

DESCRIZIONE DELLA REALTÀ DA ANALIZZARE

Si vogliono gestire le informazioni relative agli eventi di LARP Italia (Live Action Role-Playing Italia) mediante le seguenti informazioni e caratteristiche:

La **campagna** ed il tipo di campagna a cui un giocatore decide di partecipare influenzano i personaggi e la loro creazione. Le campagne si dividono in:

- Estese:
- OneShot:

La campagna decide il mondo di gioco e fonda le basi su cui si dovranno muovere i personaggi, imponendo vincoli di coerenza con la scelta dei personaggi e degli oggetti presenti nel gioco. Definita da nome univoco di campagna, descrizione e tipologia (cyberpunk, fantasy, medievale..).

Il **giocatore** è un cliente dell'azienda identificato dal codice fiscale, inoltre sono necessari i dati anagrafici e un e-mail per scopi di marketing.

Un giocatore può partecipare a una o più campagne e per ognuna può creare più personaggi ma ad ogni evento potrà giocare con un solo personaggio, allo stesso tempo il personaggio potrà essere interpretato da un solo giocatore per evento. Inoltre non è necessario che il giocatore giochi con un personaggio creato da lui.

Tutte le campagne (estese e oneshot) sono composte da eventi, di durata più o meno estesa, che plasmano il mondo di gioco e gli avvenimenti futuri all'interno della campagna.

Gli eventi a loro volta sono formati da missioni inserite nel gioco da NPC (personaggi non giocanti) interpretati da dipendenti di LARP Italia, i quali hanno il compito di indirizzare i giocatori verso lo sviluppo o la conclusione di un evento. Le missioni così come gli eventi possono essere interrotti o mai sviluppati. Le trame una volta portate a termine dai personaggi possono far guadagnare esperienza, oggetti e abilità.

Le campagne OneShot sono caratterizzate da una durata molto inferiore rispetto a quelle estese, queste infatti si riducono ad un singolo evento in cui i giocatori possono scegliere di interpretare un personaggio creato dai writer della campagna.

Gli **eventi** sono tenuti in data singola (anche su più giorni consecutivi di gioco) in una location che può essere affittata o richiesta al comune a seconda del tipo di evento e delle esigenze.

Personaggio:

I **personaggi** si dividono in:

- Personaggi giocanti
- Personaggi non giocanti (NPC)

I personaggi appartengono ad una classe (le classi vengono decise dagli organizzatori coerentemente al mondo di gioco), la quale influisce su alcune abilità del personaggio, le abilità si ottengono durante tutta la campagna e accompagnano il personaggio per tutta la durata della stessa. Non tutte le abilità possono essere ottenute fin da subito, alcune sono vincolate ad altre abilità, si sviluppa così un "albero delle abilità".

Ogni personaggio giocante ha un inventario in cui inserire tutti gli oggetti e gli equipaggiamenti che otterrà durante la campagna. Ogni oggetto ha: descrizione, statistiche di danno, statistiche di difesa, valore in moneta di gioco. Inoltre vi sono oggetti consumabili che avranno un effetto sui personaggi.

Un giocatore può partecipare ad un evento con un personaggio soltanto, ha però la libertà di cambiare il personaggio durante la campagna in caso il personaggio muoia o il giocatore non voglia proseguire con il personaggio fino a quel momento utilizzato. Se il personaggio muore non sarà in nessun modo riutilizzabile all'interno della campagna.

Mondo di gioco:

La campagna si sviluppa in un mondo di gioco unico per campagna che viene influenzato dalle azioni dei personaggi nel corso degli eventi.

Tutti gli **avvenimenti** principali nel mondo di gioco vengono salvati per fini storici, ad esempio: ritrovamenti di armi leggendarie, apertura di antichi edifici, uccisione di NPC importanti, ecc...

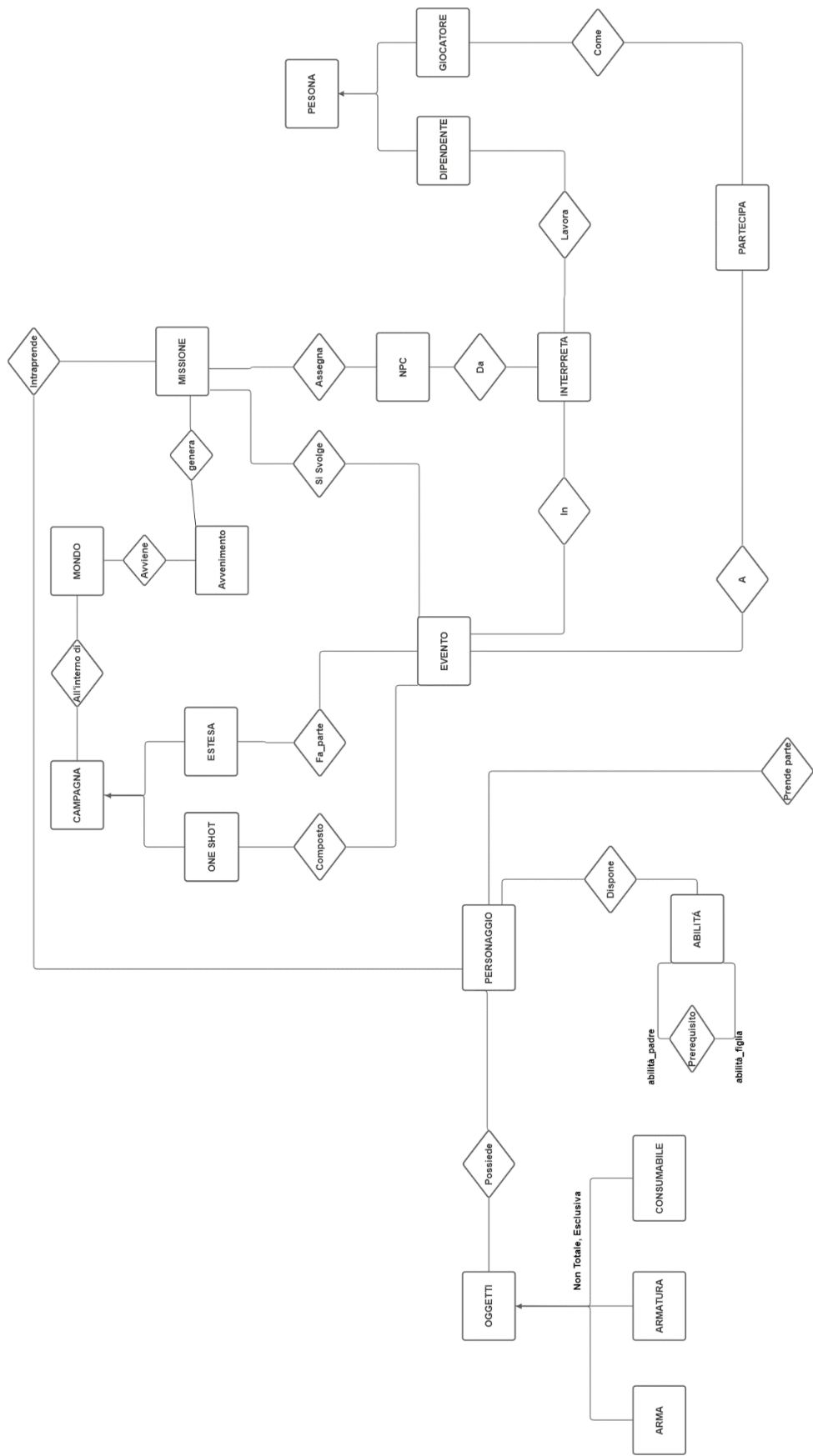
Essi perciò dovranno riportare un titolo univoco sull'evento in cui è avvenuto, una descrizione dettagliata e una lista di partecipanti, nonché l'anno (in gioco).

