

Elaborato per il corso di Basi Di Dati A.A 2022/2023

Ettore Farinelli
ettore.fainelli@studio.unibo.it
0001019995

2 settembre 2023

Indice

1	Analisi dei requisiti	3
1.1	Intervista	3
1.2	Definizioni	4
1.2.1	Operazioni Utente	4
1.2.2	Operazioni Pubblicitario	5
1.2.3	Operazioni Amministratore	5
2	Progettazione Concettuale	6
2.1	Amministratore	6
2.2	Lottatore	6
2.3	Evento	6
2.4	Scontro	7
2.5	Schema concettuale finale	8
3	Progettazione logica	9
3.1	Stima del volume dei dati	9
3.2	Descrizione delle operazioni principali e stima della loro frequenza	10
3.3	Schemi di navigazione e tabelle degli accessi	11
3.3.1	Registrazione nuovo lottatore	11
3.3.2	Rimuovere un lottatore	11
3.3.3	Registrazione/Rimozione nuovo team	12
3.3.4	Registrare/Rimuovere un nuovo sponsor	12
3.3.5	Registrare un evento	12
3.3.6	Scrivere news	13
3.3.7	Registrare/Rimuovere pubblicitario	13
3.3.8	Visualizzare la classifica	13
3.3.9	Visualizzare lottatori	13
3.3.10	Visualizzare news	14
3.3.11	Visualizzare eventi	14
3.3.12	Modificare dati di un partecipante	14

3.4	Analisi delle ridondanze	15
3.4.1	Attributo "record" all'interno di "LOTTATORE" . . .	15
3.5	Traduzione di entità e associazioni in relazioni	16
3.6	Schema relazionale finale	18
3.7	Traduzione operazioni in query SQL	19
3.7.1	Creazione tabelle	19
3.7.2	Operazioni amministratore	22
4	Progettazione dell'applicazione	26
4.1	Descrizione dell'architettura dell'applicazione realizzata	26

Capitolo 1

Analisi dei requisiti

Si pone l'obiettivo di realizzare un database capace di gestire un organizzazione di arti marziali come può essere, per esempio, L'ULTIMATE FIGHTING CHAMPIONSHIP, (UFC). La base di dati dovrà quindi essere capace di registrare nuovi **lottatori** e in caso rimuoverli (squalifica, infortunio, ritiro). Inoltre sarà possibile registrare **eventi**, dove i combattenti si scontreranno aggiornando (dopo gli **scontri**), gli **score** dei partecipanti e in caso le **classifiche**.

1.1 Intervista

A seguito di una prima intervista si sono ottenute le seguenti richieste:

Per ogni partecipante alla lega bisogna tenere traccia del nome, cognome, codice fiscale, data di nascita, team, peso e **arte marziale** in cui vuole lottare. Alla iscrizione di un nuovo lottatore esso verrà inserito all'ultimo posto nella classifica della propria categoria. Ci sarà la possibilità di registrare i **team** dei combattenti tenendo traccia di: nome, amministratore e origine. Saranno presenti diverse classifiche per ogni tipo di categoria dove i lottatori saranno ordinati in base ai loro **record** (V, P, S), dove le vittorie assegnano 3 punti, i pareggi 1 e le sconfitte 0.

Le categorie in cui verranno suddivisi i membri della lega sono: *Peso Piuma* (fino a 65kg), *Welterweight* (65kg - 77kg), *Peso Medio* (77kg - 84kg) e *Pesi Massimi* (da 84kg in poi). Inoltre non ci sarà una vera e propria divisione in arti marziali in quanto combattenti praticanti discipline diverse potranno scontrarsi tra loro, le arti marziali sono le seguenti: *MMA* (Mixed Martial Arts), *BJJ* (Brazil Jiu Jitsu) e infine *Muay Thai*. Per ogni partecipante dovrà essere inserita la disciplina di competenza possibilmente modificabile in futuro (sarà gestito in maniera analoga il peso). Inoltre,

partecipanti appartenenti a una determinata categoria potranno scontrarsi solo con altri membri della stessa. Gli eventi saranno costituiti da almeno 2 combattimenti ciascuno, bisognerà tener traccia del: nome dello stadio, luogo (nazione), costo noleggio stadio, spesa staff, data, orario inizio, orario fine, biglietti standard venduti, biglietti premium venduti, costo biglietti standard, costo biglietti premium, introiti netti, inoltre sarà necessario poter associare a un evento degli sponsor selezionabili tra quelli disponibili che hanno effettuato un contratto con la lega. Ogni lottatore partecipante riceverà un pagamento extra e verrà calcolata una quantità di guadagni tramite pubblicità, tutto in base al numero di biglietti venduti per l'evento, così da poter calcolare gli introiti dell'evento, il quale verrà aggiunto in una HISTORY dove saranno immagazzinati tutti gli eventi passati.

