api/championship/<id>` (**DELETE**): elimina un campionato e tutte le gare ad esso collegate.
- `/api/rider` (**GET**): Fornisce un elenco di tutti i piloti con i loro dati statistici.
- `/api/get_participants_riders` (**GET**): restituisce i piloti che partecipano a un campionato specifico, utilizzando un'operazione di JOIN tra le collezioni.
- `/api/track` (**GET**): restituisce un elenco di tutti i tracciati disponibili.
- `/api/get_selected_tracks` (**GET**): fornisce i tracciati selezionati per un campionato specifico, anch'esso tramite un'operazione di JOIN.

- `/api/create_race` (**POST**): registra i risultati di una gara, associandola a un campionato esistente.
- `/api/get_races_championship` (**GET**): recupera tutte le gare associate a un dato campionato.
- `/api/championship/<championship_id>/simulate-race` (**POST**): simula una gara basata sui piloti e il tracciato forniti, salvando i risultati.

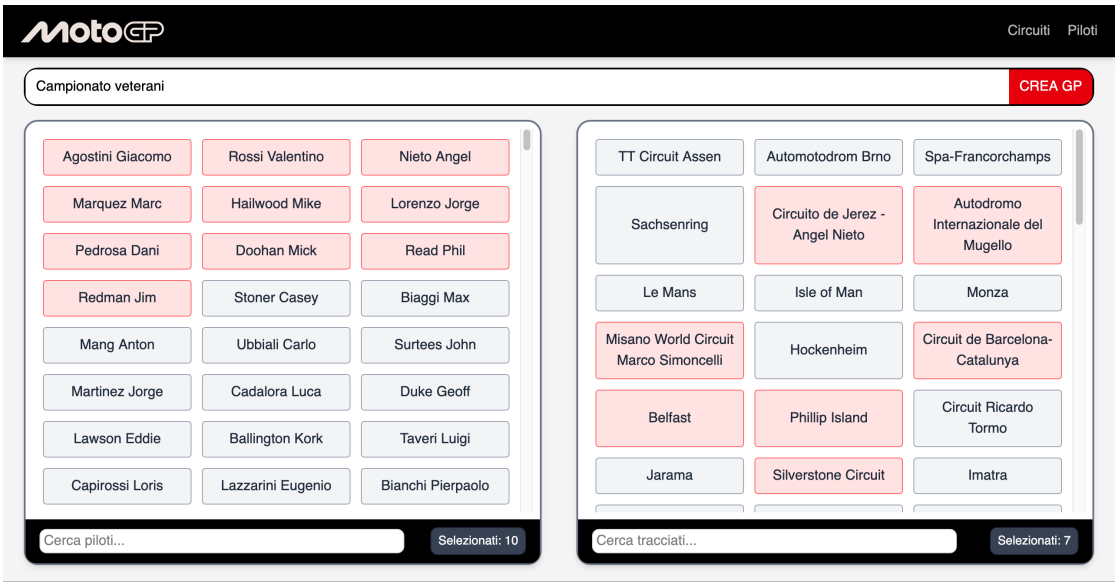


Figure 5: Creazione di un nuovo campionato

4.2 MongoDB

Per la gestione dei dati, la piattaforma utilizza **MongoDB**, un database NoSQL basato su documenti, ideale per strutture flessibili e ad alta scalabilità. Le informazioni relative ai campionati, piloti e tracciati sono organizzate in quattro principali *collection*:

- `championships`: contiene metadati e composizione dei campionati creati;

- `riders`: profili completi dei piloti, con l'aggiunta punteggi normalizzati e statistiche;
- `tracks`: elenco dei tracciati e relative informazioni geografiche;
- `race`: contiene i risultati dettagliati di ogni gara, collegati al campionato di appartenenza, con anche l'inclusione degli eventi in gara.