GLOSSARIO:

TERMINE	DESCRIZIONE	SINONIMI	ASSOCIAZIONI
Campagna	<u>Nome</u> Descrizione		One Shot Estesa Mondo
One Shot	Limite partecipanti		Campagna Evento
Estesa	Anno di partenza		Campagna Evento
Evento	Titolo Data Inizio Data Fine Indirizzo Location Costo Iscrizione Iscrizioni disponibili	Incontro Partita	Campagna Missione Interpreta
Personaggio	<u>Cod_pg</u> Nome Descrizione Tipo Exp	Ruolo Attore	Giocatore Evento Oggetti Abilità Missione Partecipa
Persona	<u>CE</u> Numero Cellulare Indirizzo		Giocatore Dipendente
Giocatore	e-mail		Partecipa Persona
Dipendente	Stipendio		Persona Interpreta
NPC	<u>Cod_npc</u> Nome Descrizione		Interpreta Missione
Missione	<u>Cod_Missione</u> Fallita Guadagno Exp	Trama	Personaggio Avvenimento NPC
Avvenimento	<u>Cod_avv</u> Descrizione Data		Mondo Missione
Mondo	<u>Nome</u> Tipo		Avvenimento Campagna
Oggetti	Quantità		Personaggio

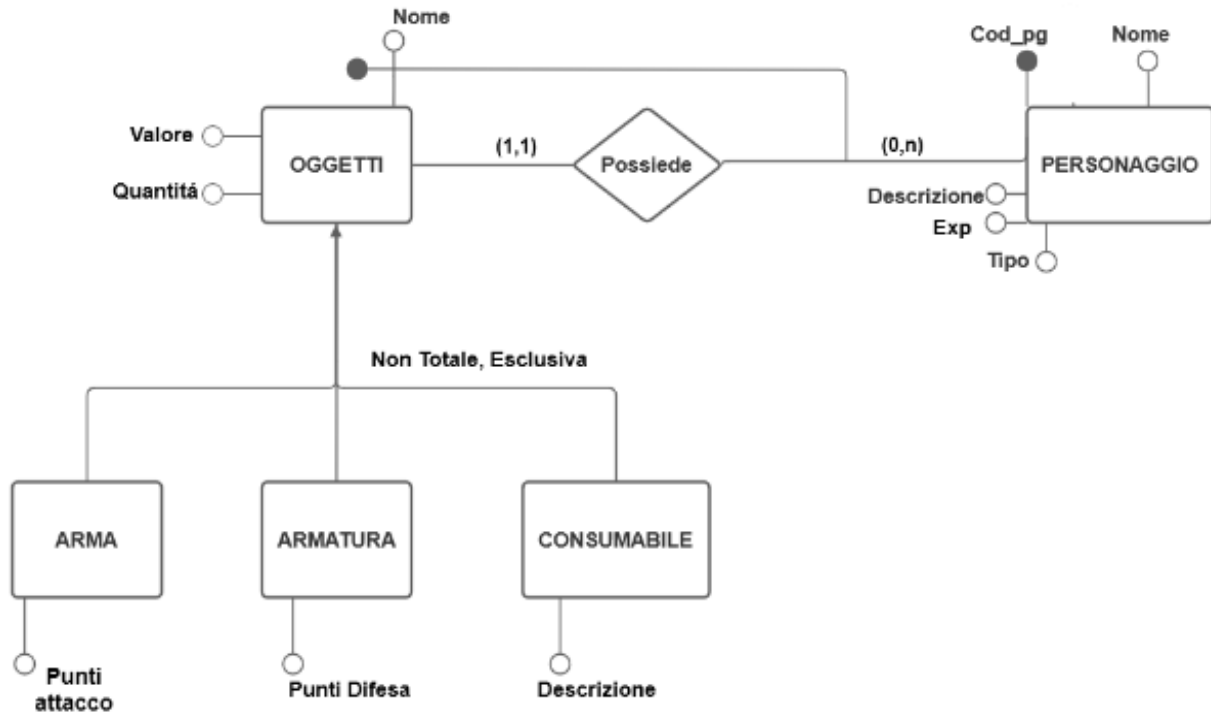
	Valore Nome		Arma Armatura Consumabile
Arma	Punti Attacco		Oggetti
Armatura	Punti Difesa		Oggetti
Consumabile	Descrizione		Oggetti
Abilità	<u>Nome</u> Costo_Exp Restrizione Descrizione		Personaggio

SCHEMA SCHELETRO:



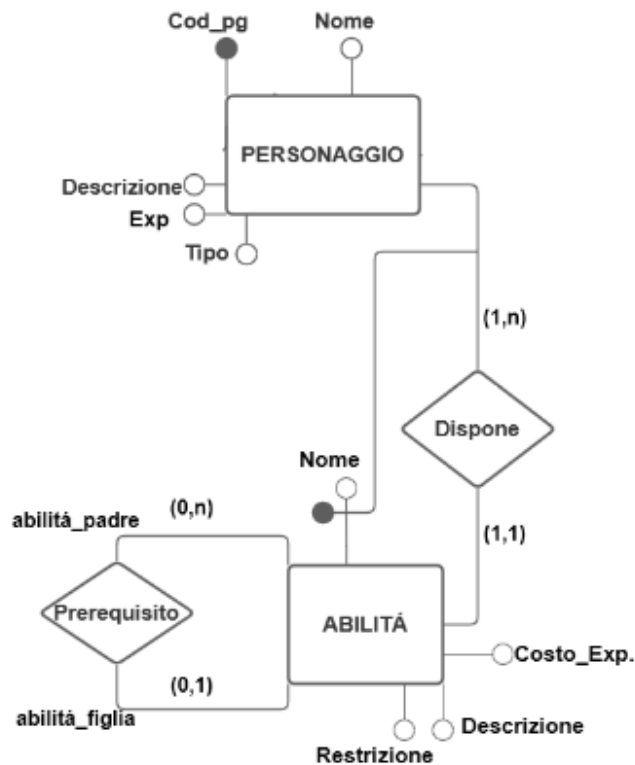
PROGETTAZIONE CONCETTUALE:

Il PERSONAGGIO è identificato da un codice univoco e caratterizzato da un nome, una descrizione, un punteggio che tiene traccia dei punti esperienza accumulati dal personaggio durante la campagna, ed infine il tipo che identifica la classe di appartenenza del personaggio.