1.2 Definizioni

- **Lottatore:** partecipante alla lega.
- **Organizzatore:** amministratore che ha l'accesso al database e le autorizzazioni per gestirlo.
- **Evento:** un insieme di scontri avente un luogo e una data.
- **Scontro:** un incontro tra due lottatori della stessa categoria.
- **Classifica:** lista numerata in ordine dal partecipante migliore al peggiore in base ai record personali.
- **Arte marziale:** disciplina frequentante da un partecipante.
- **Team:** associazione a cui possono far parte 1 o più combattenti, ha lo scopo di seguirli durante gli scontri e allenarli.
- **Record:** terna di vittorie, pareggi, sconfitte (V, P, S), ogni partecipante ha la sua che definisce la sua posizione in classifica.

1.2.1 Operazioni Utente

- Visualizzare classifiche.
- Visualizzare news.
- Visualizzare eventi.
- Visualizzare lottatori.

1.2.2 Operazioni Pubblicitario

- Scrivere news.

1.2.3 Operazioni Amministratore

- Registrare un nuovo lottatore.
- Rimuovere un lottatore.
- Registrare un nuovo team.
- Rimuovere un team.
- Aggiungere/Rimuovere uno sponsor.
- Registrare/Rimuovere pubblicitari.
- Registrare un evento.
- Rimuovere news.
- Modificare i dati dei partecipanti.

Capitolo 2

Progettazione Concettuale

2.1 Amministratore

L'amministratore è colui che utilizza effettivamente a livello di applicazione il database, quindi sarà lui che organizzerà tutte le altre parti principali del database come i lottatori, gli eventi, i team e le sponsorizzazioni.

2.2 Lottatore

I lottatori sono il cuore dell'intero database, per questo parte tutto dall'entità "lottatore" a cui sarà associata un'entità "record" creata in modo tale da poter tener traccia dello score di ogni combattente, inoltre i lottatore potranno o meno far parte di un team precedentemente registrato. Infine, ogni lottatore potrà partecipare a uno scontro per ogni evento, in questo caso non sono riuscito a gestire (a livello di schema concettuale) il vincolo per il quale due lottatori di categorie diverse non possono combattere.