4.3 Operazione di JOIN tra le collection

Il servizio implementa operazioni di "JOIN" logico tra le collezioni di MongoDB per recuperare dati correlati. Sebbene MongoDB sia un database NoSQL non relazionale, offre meccanismi come `$lookup` per effettuare aggregazioni che simulano le operazioni di JOIN tipiche dei database relazionali. Questo permette di recuperare in un'unica query i dati di un campionato insieme ai dettagli dei piloti partecipanti (`/api/get_participants_riders`) o dei tracciati selezionati (`/api/get_selected_tracks`), migliorando l'efficienza nel recupero dei dati correlati.

ULTIMA GARA SACHSENRING

Posizione	Pilota	Tempo totale	Punti
1°	Duke Geoff	27:12.21	25
2°	Surtees John	27:17.11	20
3°	Doohan Mick	27:20.10	16
4°	Ballington Kork	27:38.37	13
5°	Agostini Giacomo	27:47.22	11
6°	Redman Jim	28:11.54	10
7°	Stoner Casey	28:13.15	9
8°	Lorenzo Jorge	28:16.43	8
9°	Read Phil	28:28.29	7
10°	Rossi Valentino	28:42.49	6
11°	Hailwood Mike	28:51.83	5
12°	Martinez Jorge	28:58.53	4
13°	Marquez Marc	29:00.42	3

Figure 6: Dettagli dell'ultima gara simulata

4.4 Il Teorema CAP e le proprietà BASE

Il **Teorema CAP** stabilisce che un sistema distribuito può garantire al massimo due delle seguenti tre proprietà fondamentali:

- **Consistency (Coerenza)**: ogni lettura restituisce l'ultima scrittura confermata;
- **Availability (Disponibilità)**: il sistema risponde sempre alle richieste;
- **Partition Tolerance (Tolleranza alle partizioni)**: il sistema continua a funzionare anche in caso di partizioni di rete.

MongoDB, configurato come replica set, per impostazione predefinita favorisce la **disponibilità** e la **tolleranza alle partizioni**, offrendo una coerenza *eventuale* nelle letture dai nodi secondari. Tuttavia, grazie a opzioni configurabili quali `writeConcern` e `readConcern`, è possibile orientare il sistema verso una coerenza più forte, adattandosi così alle esigenze specifiche del contesto applicativo.

Nel progetto, l'endpoint `/api/get_selected_tracks` esegue un'operazione di lettura su due collection, `championship` e `track`, tramite un'aggregazione MongoDB con l'operatore `$lookup`. Questo permette di realizzare una *JOIN* logica, simile a quella dei database relazionali, recuperando in un'unica query i dati del campionato e i relativi tracciati.

MongoDB adotta inoltre un modello **BASE** (Basically Available, Soft state, Eventual consistency), che implica:

- **Basically Available**: il sistema garantisce disponibilità per le operazioni, anche in presenza di guasti parziali;
- **Soft state**: lo stato del database può variare temporaneamente tra nodi prima di raggiungere la coerenza finale;
- **Eventual consistency**: i dati aggiornati vengono propagati tra i nodi e, nel tempo, tutte le repliche convergono verso lo stesso valore.