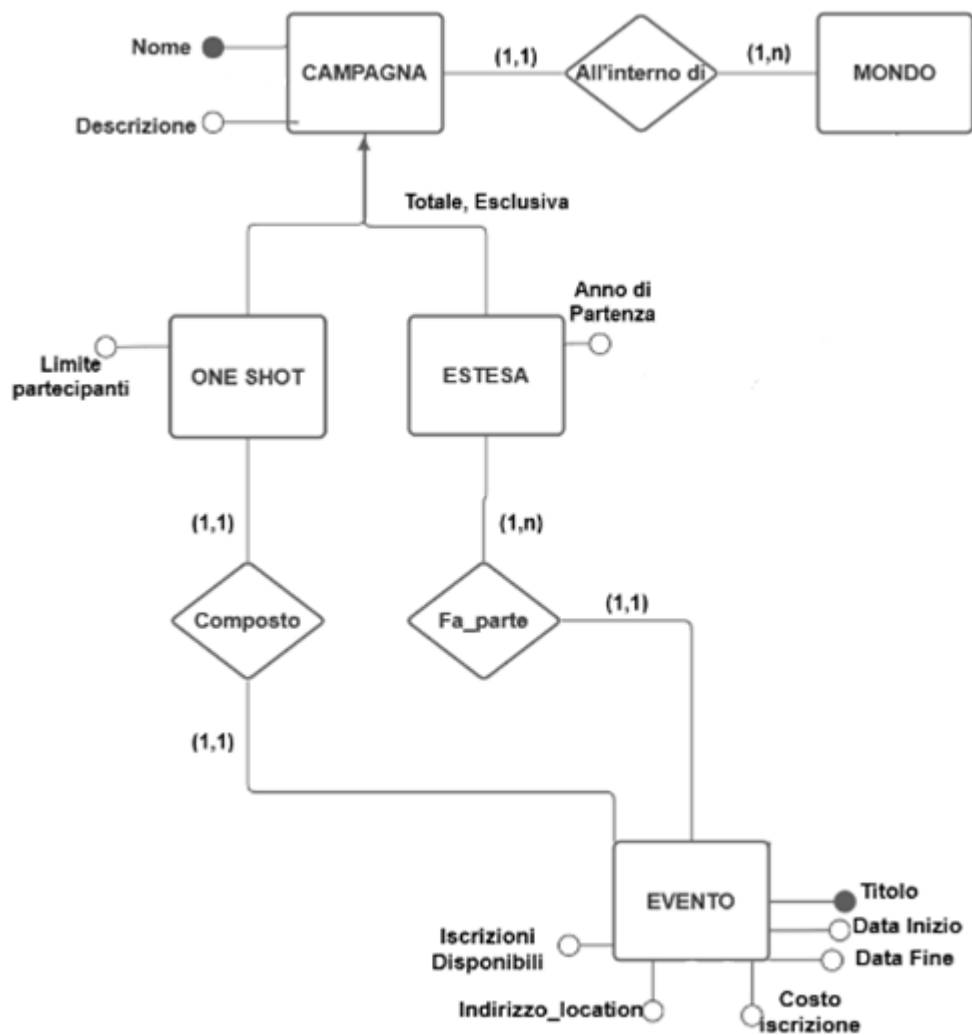


Abbiamo scelto di implementare un identificatore esterno su OGGETTI da PERSONAGGIO poichè l'entità OGGETTI è debole rispetto PERSONAGGIO, infatti se un personaggio non esiste o per qualche motivo non è più in gioco i suoi oggetti andranno persi.

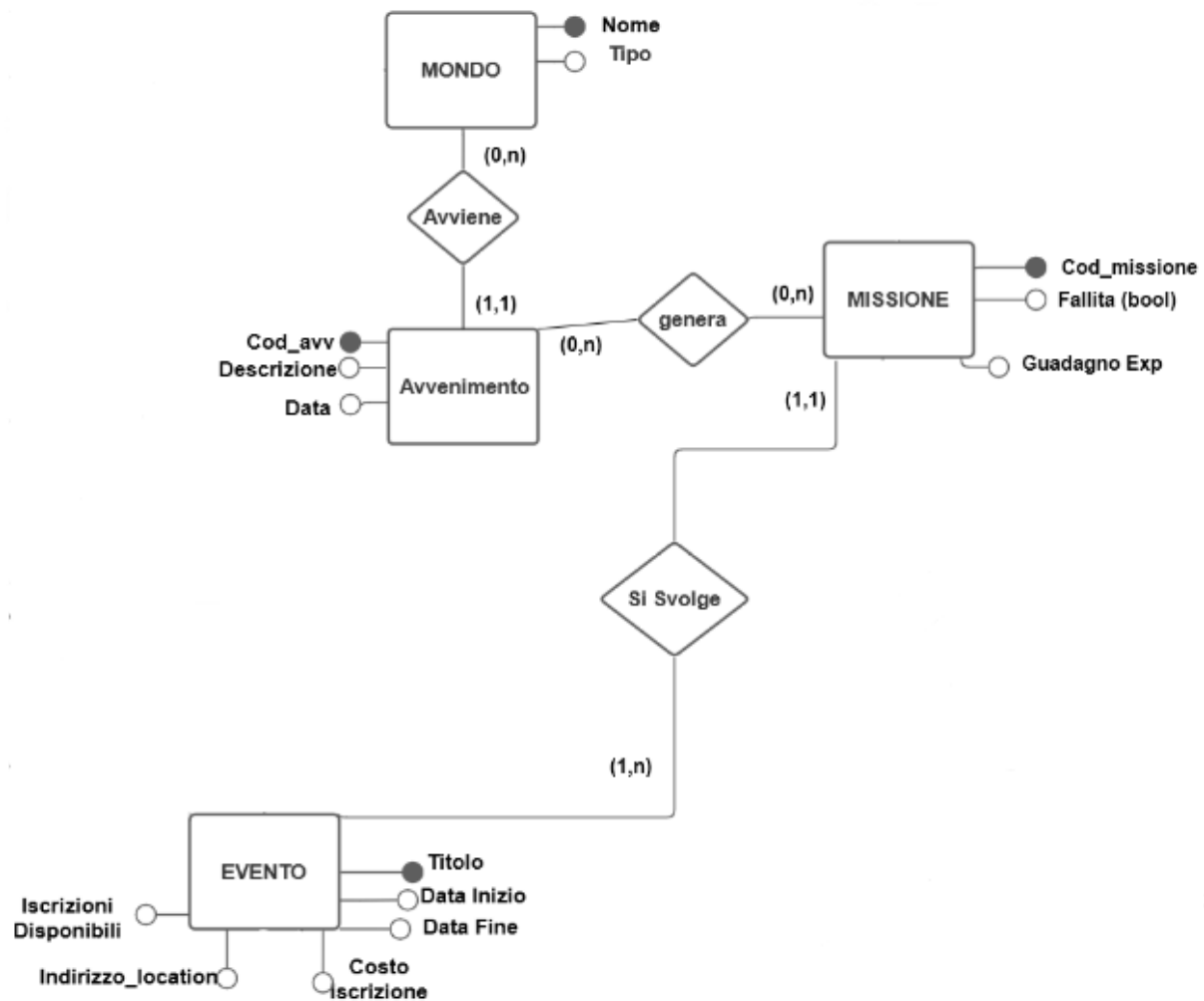
Poichè gli oggetti possono essere di tre tipologie diverse abbiamo deciso di utilizzare una gerarchia Non Totale ed Esclusiva con i diversi tipi di oggetti possibili: ARMA, ARMATURA e CONSUMABILE. La gerarchia è Non Totale perchè potrebbero esistere alcuni oggetti non ascrivibili a nessuna categoria specifica come, ad esempio, oggetti chiave per una missione specifica, la gerarchia è inoltre Esclusiva perchè non è ammesso che un oggetto appartenga contemporaneamente a più categorie. Ogni categoria di oggetti è caratterizzata dall' attributo che è più significativo per la stessa.