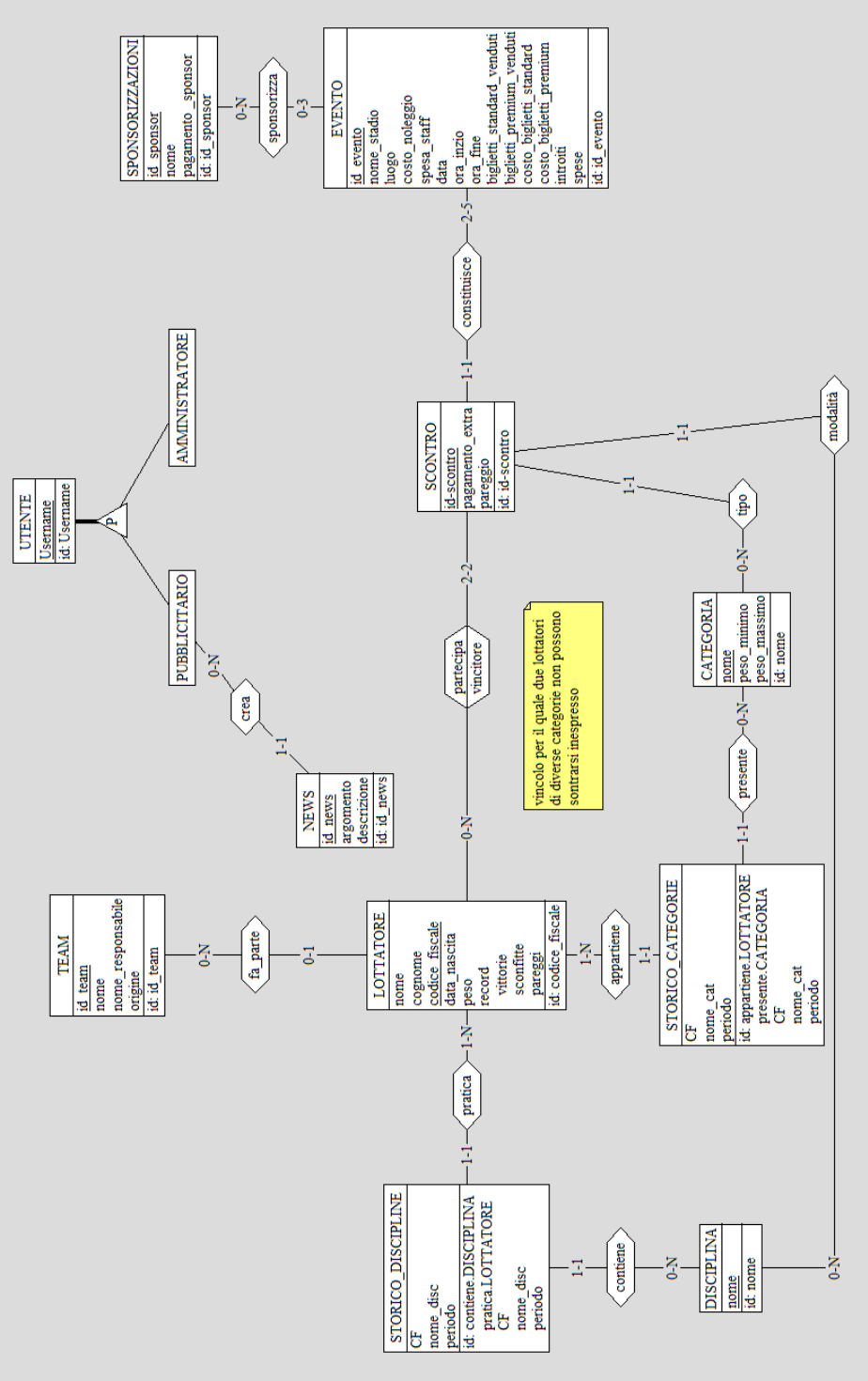
2.3 Evento

Gli eventi sono il principale elemento della lega dal quale proviene il profitto di quest'ultima. Quindi, è importante tenerne traccia con un'entità "Storico_eventi", inoltre ogni evento potrà essere sponsorizzato da uno o più sponsor registrati nella lega.

2.4 Scontro

Gli scontri costituiscono gli eventi e sono formati da due lottatori della stessa categoria, è importante anche che i due partecipanti a uno scontro ricevano un pagamento in base alle volontà dell'amministratore.

2.5 Schema concettuale finale



Capitolo 3

Progettazione logica

3.1 Stima del volume dei dati

- UTENTE: E, 150
- AMMINISTRATORE: E, 1
- PUBBLICITARIO: E, 15
- crea: A, 1000
- NEWS: E, 1000
- LOTTATORE: E, 250
- CATEGORIA: E, 4
- STORICO_CATEGORIE: E, 400
- pratica: A, 400
- contiene: A, 400
- STORICO_DISCIPLINE: E, 400
- appartiene: A, 400
- presente: A, 400
- DISCIPLINA: E, 3
- SCONTRO: E, 2000
- tipo: A, 2000

- modalità: A, 2000
- partecipa: A, 4000
- EVENTO: E, 700
- costituisce: A, 2000
- SPONSORIZZAZIONI: E, 15
- sponsorizza: A, 1400
- TEAM: E, 150
- fa_parte: A, 150

3.2 Descrizione delle operazioni principali e stima della loro frequenza

- Nuovo utente: 1/10 g
- Registrazione nuovo lottatore: 1/10 g
- Rimuovere un lottatore: 1/15 g
- Registrazione nuovo team: 1/12 g
- Rimuovere un team: 1/18 g
- Registrare un nuovo sponsor: 1/120 g
- Rimuovere uno sponsor: 1/210 g
- Registrare un evento: 1/20 g
- Scrivere news: 1/ g
- Rimuovere news: 1/20 g
- Registrare pubblicitario: 1/60 g
- Rimuovere pubblicitario: 1/90 g
- Visualizzare la classifica: 50/ g
- Visualizzare lottatori: 30/ g

- Visualizzare news: 100/ g
- Visualizzare eventi passati: 10/ g
- Modificare dati di un partecipante: 1/5 g

3.3 Schemi di navigazione e tabelle degli accessi

3.3.1 Registrazione nuovo lottatore

Tavola degli accessi

Concetto	Costrutto	Accessi	Tipo
LOTTATORE	E	1	S
TEAM	E	1	L
DISCIPLINA	E	1	L
CATEGORIA	E	1	L
STORICO_CATEGORIE	E	2	S
STORICO_DISCIPLINE	E	2	S
fa_parte	A	1	L
pratica	A	1	S
contiene	A	1	S
appartiene	A	1	S
presente	A	1	S
<i>TOTALE</i>		18	

3.3.2 Rimuovere un lottatore

Tavola degli accessi

Concetto	Costrutto	Accessi	Tipo
LOTTATORE	E	1	S
STORICO_CATEGORIE	E	1	S
STORICO_DISCIPLINE	E	1	S
appartiene	A	1	L
pratica	A	1	L
<i>TOTALE</i>		8	