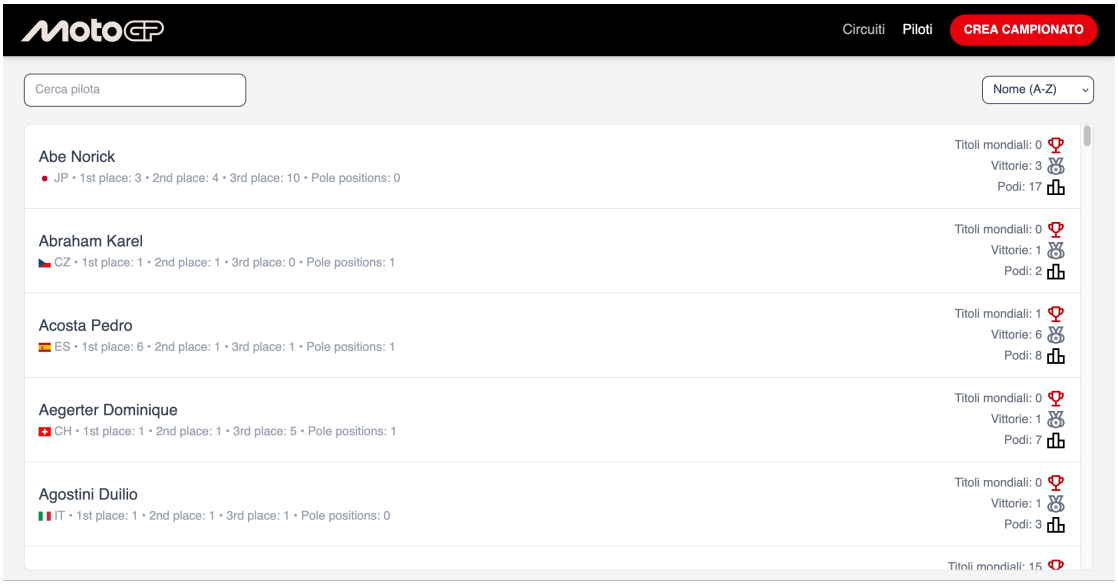


Figure 7: Visualizzazione dei piloti

Il progetto sfrutta queste proprietà per assicurare risposte rapide e una coerenza sufficiente in scenari distribuiti, bilanciando efficacemente affidabilità e prestazioni.

4.4.1 Proprietà rispettate

Il sistema può essere configurato per garantire le proprietà di **Consistency** e **Partition Tolerance** del Teorema CAP, a discapito della **Availability** in caso di partizioni di rete.

- **Coerenza (C)**: Configurando l'applicazione con `writeConcern: "majority"` (già adottato nelle operazioni di scrittura) e, opzionalmente, con `readConcern: "majority"`, si assicura che le letture riflettano solo dati replicati e confermati dalla maggioranza dei nodi. Questo garantisce una coerenza forte durante l'esecuzione di aggregazioni con `db.aggregate()`.

- **Tolleranza alle partizioni (P)**: MongoDB, essendo un sistema distribuito basato su replica set, continua a funzionare anche in presenza di partizioni di

rete tra i nodi, gestendo automaticamente il failover e mantenendo l'integrità e coerenza dei dati.

- **Disponibilità (A)**: Non viene garantita in caso di indisponibilità del nodo primario. In tali situazioni, MongoDB sospende temporaneamente le scritture fino all'elezione di un nuovo primario, privilegiando così la coerenza e la tolleranza alle partizioni rispetto alla disponibilità immediata.

Configurazioni consigliate Per evidenziare chiaramente questo comportamento, si consiglia di configurare esplicitamente:

- Scritture con writeConcern: `"majority"`;
- Letture con readConcern: `"majority"` (opzionale, ma raccomandato!)

5 Simulazione

La piattaforma integra un modulo di simulazione delle gare che consente di generare risultati realistici all'interno dei campionati personalizzati. Tale modulo sfrutta la libreria **scikit-learn** per implementare modelli predittivi che combinano dati storici e caratteristiche individuali dei piloti, fornendo una simulazione accurata e dinamica.