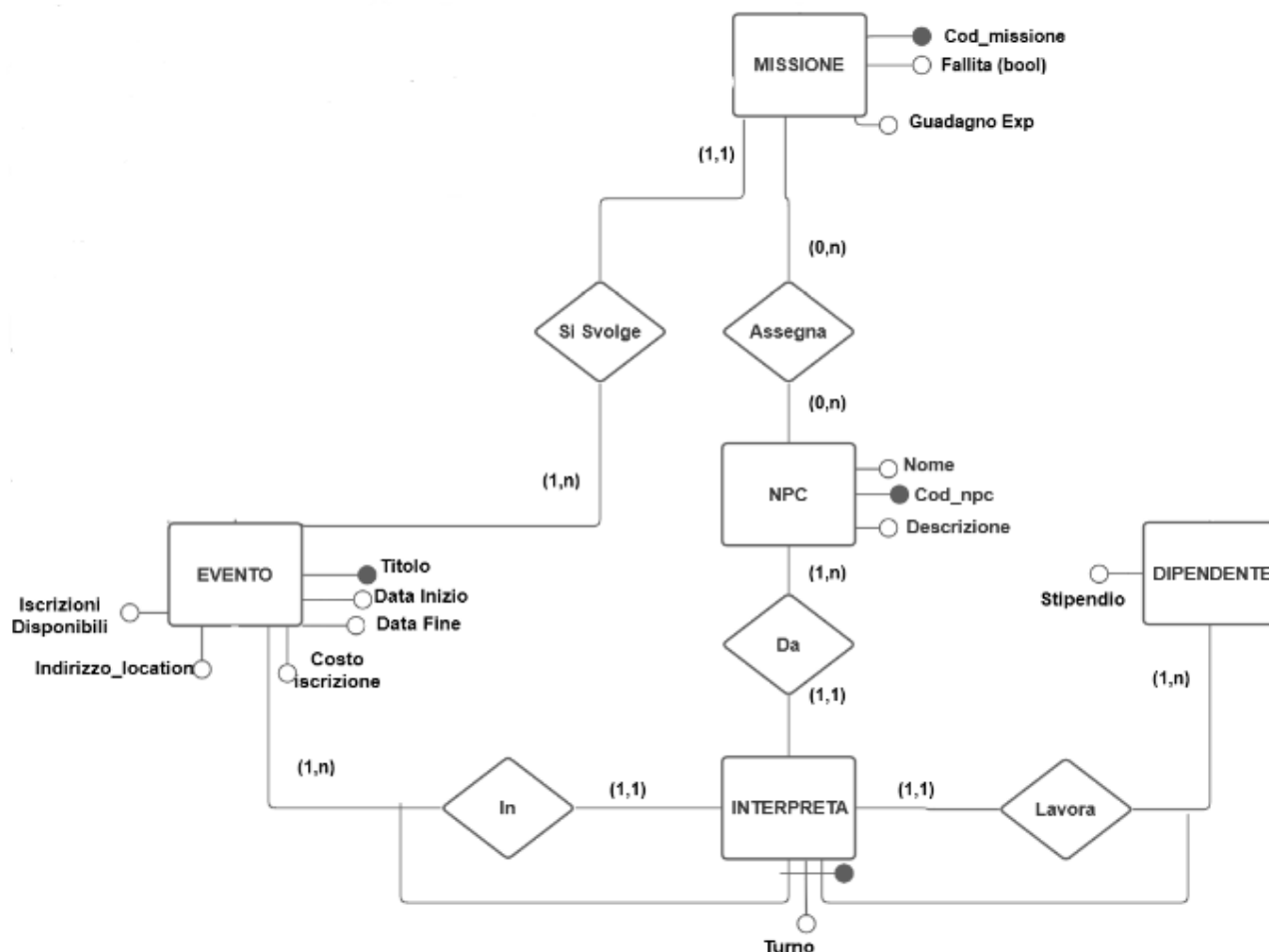
Per quanto riguarda l'attributo ABILITÀ abbiamo trovato appropriato ragionare come nel caso precedente poichè, come per gli oggetti, le abilità sono strettamente dipendenti al personaggio che le possiede, ed in caso di eliminazione del personaggio dalla partita anche quest'ultime saranno cancellate. Abbiamo quindi utilizzato un identificatore esterno da PERSONAGGIO ad ABILITÀ. Inoltre abbiamo deciso di inserire un'autoassociazione su ABILITÀ in modo da creare una gerarchia delle abilità ottenibili dai personaggi giocanti, creando così un "albero delle abilità" ottenibili dal personaggio giocante.



Gli EVENTI a cui prendono parte i personaggi giocanti sono identificati da un titolo e presentano come attributi le informazioni essenziali per i giocatori che vi vogliono partecipare. Per distinguere i due tipi di CAMPAGNE presenti abbiamo deciso di utilizzare una gerarchia totale ed esclusiva. L'entità CAMPAGNA è identificata da un Nome univoco. In questo modo abbiamo eventi differenti che si sviluppano in una campagna di gioco che a sua volta è contenuta in un MONDO di gioco, in questo modo si ha la possibilità di tornare ad utilizzare un mondo di gioco già sviluppato e in cui si sono svolte già diverse campagne.



Le MISSIONI che i personaggi possono intraprendere sono identificate da un Codice e presentano come attributi: il guadagno di esperienza da attribuire ai personaggi in caso di successo e un attributo booleano per indicarne il fallimento. Le MISSIONI fanno parte degli EVENTI e possono essere portate a termine o meno dai personaggi giocatori. La divisione in MISSIONI permette di generare uno storico di tutti gli avvenimenti di un mondo di gioco, rappresentato dall'entità AVVENIMENTO, identificata da un Codice univoco e completo di Data e Descrizione. In questo modo ogni mondo di gioco avrà il proprio storico, così facendo i giocatori potranno immergersi nel gioco più facilmente e gli stessi organizzatori avranno sempre una visione d'insieme delle storie da loro create e sarà così più facile continuare a svilupparle.

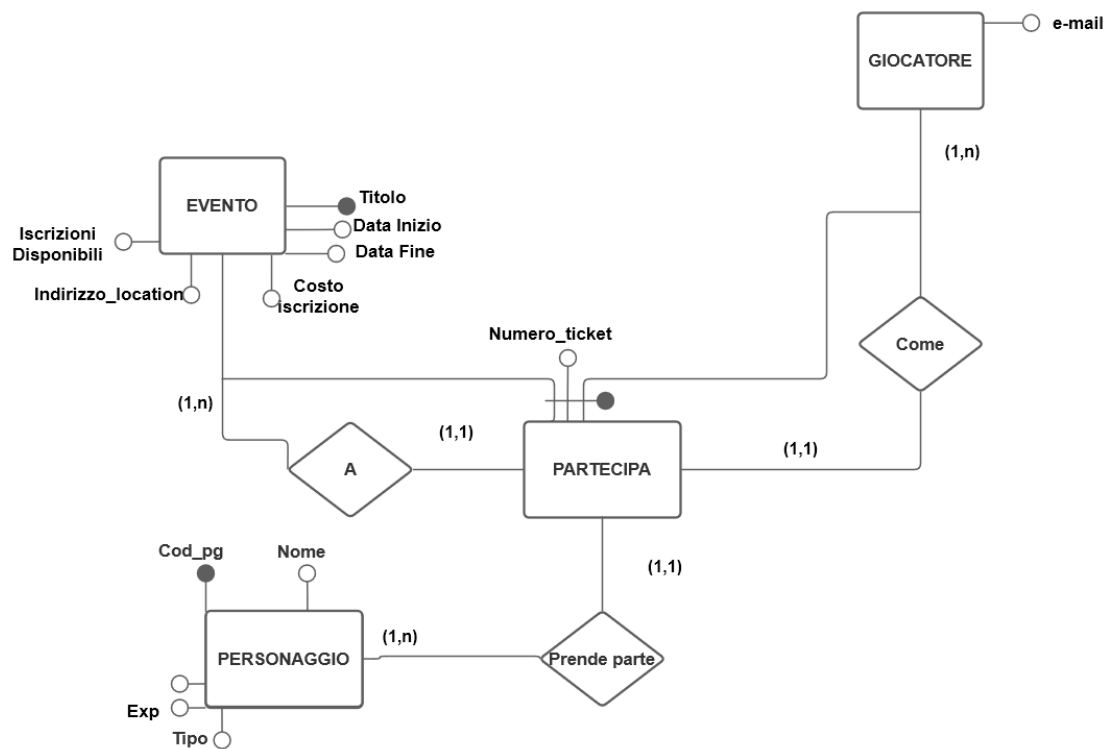


Le missioni sono assegnate ai giocatori da un NPC (Non-Player Character) un personaggio non giocatore interpretato da un DIPENDENTE di LARP Italia, in questo modo gli organizzatori possono indirizzare gli sviluppi della trama contribuendo all'immedesimazione dei giocatori. L'entità NPC è identificata da un Codice univoco per ogni personaggio non giocatore ed è caratterizzata da un Nome e da una Descrizione.

Per poter esprimere il vincolo per il quale un DIPENDENTE può interpretare un solo NPC per ogni EVENTO abbiamo deciso di reificare INTERPRETA ed utilizzare l'identificazione esterna composta, in questo modo un DIPENDENTE durante il suo turno di lavoro può interpretare un solo NPC facente parte dell'EVENTO in corso.



Abbiamo deciso di utilizzare una gerarchia totale ed esclusiva per rappresentare i DIPENDENTI e i GIOCATORI in quanto possono essere raggruppati entrambi sotto l'entità PERSONA, identificata dal Codice Fiscale e caratterizzata da un numero di cellulare ed un indirizzo, inoltre per i dipendenti è presente l'attributo stipendio, mentre per i giocatori l'attributo e-mail, utilizzabile per fini promozionali da LARP Italia. La gerarchia è totale poiché non vi sono altre figure all'interno degli eventi di gioco ed esclusiva poiché appartenere ad una categoria esclude l'appartenenza all'altra.



Per poter esprimere il vincolo per il quale un giocatore può interpretare un solo personaggio in ogni evento, un personaggio può partecipare a più eventi ed inoltre lo stesso personaggio può essere interpretato da più giocatori a patto che cambi l'evento, abbiamo deciso di utilizzare l'identificazione esterna composta reificando PARTECIPA. Così facendo il vincolo è rispettato.