3.3.3 Registrazione/Rimozione nuovo team

Tavola degli accessi

Concetto	Costrutto	Accessi	Tipo
TEAM	E	1	S
<i>TOTALE</i>		2	

3.3.4 Registrare/Rimuovere un nuovo sponsor

Tavola degli accessi

Concetto	Costrutto	Accessi	Tipo
SPONSORIZZAZIONI	E	1	S
<i>TOTALE</i>		2	

3.3.5 Registrare un evento

Tavola degli accessi

Concetto	Costrutto	Accessi	Tipo
EVENTO	E	1	S
SCONTRO	E	3	S
SPONSORIZZAZIONI	E	2	L
LOTTATORE	E	6	L
STORICO_CATEGORIE	E	6	L
LOTTATORE	E	6	S
CATEGORIA	E	6	L
sponsorizza	A	2	L
partecipa	A	6	S
costituisce	A	3	S
tipo	A	3	L
modalità	A	3	L
appartiene	A	6	L
<i>TOTALE</i>		72	

3.3.6 Scrivere news

Tavola degli accessi

Concetto	Costrutto	Accessi	Tipo
PUBBLICITARIO	E	1	L
NEWS	E	1	S
crea	A	1	L
<i>TOTALE</i>		4	

3.3.7 Registrare/Rimuovere pubblicitario

Tavola degli accessi

Concetto	Costrutto	Accessi	Tipo
PUBBLICITARIO	E	1	S
<i>TOTALE</i>		2	

3.3.8 Visualizzare la classifica

Tavola degli accessi

Concetto	Costrutto	Accessi	Tipo
LOTTATORE	E	250	L
STORICO_CATEGORIE	E	250	L
appartiene	A	250	L
<i>TOTALE</i>		750	

3.3.9 Visualizzare lottatori

Tavola degli accessi

Concetto	Costrutto	Accessi	Tipo
LOTTATORE	E	250	L
STORICO_CATEGORIE	E	400	L
STORICO_DISCIPLINE	E	400	L
TEAM	E	150	L
appartiene	A	400	L
pratica	A	400	L
fa_parte	A	150	L
<i>TOTALE</i>		2150	

3.3.10 Visualizzare news

Tavola degli accessi

Concetto	Costrutto	Accessi	Tipo
PUBBLICITARIO	E	15	L
NEWS	E	1000	L
crea	E	1000	L
<i>TOTALE</i>		2015	

3.3.11 Visualizzare eventi

Tavola degli accessi

Concetto	Costrutto	Accessi	Tipo
EVENTO	E	700	L
SPONSORIZZAZIONI	E	1400	L
SCONTRO	E	2000	L
sponsorizza	A	1400	L
costituisce	A	400	L
<i>TOTALE</i>		5800	

3.3.12 Modificare dati di un partecipante

Tavola degli accessi

Concetto	Costrutto	Accessi	Tipo
LOTTATORE	E	1	S
STORICO_CATEGORIE	E	1	S
STORICO_DISCIPLINE	E	1	S
TEAM	E	1	L
DISCIPLINA	E	1	L
CATEGORIA	E	1	L
fa_parte	A	1	L
pratica	A	1	S
contiene	A	1	S
appartiene	A	1	S
presente	A	1	S
<i>TOTALE</i>		18	

3.4 Analisi delle ridondanze

3.4.1 Attributo "record" all'interno di "LOTTATORE"

L'attributo record presente nell'entità lottatore potrebbe essere ricavato attraverso una ricerca di tutte le vittorie, sconfitte e pareggi effettuati da un lottatore attraverso l'entità scontro durante la creazione delle classifiche.

Caso senza ridondanza

Tavola degli accessi

Concetto	Costrutto	Accessi	Tipo
LOTTATORE	E	250	L
SCONTRO	E	2000	L
STORICO_CATEGORIE	E	250	L
partecipa	A	2000	L
appartiene	A	250	L
<i>TOTALE</i>		4750	

$$\frac{50}{g} \times 4750R = 237500 \frac{R}{g} \quad (3.1)$$

Caso con ridondanza

Tavola degli accessi

Concetto	Costrutto	Accessi	Tipo
LOTTATORE	E	250	L
STORICO_CATEGORIE	E	250	L
appartiene	A	250	L
<i>TOTALE</i>		750	

$$\frac{50}{g} \times 250R = 12500 \frac{R}{g} \quad (3.2)$$

Concludo quindi che devo preservare la ridondanza.

