Specifiche Funzionali: Applicazione Gestione Statistiche Squadra di Calcio

1. Obiettivo

L'applicazione deve simulare la gestione delle statistiche di una squadra di calcio, permettendo di visualizzare un elenco di giocatori con le relative informazioni. I dati devono essere recuperati da una simulazione di chiamata API utilizzando TypeScript.

---

2. Requisiti Funzionali

2.1 Interfaccia Utente (UI)

L'applicazione deve essere strutturata in una pagina HTML con:

- Un **titolo** della squadra.
- Una **tabella o lista** per visualizzare i giocatori con le loro statistiche.
- Un **indicatore di stato** della richiesta API (Caricamento, Successo, Errore).
- Un **pulsante per aggiornare i dati**.

2.2 Struttura dei Dati

2.2.1 Interfaccia Giocatore

Ogni giocatore deve essere rappresentato con la seguente interfaccia TypeScript:

```
interface Giocatore {
    id: number;
    nome: string;
    ruolo: Ruolo;
    goal: number;
    assist: number;
    partiteGiocate: number;
}
```

2.2.2 Enum Ruolo

I ruoli possibili dei giocatori devono essere definiti tramite un enum:

```
enum Ruolo {
        Portiere = "Portiere",
        Difensore = "Difensore",
        Centrocampista = "Centrocampista",
        Attaccante = "Attaccante"
}
```

2.3 Simulazione API e Gestione Dati

2.3.1 Recupero Dati

L'applicazione deve simulare una chiamata API utilizzando Promise e setTimeout per restituire un array di giocatori dopo un ritardo di 2 secondi implementando la seguente funzione con il seguente array di Giocatore

```
function fetchGiocatori(): Promise<Giocatore[]>

const datiGiocatori: Giocatore[] = [
    { id: 1, nome: "Gianluigi Donnarumma", ruolo: Ruolo.Portiere, goal: 0, assist: 0, partiteGiocate: 30 },
    { id: 2, nome: "Giovanni Di Lorenzo", ruolo: Ruolo.Difensore, goal: 1, assist: 3, partiteGiocate: 28 },
    { id: 3, nome: "Nicolo Barella", ruolo: Ruolo.Centrocampista, goal: 4, assist: 6, partiteGiocate: 29 },
    { id: 4, nome: "Federico Chiesa", ruolo: Ruolo.Attaccante, goal: 7, assist: 5, partiteGiocate: 25 },
    { id: 5, nome: "Ciro Immobile", ruolo: Ruolo.Attaccante, goal: 12, assist: 4, partiteGiocate: 27 }
]
```

2.3.2 Stato della Richiesta

Il sistema deve gestire tre stati della richiesta API:

- **In attesa**: Durante il recupero dei dati, deve essere mostrato un messaggio "Caricamento dati...".
- **Successo**: Dopo il recupero dei dati, devono essere visualizzate le statistiche dei giocatori.
- **Errore**: Se la chiamata API fallisce, deve essere mostrato un messaggio di errore.

```
enum StatoRichiesta {
  Caricamento = "Caricamento",
  Successo = "Successo",
  Errore = "Errore"
}
```

2.4 Logica dell’Applicazione

2.4.1 Classe Squadra

La squadra deve essere gestita tramite una classe TypeScript, che contiene i giocatori e un metodo per aggiornarli:

```
class Squadra {
  nome: string;
  giocatori: Giocatore[];
  stato: StatoRichiesta;

  aggiornaGiocatori(): void
  private render(): void // Metodo per aggiornare il DOM con i dati della squadra
}
```

2.4.2 Creazione e Gestione dell'Interfaccia

L'istanza della classe Squadra deve essere creata nel codice TypeScript principale e i dati devono essere aggiornati al caricamento della pagina e quando l’utente clicca sul pulsante di aggiornamento.

```
const squadra = new Squadra("Italia");
document.getElementById("aggiornaDati")?.addEventListener("click", () => {
  squadra.aggiornaGiocatori();
});
squadra.aggiornaGiocatori();
```

---

3. Requisiti Non Funzionali

- Il codice TypeScript deve essere compilato in JavaScript ed eseguito nel browser.
- Deve essere utilizzato **SCSS o CSS** per lo stile della pagina.
- Il codice deve essere modulare e ben organizzato, con **commenti esplicativi**.

---

4. Casi d'Uso

Caso d'Uso 1: Avvio Applicazione

1. L'utente apre la pagina web.
2. L'applicazione mostra il messaggio **"Caricamento dati..."**.
3. Dopo 2 secondi, vengono visualizzati i dati dei giocatori.

Caso d'Uso 2: Aggiornamento Dati

1. L'utente clicca sul pulsante **"Aggiorna Dati"**.
2. L'applicazione mostra il messaggio **"Caricamento dati..."**.
3. Dopo 2 secondi, i dati vengono aggiornati e visualizzati.

Caso d'Uso 3: Errore di Recupero Dati

1. Se la chiamata API fallisce, l'applicazione mostra un messaggio di errore **"Errore nel caricamento dei dati"**.