The diagram is an Entity-Relationship (ER) model for a game system. It features the following components:

- Entities and Attributes:**
 - PESONA:** Attributes include *Numero*, *Cellulare*, and *Indirizzo*. It has a primary key *CF*.
 - NPC:** Attributes include *Nome*, *Cod_npc*, and *Descrizione*. It has a primary key *Cod_npc*.
 - GIOCATTORE:** Attribute is *e-mail*.
 - DIPENDENTE:** Attribute is *Stipendio*.
 - EVENTO:** Attributes include *Titolo*, *Data_Inizio*, *Data_Fine*, *Costo*, and *Iscrizione*. It has a primary key *Titolo*.
 - PERSONAGGIO:** Attributes include *Cod_pg*, *Nome*, *Descrizione*, *Exp*, and *Tipo*. It has a primary key *Cod_pg*.
 - ABILITÀ:** Attributes include *Nome*, *Costo_Exp.*, *Descrizione*, and *Restrizione*. It has a primary key *Nome*.
 - ARMA:** Attribute is *Punti attacco*.
 - ARMATURA:** Attribute is *Punti Difesa*.
 - CONSUMABILE:** Attribute is *Descrizione*.
- Relationships and Cardinalities:**
 - Assegna:** Connects *PESONA* and *NPC* with cardinalities (0,n) and (0,n).
 - Si Svolge:** Connects *EVENTO* and *NPC* with cardinalities (1,n) and (1,n).
 - Fa parte:** Connects *EVENTO* and *PERSONAGGIO* with cardinalities (1,1) and (1,n).
 - Composto:** Connects *EVENTO* and *PERSONAGGIO* with cardinalities (1,1) and (1,1).
 - Da:** Connects *EVENTO* and *INTERPRETA* with cardinalities (1,1) and (1,1).
 - In:** Connects *PERSONAGGIO* and *INTERPRETA* with cardinalities (1,1) and (1,n).
 - Lavora:** Connects *DIPENDENTE* and *INTERPRETA* with cardinalities (1,n) and (1,1).
 - Come:** Connects *GIOCATTORE* and *INTERPRETA* with cardinalities (1,n) and (1,1).
 - A:** Connects *EVENTO* and *PARTECIPA* with cardinalities (1,n) and (1,1).
 - Prende parte:** Connects *PERSONAGGIO* and *PARTECIPA* with cardinalities (1,n) and (1,1).
 - Possiede:** Connects *OGGETTI* and *PERSONAGGIO* with cardinalities (1,1) and (0,n).
 - Prerequisito:** Connects *ABILITÀ* and *ABILITÀ* with cardinalities (0,1) and (0,1).
- Other Elements:**
 - OGGETTI:** Attributes include *Valore* and *Quantità*.
 - Interpretation:** *OGGETTI* is a generalization of *ARMA*, *ARMATURA*, and *CONSUMABILE* under the constraint "Non Totale, Esclusiva".
 - Participation:** *PARTECIPA* is associated with *Numero_ticket* and *Turno*.

PROGETTAZIONE LOGICA:

A questo punto della progettazione è necessario trasformare lo schema ER in uno schema logico relazionale in modo che il DBMS sia in grado di operare sui concetti espressi durante la progettazione concettuale.

Eliminazione delle gerarchie:

Gerarchia su persona: si è scelto di collassare verso il basso la gerarchia, senza mantenere la relazione persona per evitare ridondanze, tenendo comunque conto di eventuali valori NULL su dipendente e giocatore

Gerarchia su campagna: si è scelto di collassare verso l'alto la gerarchia ed utilizzare dei valori NULL sull'attributo limite_partecipanti per le campagne estese ed integrare l'attributo anno_di_partenza anche sulle campagne oneshot. Inoltre si è aggiunto un attributo booleano oneshot per differenziare le due possibili tipologie di campagna.

Gerarchia su oggetto: abbiamo deciso di collassare la gerarchia verso l'alto e per evitare inutili valori a NULL mantenendo i punti_difesa ed i punti_attacco come DEFAULT 0. La descrizione verrà utilizzata su tutte le istanze.

Eliminazione chiavi esterne e selezione chiavi primarie:

Identificatore esterno su ABILITÀ: si è deciso di rimuovere l'identificatore esterno mantenendo l'unicità sulla chiave nome dell'abilità.

Auto-associazioni:

Auto-associazione su abilità: si è deciso di utilizzare un nuovo attributo che richiamasse il nome di un'abilità padre nel caso in cui si ritenesse necessario, diversamente viene tenuto a NULL.

Traduzione di entità e associazioni in schemi relazionali

dipendente(**cf**, cellulare, indirizzo, nome, cognome, stipendio)

giocatore(**cf**, cellulare, indirizzo, nome, cognome, email)

mondo(**nome**, tipo)

npc(**cod_npc**, nome, descrizione)

personaggio(**cod_pg**, nome, descrizione, exp, tipo)

missione(**cod_missione**, titolo, guadagno_exp, fallita)

campagna(**nome**, descrizione, anno_di_partenza, nome_mondo, oneshoot)

- FK: nome_mondo references mondo

evento(**titolo**, nome_campagna, data_inizio, data_fine, costo_iscrizione, indirizzo_location,

limite_partecipanti)

- FK: nome_campagna references campagna

avvenimento(**cod_avv**, nome_mondo, descrizione, data) -> data (del mondo)

- FK: nome_mondo references mondo

genera(**cod_avv, cod_missione**)

- FK: cod_avv references avvenimento
- FK: cod_missione references missione

assegna(**cod_npc, cod_missione**)

- FK: cod_npc references npc
- FK: cod_missione references missione

interpreta(**cf, titolo**, turno, cod_npc)

- FK: cf references dipendente
- FK: titolo references evento
- FK: cod_npc references npc

Intraprende(**cod_pg, cod_missione**)

- FK: cod_pg references personaggio
- FK: cod_missione references missione

partecipa(**cf, titolo**, nro_ticket, cod_pg)

- FK: cf references giocatore
- FK: titolo references evento
- FK: cod_pg references personaggio

interpreta(**cf, titolo**, turno, cod_npc)

- FK: cf references dipendente
- FK: titolo references evento
- FK: cod_npc references npc

oggetto(**nome, cod_pg**, valore, quantità, descrizione, punti_attacco, punti_difesa)

- FK: cod_pg references personaggio

abilità(**cod_pg, nome**, descrizione, restrizione, abilità_padre, costo_exp)

- FK: cod_pg references personaggio
- FK: abilità_padre references abilità

DATO DERIVATO

Il dato derivato è un dato che può essere ottenuto attraverso una serie di operazioni da altri dati, sulla base delle operazioni e delle loro frequenze è possibile valutare se è conveniente o meno mantenere nello schema attributi derivati.

Il dato derivato **exp** è un attributo di **personaggio** che serve per capire quanta esperienza ha attualmente accumulato quest'ultimo.

L'exp aumenta al conseguimento di una **missione** completata e diminuisce quando il personaggio spende punti per l'acquisto di nuove **abilità**.