3.5 Traduzione di entità e associazioni in relazioni

Lottatore(codice_fiscale, nome, cognome, data_nascita, peso, id_team*)
fk: id_team references Team

Team(id_team, nome, nome_responsabile, origine)

Utente(Username, pubblicitario*, amministratore*)

Pubblicitario(Username)
fk: Username references Utente

Amministratore(Username)
fk: Username references Utente

News(id_news, argomento, descrizione, Username)
fk: Username references Pubblicitario

Sponsorizzazioni(id_sponsor, nome, pagamento_sponsor)

Evento(id_evento, nome_stadio, luogo, costo_noleggio, spesa_staff, data, ora_inizio, ora_fine, sponsor*, introiti, spese, biglietti_standard_venduti, biglietti_premium_venduti, costo_biglietti_standard, costo_biglietti_premium)
fk: sponsor references Sponsorizzazioni

Scontro(id_scontro, id_evento, disciplina, categoria, pagamento_extra)
fk: id_evento references Evento
fk: categoria references Categoria
fk: disciplina references Disciplina

Partecipa(id_scontro, id_evento, codice_fiscale, vincitore)
fk: (id_evento, id_scontro) references Scontro
fk: codice_fiscale references Lottatore

Categoria(nome, peso_minimo, peso_massimo)

Storico_Categorie(codice_fiscale, nome_cat, periodo)
fk: codice_fiscale references Lottatore
fk: nome_cat references Categoria

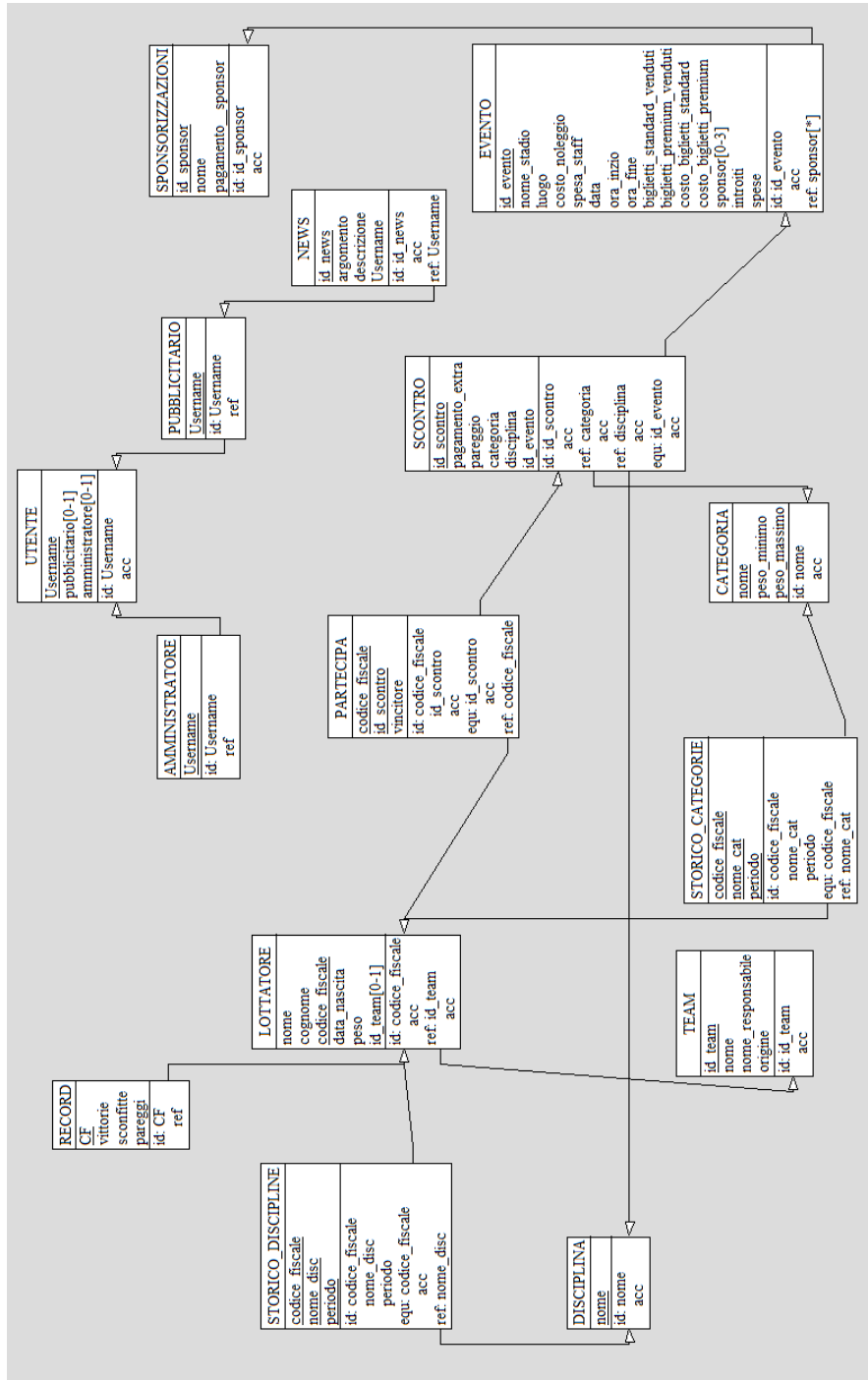
Disciplina(nome)