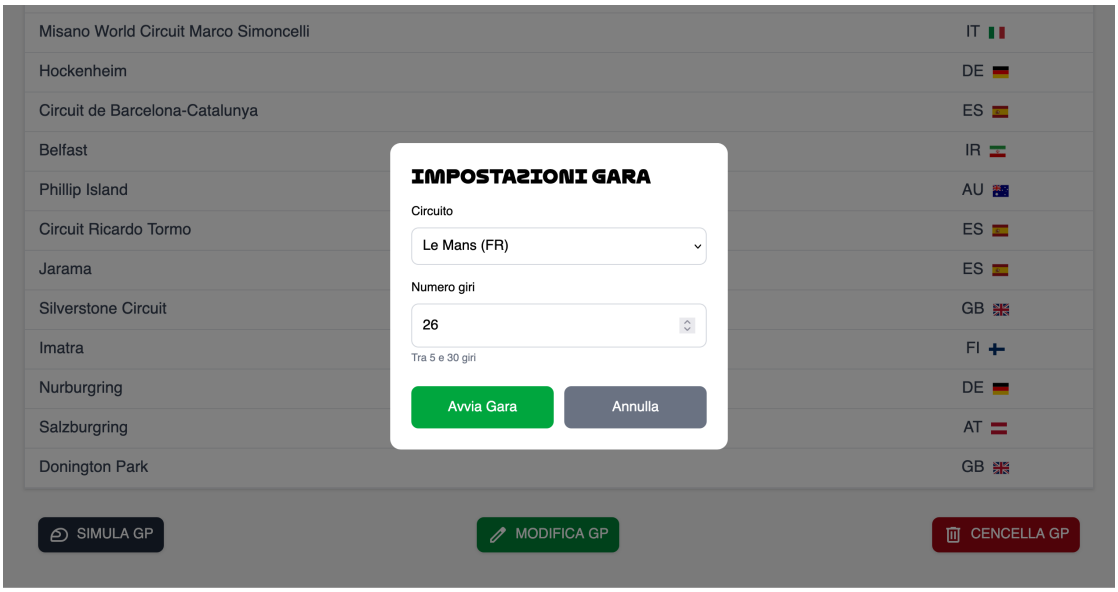


Figure 8: Sezione per la simulazione gara

5.1 Modello del pilota

Ogni pilota coinvolto nella simulazione è rappresentato dall'istanza della classe `PilotaGara`. Questa classe incapsula non solo informazioni anagrafiche e storiche, come il nome, la nazionalità e il punteggio normalizzato basato sulle prestazioni reali, ma anche una serie di parametri comportamentali cruciali che influenzano la dinamica della gara. Questi parametri sono:

- La **coerenza**: determina la stabilità dei tempi sul giro del pilota. Un valore più alto indica meno variabilità.

- La **resistenza sotto pressione**: influisce sulla capacità del pilota di mantenere prestazioni elevate in situazioni critiche, come la fase finale della gara o quando è in lotta per una posizione.
- Il **livello di affaticamento**: questo parametro, che può incrementare durante la gara, contribuisce a ridurre progressivamente l'efficacia del pilota nei giri successivi.
- La **propensione al rischio**: condiziona la probabilità che il pilota incorra in incidenti, cadute o penalità durante la competizione.

Questi fattori consentono di simulare fenomeni complessi come la variabilità dei tempi sul giro, l'insorgenza di penalità o la possibilità di ritiro, preservando la coerenza con le caratteristiche storiche e lo stile di guida di ciascun pilota. All'inizio della simulazione, le caratteristiche `consistency_score`, `pressure_resistance`, `fatigue_factor`, e `risk_profile` vengono assegnate ai piloti attraverso un algoritmo di clustering K-means, che raggruppa i piloti in categorie (es. conservativi, bilanciati, aggressivi) basandosi sul loro punteggio normalizzato e aggiungendo una componente casuale per maggiore realismo.

5.2 Funzionamento della simulazione

La classe `SimulatoreGara` orchesta l'intero processo di simulazione di una gara. Durante la simulazione, per ogni pilota vengono generati dinamicamente tempi parziali e totali giro per giro. L'algoritmo aggiorna continuamente la posizione in gara e lo stato di ciascun partecipante (ad esempio, se ha completato tutti i giri previsti o si è ritirato). L'integrazione delle variabili comportamentali e dei modelli predittivi permette di ottenere risultati che riflettono sia la qualità sportiva storica dei piloti sia l'imprevedibilità tipica delle competizioni reali.

La simulazione inizia con una griglia di partenza randomizzata ma influenzata dal punteggio normalizzato dei piloti. Ad ogni giro, il sistema calcola i tempi dei singoli piloti, verifica la probabilità di incidenti e aggiorna le posizioni, tenendo traccia di eventi significativi come i sorpassi o i ritiri.

I dati risultanti dalla simulazione vengono memorizzati nel database MongoDB e sono accessibili tramite l'interfaccia web della piattaforma, garantendo così un'esperienza interattiva e immersiva per l'utente.