Operazioni:

1. Lettura dei dati di personaggio (compresi i punti exp)
2. Completamento di una missione
3. Acquisto di una nuova abilità

Concetto	Tipo	Volume dati
Personaggio	E	5000
Missione	E	60000
Abilità	E	25000
Intraprende	R	120000
Dispone	R	25000

Operazione	Tipo	Frequenza
Op 1	I	250/GIORNO
Op 2	I	500/GIORNO
Op 3	I	100/GIORNO

Con dato derivato

Operazione	Concetto	Tipo	Accesso
1	Personaggio	L	1
2	Missioni	S	1

	Intraprende	S	2
	Personaggio	L	2
	Personaggio	S	2
3	Abilità	S	1
	Dispone	S	c
	Personaggio	L	1
	Personaggio	S	1

Operazioni:

1. $1L * 250/G = 250/\text{Giorno}$
2. $(5S * 2 + 2L) * 500/G = 6.000/\text{Giorno}$
3. $(3S * 2 + 1L) * 100/G = 700/\text{Giorno}$

TOT: 6.950/Giorno

Senza dato derivato

Operazione	Concetto	Accesso	Tipo
1	Personaggio	L	1
	Intraprende	L	24
	Missione	L	24
	Dispone	L	5
	Abilità	L	5
2	Missioni	S	1
	Intraprende	S	2
3	Abilità	S	1
	Dispone	S	1

Operazioni:

1. $59L * 250/G = 14.750/\text{Giorno}$
2. $3S * 2 * 500/G = 3.000/\text{Giorno}$
3. $2S * 2 * 100/G = 400/\text{Giorno}$

TOT: 18.150/Giorno

Conclusione: Risulta più conveniente mantenere il dato derivato **EXP**

QUERY DI CREAZIONE

```
CREATE TABLE DIPENDENTE (
    CF VARCHAR(15) PRIMARY KEY,
    CELLULARE VARCHAR(15) NOT NULL,
    INDIRIZZO VARCHAR(63) NOT NULL,
    NOME VARCHAR(63) NOT NULL,
    COGNOME VARCHAR(63) NOT NULL,
    STIPENDIO INTEGER NOT NULL
);

CREATE TABLE GIOCATORE (
    CF VARCHAR(15) PRIMARY KEY,
    CELLULARE VARCHAR(15) NOT NULL,
    INDIRIZZO VARCHAR(63) NOT NULL,
    NOME VARCHAR(63) NOT NULL,
    COGNOME VARCHAR(63) NOT NULL,
    EMAIL VARCHAR(63) NOT NULL
);

CREATE TABLE MONDO (
    NOME VARCHAR(15) PRIMARY KEY,
    TIPO VARCHAR(31)
);

CREATE TABLE CAMPAGNA (
    NOME VARCHAR(63) PRIMARY KEY,
    DESCRIZIONE VARCHAR(255) NOT NULL,
    ANNO_DI_PARTENZA INTEGER NOT NULL,
    NOME_MONDO VARCHAR(15) NOT NULL,
    ONESHOT BOOLEAN DEFAULT FALSE NOT NULL,
    FOREIGN KEY (NOME_MONDO) REFERENCES MONDO (NOME)
);

CREATE TABLE EVENTO (
    TITOLO VARCHAR(63) PRIMARY KEY,
    NOME_CAMPAGNA VARCHAR(15) NOT NULL,
    DATA_INIZIO DATE NOT NULL CHECK (DATA_INIZIO >= NOW()),
    DATA_FINE DATE NOT NULL CHECK (DATA_FINE >= NOW()),
    LIMITE_PARTECIPANTI INTEGER,
    COSTO_ISCRIZIONE INTEGER DEFAULT 0,
    INDIRIZZO_LOCATION VARCHAR(63) NOT NULL,
    FOREIGN KEY (NOME_CAMPAGNA) REFERENCES CAMPAGNA (NOME),
    CHECK (DATA_INIZIO <= DATA_FINE)
);
```

```

CREATE TABLE NPC (
    COD_NPC SERIAL PRIMARY KEY,
    NOME VARCHAR(31) NOT NULL,
    DESCRIZIONE VARCHAR(255) NOT NULL
);

CREATE TABLE PERSONAGGIO (
    COD_PG SERIAL PRIMARY KEY,
    NOME VARCHAR(31) NOT NULL,
    DESCRIZIONE VARCHAR(255) NOT NULL,
    EXP INTEGER DEFAULT 2600 NOT NULL, -- Exp DI DEFAULT ALLA
    CREAZIONE DEL PERSONAGGIO
    TIPO VARCHAR(31)
);

CREATE TABLE MISSIONE (
    COD_MISSIONE SERIAL PRIMARY KEY,
    TITOLO VARCHAR(31) NOT NULL,
    GUADAGNO_EXP INTEGER DEFAULT 0,
    FALLITA BOOLEAN DEFAULT NULL -- NULL = IN CORSO
);

CREATE TABLE INTERPRETA (
    CF VARCHAR(15) NOT NULL,
    TITOLO VARCHAR(63) NOT NULL,
    TURNO SMALLINT NOT NULL CHECK (TURNO BETWEEN 1 AND 7),
    COD_NPC INTEGER NOT NULL,
    PRIMARY KEY (CF, TITOLO),
    FOREIGN KEY (CF) REFERENCES DIPENDENTE (CF),
    FOREIGN KEY (TITOLO) REFERENCES EVENTO (TITOLO) ON DELETE
    CASCADE,
    FOREIGN KEY (COD_NPC) REFERENCES NPC (COD_NPC)
);

CREATE TABLE PARTECIPA (
    CF VARCHAR(15) NOT NULL,
    TITOLO VARCHAR(63) NOT NULL,
    NRO_TICKET SERIAL,
    COD_PG INTEGER NOT NULL,
    PRIMARY KEY (CF, TITOLO),
    FOREIGN KEY (CF) REFERENCES GIOCATORE (CF),
    FOREIGN KEY (TITOLO) REFERENCES EVENTO (TITOLO) ON DELETE
    CASCADE,
    FOREIGN KEY (COD_PG) REFERENCES PERSONAGGIO (COD_PG)
);

CREATE TABLE AVVENIMENTO (

```

```

        COD_AVV SERIAL PRIMARY KEY,
        NOME_MONDO VARCHAR(15) NOT NULL,
        DESCRIZIONE VARCHAR(255) NOT NULL,
        "DATA" VARCHAR(15) NOT NULL,
        FOREIGN KEY(NOME_MONDO) REFERENCES MONDO(NOME)
        ON DELETE CASCADE
    );

CREATE TABLE GENERA(
    COD_AVV INTEGER,
    COD_MISSIONE INTEGER,
    PRIMARY KEY(COD_AVV,COD_MISSIONE),
    FOREIGN KEY(COD_AVV) REFERENCES AVVENIMENTO(COD_AVV)
    ON DELETE CASCADE,
    FOREIGN KEY(COD_MISSIONE) REFERENCES
    MISSIONE(COD_MISSIONE)
    ON DELETE CASCADE
);

CREATE TABLE INTRAPRENDE(
    COD_PG INTEGER NOT NULL,
    COD_MISSIONE INTEGER NOT NULL,
    PRIMARY KEY(COD_PG,COD_MISSIONE),
    FOREIGN KEY(COD_MISSIONE) REFERENCES
    MISSIONE(COD_MISSIONE) ON DELETE CASCADE,
    FOREIGN KEY(COD_PG) REFERENCES PERSONAGGIO(COD_PG)
);

CREATE TABLE ASSEGNA(
    COD_NPC INTEGER NOT NULL,
    COD_MISSIONE INTEGER NOT NULL,
    PRIMARY KEY(COD_NPC,COD_MISSIONE),
    FOREIGN KEY(COD_MISSIONE) REFERENCES
    MISSIONE(COD_MISSIONE) ON DELETE CASCADE,
    FOREIGN KEY(COD_NPC) REFERENCES NPC(COD_NPC) ON DELETE
    CASCADE
);

CREATE TABLE OGGETTO(
    NOME VARCHAR(31) NOT NULL,
    COD_PG INTEGER NOT NULL,
    DESCRIZIONE VARCHAR(255) NOT NULL,
    VALORE INTEGER DEFAULT 0 NOT NULL,
    QUANTITA INTEGER DEFAULT 1 NOT NULL,
    PUNTI_ATTACCO INTEGER DEFAULT 0 NOT NULL,
    PUNTI_DIFESA INTEGER DEFAULT 0 NOT NULL,
    PRIMARY KEY(NOME,COD_PG),