Storico_Discipline(codice_fiscale, nome_disc, periodo)

fk: codice_fiscale references Lottatore

fk: nome_disc references Disciplina

3.6 Schema relazionale finale



3.7 Traduzione operazioni in query SQL

3.7.1 Creazione tabelle

```
CREATE TABLE UTENTE (  
    username varchar(40) NOT NULL,  
    passw varchar(40) NOT NULL,  
    tipo varchar(30) NOT NULL,  
    PRIMARY KEY (username)  
);  
  
CREATE TABLE LOTTATORE (  
    codiceFiscale varchar(50) NOT NULL,  
    nome varchar(20) NOT NULL,  
    cognome varchar(30) NOT NULL,  
    dataNascita date NOT NULL,  
    peso float NOT NULL,  
    id_team integer,  
    PRIMARY KEY (codiceFiscale)  
);  
  
CREATE TABLE TEAM (  
    idTeam integer NOT NULL,  
    nome varchar(50) NOT NULL,  
    nome_responsabile varchar(20),  
    origine varchar(40),  
    PRIMARY KEY (idTeam)  
);  
  
CREATE TABLE RECORD (  
    codiceFiscale varchar(50) NOT NULL,  
    vittorie integer,  
    sconfitte integer,  
    pareggi integer,  
    PRIMARY KEY (codiceFiscale)  
);  
  
CREATE TABLE STORICO_CATEGORIE (  
    codiceFiscale varchar(50) NOT NULL,  
    nome_cat varchar(16) NOT NULL,  
    periodo varchar(32) NOT NULL,
```

```

        PRIMARY KEY (codiceFiscale, nome_cat, periodo)
    );

CREATE TABLE DISCIPLINE (
    codiceFiscale varchar(50) NOT NULL,
    nome_disc varchar(16) NOT NULL,
    periodo varchar(32) NOT NULL,
    PRIMARY KEY (codiceFiscale, nome_disc, periodo)
);

CREATE TABLE CLASSIFICA_PIUMA (
    codiceFiscale varchar(50) NOT NULL,
    nome varchar(20) NOT NULL,
    cognome varchar(30) NOT NULL,
    peso float NOT NULL,
    arteMarziale ENUM('BJJ', 'MMA', 'MuayThai') NOT NULL,
    PRIMARY KEY (codiceFiscale)
);

CREATE TABLE CLASSIFICA_WELTERWEIGHT (
    codiceFiscale varchar(50) NOT NULL,
    nome varchar(20) NOT NULL,
    cognome varchar(30) NOT NULL,
    peso float NOT NULL,
    arteMarziale ENUM('BJJ', 'MMA', 'MuayThai') NOT NULL,
    PRIMARY KEY (codiceFiscale)
);

CREATE TABLE CLASSIFICA_MEDIO (
    codiceFiscale varchar(50) NOT NULL,
    nome varchar(20) NOT NULL,
    cognome varchar(30) NOT NULL,
    peso float NOT NULL,
    arteMarziale ENUM('BJJ', 'MMA', 'MuayThai') NOT NULL,
    PRIMARY KEY (codiceFiscale)
);

CREATE TABLE CLASSIFICA_MASSIMI (
    codiceFiscale varchar(50) NOT NULL,
    nome varchar(20) NOT NULL,
    cognome varchar(30) NOT NULL,

```

```

    peso float NOT NULL,
    arteMarziale ENUM('BJJ', 'MMA', 'MuayThai') NOT NULL,
    PRIMARY KEY (codiceFiscale)
);

CREATE TABLE DISCIPLINA (
    nome ENUM('BJJ', 'MMA', 'MuayThai') PRIMARY KEY
);

CREATE TABLE CATEGORIA (
    nome ENUM('PesoPiuma', 'Welterweight', 'PesoMedio', 'PesiMassimi') PRIMARY KEY,
    pesoMinimo integer,
    pesiMassimi integer
);

CREATE TABLE PARTECIPA (
    codiceFiscale varchar(50),
    idScontro integer,
    vincitore varchar(50),
    PRIMARY KEY (codiceFiscale, idScontro)
);

CREATE TABLE SCONTRO (
    idEvento integer NOT NULL,
    idScontro integer NOT NULL,
    disciplina ENUM('BJJ', 'MMA', 'MuayThai'),
    categoria ENUM('PesoPiuma', 'Welterweight', 'PesoMedio', 'PesiMassimi'),
    pagamentoExtra float,
    PRIMARY KEY (idEvento, idScontro)
);

CREATE TABLE EVENTO (
    idEvento integer NOT NULL,
    nomeStadio varchar(40) NOT NULL,
    luogo varchar (50) NOT NULL,
    costoNoleggio float,
    spesaStaff float,
    dataEvento date NOT NULL,
    oraInizio varchar(10) NOT NULL,
    oraFine varchar(10) NOT NULL,
    bigliettiStandardVenduti integer,