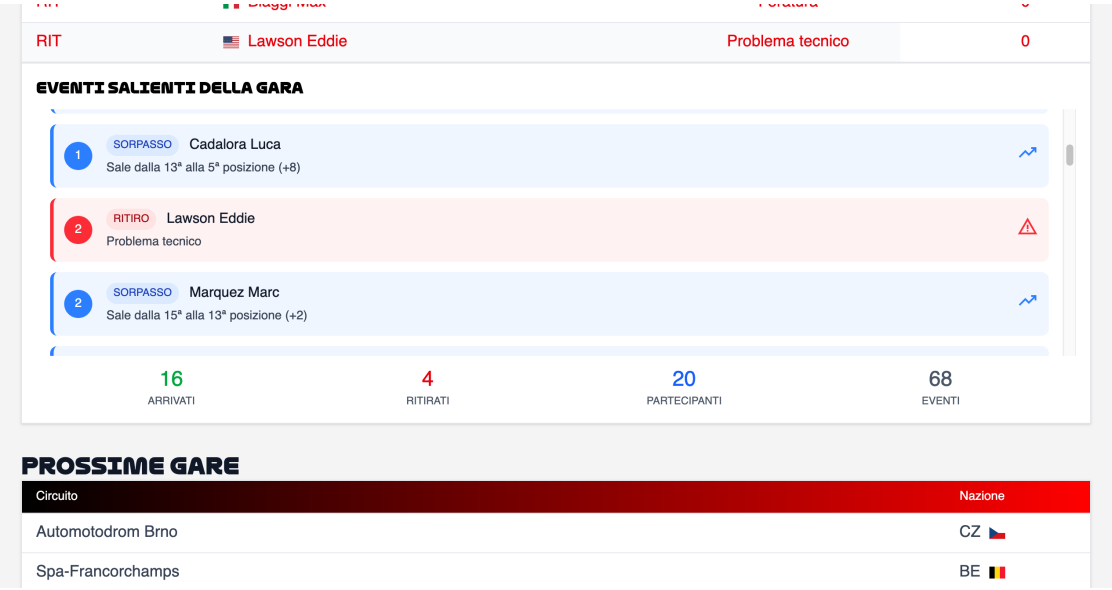


Figure 9: Eventi salienti dell'ultima gara

5.3 Modello predittivo basato su Machine Learning

Per migliorare ulteriormente il realismo della simulazione, è stato implementato un modulo predittivo contenuto nella classe `MLRacePredictor`. Questo modulo impiega due modelli di regressione **Random Forest** per stimare:

- Il **tempo previsto** per ogni singolo giro di un pilota: calcolato in base alle caratteristiche individuali del pilota e allo stato attuale della gara;
- La **probabilità di incorrere in un incidente** o evento negativo durante un determinato giro: similmente al tempo giro, questa stima si basa sulle caratteristiche del pilota e sullo stato della gara, con una maggiore enfasi sul `risk_profile` del pilota.