```

```
FOREIGN KEY(COD_PG) REFERENCES PERSONAGGIO(COD_PG) ON  
DELETE CASCADE  
);
```

```
CREATE TABLE ABILITA(  
    NOME VARCHAR(31) NOT NULL,  
    COD_PG INTEGER NOT NULL,  
    DESCRIZIONE VARCHAR(255) NOT NULL,  
    RESTRIZIONE VARCHAR(15),  
    ABILITA_PADRE VARCHAR(31),  
    COSTO_EXP INTEGER NOT NULL DEFAULT 0,  
    PRIMARY KEY(NOME,COD_PG), -- KEY COMPOSTA  
    FOREIGN KEY(COD_PG) REFERENCES PERSONAGGIO(COD_PG) ON  
    DELETE CASCADE, -- RELAZIONATO AL PERSONAGGIO  
    FOREIGN KEY(ABILITA_PADRE,COD_PG) REFERENCES  
    ABILITA(NOME,COD_PG) ON DELETE CASCADE --SE PERDE UNA  
    ABILITA QUELLE FIGLIE MUOIONO  
);
```

Trigger e Stored Procedure

Procedura per l'iscrizione di un nuovo partecipante ad un evento, esegue un controllo sul superamento del limite dei partecipanti ed assegna il numero del ticket:

```
CREATE FUNCTION ISCRIZIONE(CF VARCHAR(15), EVENTO VARCHAR(63))
RETURNS INTEGER AS $$
DECLARE
    LIM INTEGER;
    TOT INTEGER;
    NRO_TICKET INTEGER;
BEGIN
    SELECT E.LIMITE_PARTICIPANTI INTO LIM FROM EVENTO AS E WHERE
E.TITOLO = EVENTO;
    SELECT COUNT(*) INTO TOT FROM PARTECIPA AS P WHERE P.TITOLO =
EVENTO;
    NRO_TICKET := TOT+1;

    IF LIM < TOT + 1 THEN
        RAISE EXCEPTION 'LIMITE PARTECIPANTI RAGGIUNTO';
    END IF;

    INSERT INTO PARTECIPA(cf,titolo,cod_pg,nro_ticket)
VALUES (CF,EVENTO,1,NRO_TICKET);

    RETURN NRO_TICKET;
END;
$$ LANGUAGE 'plpgsql';
-- ESEMPIO DI CHIAMATA
-- SELECT ISCRIZIONE('CD124','Gli antenati perduti');
```

Funzione per l'update dell'exp del personaggio a seguito dell'acquisto di un'abilità, esegue un ciclo con tutte le missioni completate intraprese dal personaggio e somma la quantità di exp parziale al suo totale in seguito prende la somma della spesa in quantitativo di exp per le sue abilità e la sottrae al totale se questa totale è inferiore a 0 allora lancia un'eccezione e non permette l'inserimento.

```
create OR REPLACE FUNCTION update_pg_exp_for_abilita() RETURNS
TRIGGER AS $$
declare
    exp_parziale integer;
    tot_exp integer;
    nro_part integer;
    spesa_exp integer;
    R integer;
begin
    tot_exp = 2600;--EXP ALLA CREAZIONE DEL PERSONAGGIO
    FOR R IN
        -- Loop delle missioni a cui il pg ha partecipato e che siano
completate
        SELECT M.COD_MISSIONE
        FROM INTRAPRENDE I, MISSIONE M
        where I.cod_pg = NEW.cod_pg
        AND M.FALLITA = FALSE
        AND M.COD_MISSIONE = I.COD_MISSIONE
        -- Aggiunta exp
        loop
            select count(*) into nro_part from INTRAPRENDE where
INTRAPRENDE.COD_MISSIONE = R;-- Numero di partecipanti per missione
            select GUADAGNO_EXP/nro_part into exp_parziale from MISSIONE
where MISSIONE.COD_MISSIONE = R ;
            tot_exp = tot_exp + exp_parziale;
        end loop;
        -- Riduzione exp
        select sum(costo_exp) into spesa_exp from abilita where cod_pg =
new.cod_pg;--Costo totale delle abilità del personaggio
        tot_exp = tot_exp - spesa_exp;

        --Update personaggio
        IF TOT_EXP < 0 THEN
            RAISE EXCEPTION 'EXP_INSUFFICIENTE';
        ELSE
            UPDATE PERSONAGGIO SET EXP = TOT_EXP WHERE COD_PG = NEW.COD_PG;
        END IF;
        RETURN NEW;
    END;
    $$ LANGUAGE 'plpgsql';
```

Trigger che viene eseguito quando si va a toccare la tabella abilità (inserimento cancellazione o modifica) chiama la funzione update_pg_exp_for_abilita su ogni row.

```
CREATE TRIGGER acquista_abilita
AFTER INSERT OR UPDATE OR DELETE ON abilita
FOR EACH ROW EXECUTE PROCEDURE update_pg_exp_for_abilita();
```

Funzione per l'update dell'exp del personaggio a seguito del completamento di una missione, se la missione ha stato iniziale false (completata) allora esegue un loop prendendo tutti i personaggi che hanno intrapreso la missione, prende il loro totale attuale di exp e lo aumenta per il guadagno di exp in questa missione.

```
create OR REPLACE FUNCTION update_pg_exp_for_missione() RETURNS
TRIGGER AS $$
declare
    exp_parziale integer;
    tot_exp integer;
    nro_part integer;
    spesa_exp integer;
    pg integer;
    R integer;
begin
    -- SE LA MISSIONE è COMPLETATA PROCEDO
    IF NEW.FALLITA = FALSE THEN
        FOR PG IN
            -- Loop dei personaggi che hanno partecipato alla missione
            completata
            SELECT I.COD_PG
            FROM INTRAPRENDE I
            where NEW.COD_MISSIONE = I.COD_MISSIONE
            -- Aggiunta exp
            loop
                SELECT EXP INTO tot_exp FROM PERSONAGGIO WHERE COD_PG = PG;--EXP
                ATTUALE DEL PERSONAGGIO
                select count(*) into nro_part from INTRAPRENDE
                where INTRAPRENDE.COD_MISSIONE = NEW.COD_MISSIONE;
                -- Numero di partecipanti per la missione completata
                select GUADAGNO_EXP/nro_part into exp_parziale from MISSIONE
                where MISSIONE.COD_MISSIONE = NEW.COD_MISSIONE;
                tot_exp = tot_exp + exp_parziale;
                --Update personaggio
                UPDATE PERSONAGGIO SET EXP = TOT_EXP WHERE COD_PG = PG;
            end loop;
        END IF;
        RETURN NEW;
    END;
$$ LANGUAGE 'plpgsql';
```


Trigger che viene eseguito quando si va a toccare la tabella abilità (inserimento cancellazione o modifica) chiama la funzione `completamento_missione` su ogni row.

```
CREATE TRIGGER completamento_missione
AFTER INSERT OR UPDATE OR DELETE ON missione
FOR EACH ROW EXECUTE PROCEDURE update_pg_exp_for_missione();
```

QUERY DI INSERIMENTO

Popolamento della tabella DIPENDENTE:

```
INSERT INTO DIPENDENTE
VALUES
('CF34','+3998676556','Via dei boschi 17','luca','longagnani',1200),
('CF89','+3998987087','Via dei ragazzi del 99','elena','vita',1000),
('CF12','+3905987698','Via degli ulivi
199','margherita','pratiche',1350),
('CF123','+395475475','Via dei partori
47','francesco','rugolo',890);
```

Popolamento della tabella GIOCATORE:

```
INSERT INTO GIOCATORE
VALUES
('CF1','+3923456','Via dei luogi
15','valentina','longo','foo@foo.it'),
('CD124','+31000329489','Via paolo
12','mario','rossi','bar@bar.it'),
('CH19','+3200293209','Via dei mario
16','giovanna','fusco','givo@gmail.com'),
('DF09','+3954728459','Via dei boschi
23','giovanni','fusco','pulcinella@infostrada.it'),
('CH1','+392350865123','Viale italia
189','stefano','amico','iphone@icloud.com'),
('KJ1','+3932747669988','Via dei marchi
12','giovanna','montanari','barone@gmail.com');
```

Popolamento della tabella MONDO:

```
INSERT INTO MONDO(NOME, TIPO)
VALUES
('Nirn','fantasy'),
('Pandora','fantascientifico');
```

Popolamento della tabella CAMPAGNA:

```
INSERT INTO CAMPAGNA(NOME, DESCRIZIONE, ANNO_DI_PARTENZA, NOME_MONDO,
ONESHOOT)
VALUES
('Avatar', 'Una campagna ricca di azione nei panni di alieni
giganteschi che si trovano a lottare per il proprio
pianeta.', 4600, 'Pandora', true),
('Skyrim', 'Una avventura senza paragoni, in un mondo di eroi, draghi
e non morti da sconfiggere.', 568, 'Nirn', false);
```