```

```

        bigliettiPremiumVenduti integer,
        costoBigliettiPremium integer,
        costoBigliettiStandard integer,
        sponsor JSON,
        introiti float,
        spese float,
        PRIMARY KEY (idEvento)
        CONSTRAINT ORARIO CHECK (oraInizio <= oraFine) => (Rimosso perchè risultava
        finivano dopo mezzanotte)
    );

CREATE TABLE SPONSORIZZAZIONI (
    idSponsor integer NOT NULL,
    nome varchar(40) NOT NULL,
    pagamentoSponsor float,
    PRIMARY KEY (idSponsor)
);

CREATE TABLE NEWS (
    idNews integer NOT NULL,
    argomento varchar(128) NOT NULL,
    descrizione varchar(4096) NOT NULL,
    scrittore varchar(40) NOT NULL,
    PRIMARY KEY (idNews)
);

```

3.7.2 Operazioni amministratore

Aggiungere lottatore

```

INSERT INTO LOTTATORE (nome, cognome, codiceFiscale,
    dataNascita, team, peso, categoria, arteMarziale)
VALUES (?, ?, ?, ?, ?, ?, ?, ?);

-- Aggiungere Record --

INSERT INTO RECORD (idRecord, vittorie, sconfitte, pareggi)
VALUES (?, 0, 0, 0);

-- Aggiungere classifica --

```

```
INSERT INTO CLASSIFICA (nome, cognome, codiceFiscale, peso,  
    arteMarziale)  
VALUES (?, ?, ?, ?, ?);
```

P.S: In questo caso inserisco il lottatore nella tabella 'classifica' che non esiste per comodità, nell'applicazione invece sarà presente un controllo che farà in modo che il partecipante venga inserito nella classifica giusta in base al suo peso.

Rimuovere lottatore

```
DELETE FROM LOTTATORE  
WHERE codiceFiscale = ?
```

-- Elimina il record corrispondente --

```
DELETE FROM RECORD  
WHERE codiceFiscale = ?
```

-- Elimina l'elemento in classifica

```
DELETE FROM CLASSIFICA  
WHERE codiceFiscale = ?
```

P.S: vale la stessa cosa che ho detto prima per l'aggiunta classifica.

Aggiungere team

```
INSERT INTO TEAM (idTeam, nome, nome_responsabile, origine)  
VALUES(?, ?, ?, ?);
```

Rimuovere team

```
DELETE FROM TEAM  
WHERE idTeam = ?
```

Aggiungere sponsor

```
INSERT INTO SPONSORIZZAZIONI(idSponsor, nome, pagamentoSponsor)  
VALUES (?, ?, ?);
```


Rimuovere sponsor

```
DELETE FROM SPONSORIZZAZIONI
WHERE idSponsor = ?
```

Registrare evento

```
INSERT INTO EVENTO (idEvento, nomeStadio, luogo, costoNoleggio,
    spesaStaff, dataEvento, oraInizio, oraFine,
    bigliettiStandardVenduti, bigliettiPremiumVenduti,
    costoBigliettiStandard, costoBigliettiPremium, sponsor)
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);
```

-- Inserire evento in storicoEventi --

```
INSERT INTO STORICO_EVENTI (idEvento, introiti, spese,
    guadagniComplessivi)
VALUES (?, ?, ?, ?);
```

-- Aggiungere Scontro (2-5 volte, in base al tipo di evento) --

```
INSERT INTO SCONTRO (idEvento, idScontro, disciplina,
    categoria, pagamentoExtra)
VALUES (?, ?, ?, ?, ?);
```

-- Inserire Scontro nello storicoScontri --

```
INSERT INTO STORICO_SCONTRI (idEvento, idScontro,
    primoPartecipante, secondoPartecipante, vincitore,
    perdente, pareggio)
VALUES (?, ?, ?, ?, ?, ?, ?);
```

Modificare lottatore

```
UPDATE LOTTATORE
SET nome = ?, cognome = ?,
    dataNascita = ?, team = ?,
    peso = ?, categoria = ?,
    arteMarziale = ?
WHERE codiceFiscale = ?
```

```
UPDATE RECORD
SET vittore = ?,
    sconfitte = ?,
    pareggi = ?
WHERE codiceFiscale = ?
```

Visualizzare classifiche

```
SELECT c.codiceFiscale
FROM CLASSIFICA c
JOIN RECORD r ON c.codiceFiscale = r.codiceFiscale
ORDER BY (r.vittoria * 3 + r.pareggio) DESC;
```

Capitolo 4

Progettazione dell'applicazione

4.1 Descrizione dell'architettura dell'applicazione realizzata

L'applicazione che permette di gestire il database è stata creata utilizzando Typescript in combinazione con il framework *React*, mentre per quanto riguarda il database, esso risiede in locale ed è stato sviluppato usando *MySQL*. L'applicazione è un'applicazione web che si appoggia a un server locale. Essa consente quindi la possibilità di connettere frontend e backend utilizzando il protocollo **HTTP**, e quindi attraverso delle **API REST**. Grazie a un server locale, il backend avrà la possibilità di ricevere richieste dal frontend, portando con sé parametri e restituendo valori, il tutto evitando una connessione diretta tramite codice. Questo design permette all'applicazione di avere una scalabilità maggiore, eliminando eventuali dipendenze tra frontend e backend. Inoltre, per la gestione del database da parte del backend, si utilizza la libreria *TypeORM* di Typescript. Libreria che permette di eseguire operazioni sui dati mediante codice.