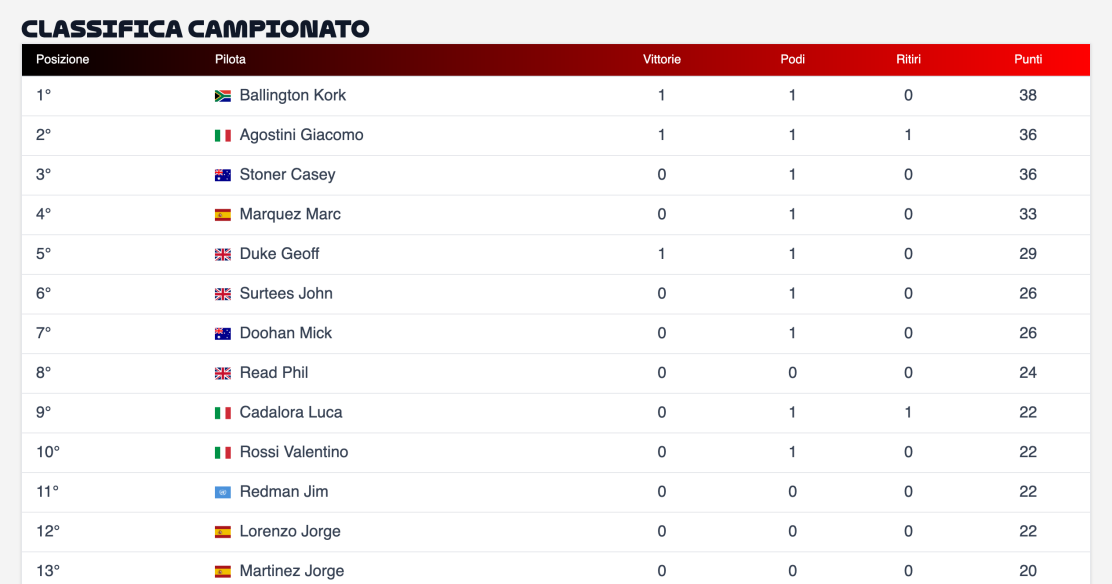
Per l'addestramento di tali modelli è stato generato un dataset sintetico che simula diverse condizioni di gara e profili pilota. Se i modelli non sono già stati addestrati al primo utilizzo, la classe `MLRacePredictor` si auto-allena generando un ampio set di dati simulati. Questo approccio elimina la dipendenza da dati storici reali complessi, pur garantendo che i modelli apprendano correlazioni realistiche tra i fattori in gioco. Le variabili utilizzate come input per l'addestramento includono sia parametri personali del pilota (punteggio normalizzato, coerenza, resistenza sotto pressione, livello di affaticamento e propensione al rischio) sia variabili legate alla gara (numero del giro corrente, progresso della gara, posizione attuale in classifica e fase della corsa).

Entrambi i modelli sono stati addestrati su dati sintetici generati casualmente ma aderenti a distribuzioni e correlazioni plausibili, garantendo così una simulazione credibile senza la necessità di dati storici completi.

5.4 Aggiornamento dinamico dello stato della gara

Durante l'esecuzione della simulazione, il sistema utilizza in tempo reale le predizioni fornite dai modelli per aggiornare lo stato di ciascun pilota. In particolare:

- I **tempi giro** vengono calcolati integrando la previsione ML con fattori casuali che simulano l'imprevedibilità di un singolo giro, oltre a un "bonus giornata" che introduce una variabilità ulteriore per ciascun pilota per l'intera gara.
- La **posizione** in gara viene aggiornata dinamicamente sulla base dei tempi totali accumulati e dei giri completati da ciascun pilota.
- Eventuali **ritiri** o penalità vengono determinati in base alla probabilità stimata di incidente, con la possibilità di definire diverse cause di ritiro (es. caduta, problema tecnico, foratura, problema motore).



CLASSIFICA CAMPIONATO

Posizione	Pilota	Vittorie	Podi	Ritiri	Punti
1°	Ballington Kork	1	1	0	38
2°	Agostini Giacomo	1	1	1	36
3°	Stoner Casey	0	1	0	36
4°	Marquez Marc	0	1	0	33
5°	Duke Geoff	1	1	0	29
6°	Surtees John	0	1	0	26
7°	Doohan Mick	0	1	0	26
8°	Read Phil	0	0	0	24
9°	Cadalora Luca	0	1	1	22
10°	Rossi Valentino	0	1	0	22
11°	Redman Jim	0	0	0	22
12°	Lorenzo Jorge	0	0	0	22
13°	Martinez Jorge	0	0	0	20

Figure 10: Classifica aggiornata ad ogni gara

Questa architettura incrementa significativamente la profondità e il realismo del motore di simulazione, permettendo di ottenere risultati verosimili e coinvolgenti per gli utenti. Ogni evento di gara, come sorpassi, ritiri o lievi errori, viene registrato e incluso nei risultati finali per fornire un resoconto dettagliato della simulazione.