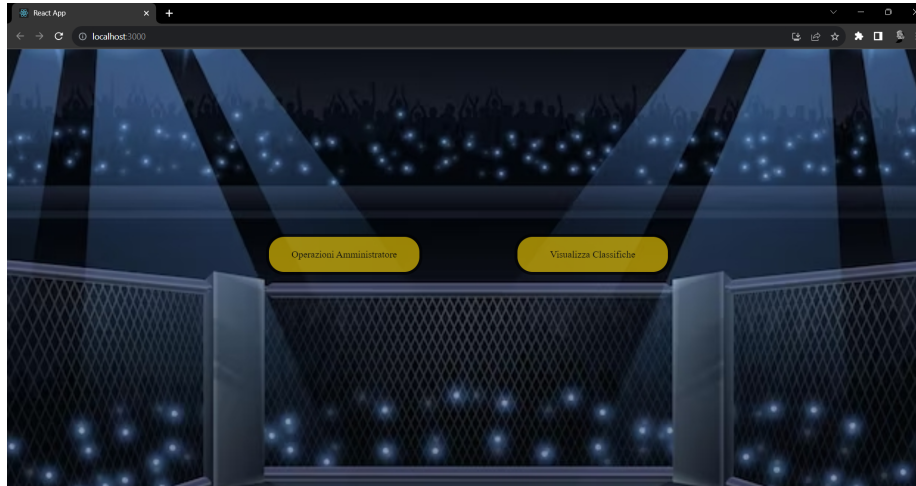


Figura 4.1: *Menù iniziale dell'applicazione*

In questo menù sarà possibile accedere alle operazioni di gestione del database(es: 4.3 per la registrazione di un evento) oppure alle classifiche generali dei lottatori 4.2.

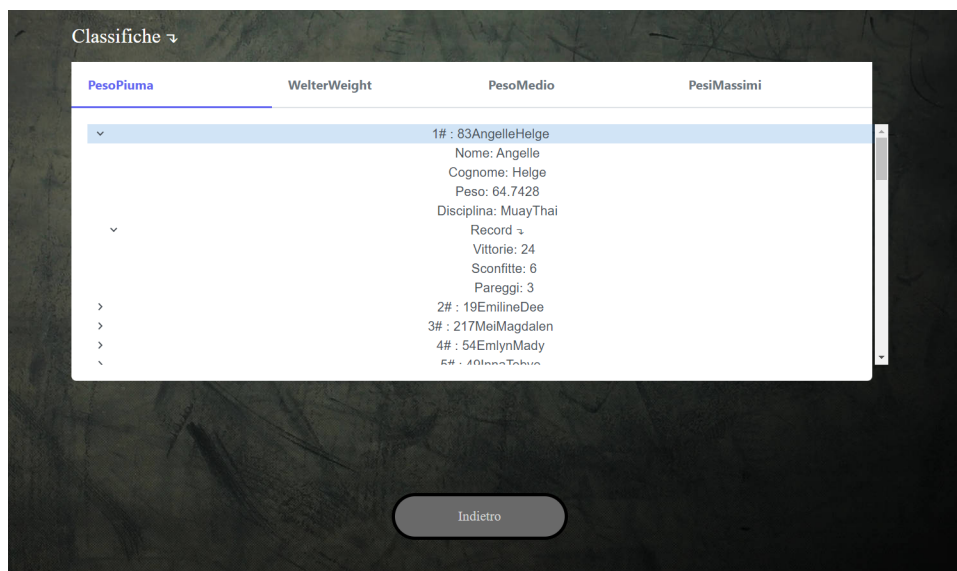


Figura 4.2: *Classifiche di tutti i partecipanti divisi in categorie*

Figura 4.3: *Pagina che permette di registrare un evento e i proprio scontri*

Come si puo notare dall'immagine qua abbiamo una schermata dove è possibile registrare Eventi inserendo tutti i dati necessari nella parte sinistra dello schermo, nella parte destra invece potremmo aggiungere da 2 a 5 scontri (in questo caso notiamo che uno è già stato aggiunto e ci viene proposta la possibilità di rimuoverlo e modificarlo se desiderato).

Figura 4.4: *Pagina che ci permette di aggiungere e rimuovere lottatori*