Popolamento della tabella EVENTO:

```
INSERT INTO
EVENTO(TITOLO, NOME_CAMPAGNA, DATA_INIZIO, DATA_FINE, LIMITE_PARTICIPANT
I, COSTO_ISCRIZIONE, INDIRIZZO_LOCATION)
VALUES
('Gli antenati perduti', 'Avatar', now(), now(), 20, 20, 'Via dei mille
99, San Giuliano PI'),
('Il nuovo imperatore', 'Skyrim', to_date('2021-1120',
'YYYY-MMDD'), to_date('2021-1123', 'YYYY-MMDD'), null, 60, 'Pascoli
albinetti presso villa de Paoli, Milano MI');
```

Popolamento della tabella NPC:

```
INSERT INTO NPC(NOME, DESCRIZIONE)
VALUES
('Grace Augustine', 'Dottoressa che studia gli abitanti ed i biomi
del pianeta Pandora.'),
('Jake Sully', 'Giovane soldato costretto ad una sedia a rotelle per
colpa di un incidente in guerra.'),
('Azura', 'Azura è il principe daedrico del tramonto e dell'alba e
uno dei pochi a non essere considerato intrinsecamente malvagio.'),
('Sanguine', 'Sanguine è il principe daedrico dell'edonismo, della
dissolutezza e delle oscure indulgenze. È più probabile che la
maggior parte dei principi si interessi agli affari mortali.');
```

Popolamento della tabella INTERPRETA:

```
INSERT INTO INTERPRETA(CF, TITOLO, TURNO, COD_NPC)
VALUES
('CF34', 'Gli antenati perduti', 1, 1),
('CF89', 'Gli antenati perduti', 1, 2),
('CF12', 'Il nuovo imperatore', 1, 3),
```

```
('CF123','Il nuovo imperatore',1,4);
```

Popolamento della tabella PERSONAGGIO:

```
INSERT INTO PERSONAGGIO (NOME, DESCRIZIONE, TIPO)
VALUES
-- skyrim
('Pascol','Soldato di razza imperiale abile nel combattimento corpo
a corpo.','guerriero'),
('Rinocy','Elfo oscuro nato per castare gli incantesimi più
forti.','mago'),
('Lane mart','Giovane orco lasciato orfano dalle guerre.','null'),
('Loroc','Bretone visto ed allegro un abile contadino.','null'),
('Xarad','Un Khajiiti nomade che ha perso il proprio gruppo ed è
alla ricerca di uno nuovo.','ladro'),
('Xyvir Amen','Una nobile elfa alta abile con arco e
frecce.','ranger'),
-- avatar oneshoot
('Xoror Iomor','Anziano guerriero che lotta per la propria
casa.','guerriero'),
('Xyam Umridad','Giovane Navi asperto nella caccia.','ranger'),
('Jhon wine','Guerriero scelto abile nel combattimento con robot da
guerra.','mech'),
('Dottor Much','Dottore che studia la razza Navi sul pianeta
pandora.','null');
```

Popolamento della tabella PARTECIPA:

```
SELECT ISCRIZIONE('CF1','Il nuovo imperatore',1);
SELECT ISCRIZIONE('CD124','Il nuovo imperatore',2);
SELECT ISCRIZIONE('CH19','Il nuovo imperatore',3);
SELECT ISCRIZIONE('DF09','Il nuovo imperatore',4);
SELECT ISCRIZIONE('CH1','Il nuovo imperatore',5);
SELECT ISCRIZIONE('KJ1','Il nuovo imperatore',6);
SELECT ISCRIZIONE('CF1','Gli antenati perduti',7);
SELECT ISCRIZIONE('CH19','Gli antenati perduti',8);
SELECT ISCRIZIONE('DF09','Gli antenati perduti',9);
SELECT ISCRIZIONE('KJ1','Gli antenati perduti',10);
```

Popolamento della tabella MISSIONE:

```
insert into missione(titolo,guadagno_exp)
values
('prima missione',150),
('seconda missione',150),
('terza missione',320),
('quarta missione',1200);
```

Popolamento della tabella INTRAPRENDE:

```
INSERT INTO INTRAPRENDE(COD_PG,COD_MISSIONE)
VALUES
(1,1),
(10,2),
(2,1),
(3,1),
(4,2),
(5,3),
(6,4),
(7,3),
(8,4),
(9,4);
```

Popolamento della tabella AVVENIMENTO:

```
INSERT INTO AVVENIMENTO(NOME_MONDO,DESCRIZIONE,"DATA")
VALUES
('Nirn', 'Il mago Rinocy diventa arcimago.','12-12-4600'),
('Pandora','Gli avatar vincono la battaglia.','03-01-568');
```

Popolamento della tabella ABILITÀ:

```
INSERT INTO ABILITA(NOME,COD_PG, DESCRIZIONE, RESTRIZIONE,
ABILITA_PADRE, COSTO_EXP)
VALUES
('armi corte', 1, 'Abilità che permette di utilizzare armi
corte.',null,null,800),
('armi lunghe', 1, 'Abilità che permette di utilizzare armi
lunghe.','guerriero','armi corte',800),
('magia', 2, 'Abilità che permette di utilizzare la
magia.','mago',null,1200),
('palla di fuoco', 2, 'Abilità che permette di utilizzare la magia
palla di fuoco.','mago','magia',600),
('furtività', 5, 'Abilità che permette di muoversi senza farsi
sentire.','ladro',null,900),
('armi da distanza', 6, 'Abilità che permette di utilizzare meglio
archi lunghi e corti.','ranger',null,1200),
('armature leggere', 7, 'Abilità che permette di utilizzare armature
leggere.',null,null,1000),
('armature pesanti', 7, 'Abilità che permette di utilizzare armature
pesanti.','guerriero','armature leggere',1600),
('macchine leggere', 9, 'Abilità che permette di utilizzare macchine
da guerra leggere.','mech',null,1000),
('macchine pesanti', 9, 'Abilità che permette di utilizzare macchine
da guerra pesanti.','mech','macchine leggere',1600);
```

Popolamento della tabella OGGETTO:

```
INSERT INTO OGGETTO(NOME, COD_PG, VALORE, QUANTITA, DESCRIZIONE,
PUNTI_ATTACCO, PUNTI_DIFESA)
VALUES
('pugnale',1,2,1,'pugnale corto ed affilato.',3,0),
('pugnale',2,2,1,'pugnale corto ed affilato.',3,0),
('pugnale',3,2,1,'pugnale corto ed affilato.',3,0),
('pugnale',4,2,1,'pugnale corto ed affilato.',3,0),
('pugnale',7,2,1,'pugnale corto ed affilato.',3,0),
('pugnale da guerriero',8,15,1,'pugnale Lungo ed ben
affilato.',7,1),
('bastone magico',2,50,1,'bastone magico.',20,0),
('cappa',2,5,1,'cappa per proteggersi dal sole.',0,3),
('corpetto leggero',5,10,1,'corpetto protettivo.',0,10),
('armatura in maglie leggera',6,20,1,'armatura leggera per arcieri e
cacciatori.',0,15),
('Ascia affilata',1,170,1,'Ascia adatta ad ogni evenienza.',65,0),
('arco corto',6,2,1,'arco corto e frecce.',10,0),
('pugnale',10,2,1,'pugnale corto ed affilato.',3,0),
('Macchina xYp01',9,20000,1,'Macchinario da guerra pesante adatto
contro ogni bestia che ospita pandora.',240,240),
('Siringhe di veleno',10,150,10,'siringhe con del veleno.',35,0),
('Spada corta',7,25,1,'spada corta arrugginita.',15,0),
('Armatura pesante in ferro',7,200,1,'armatura per veri
guerrieri.',0,30),
('cotta di maglia',8,25,1,'una armatura leggera e ben
decorata.',0,15),
('arco lungo',8,120,1,'arco lungo e frecce.',0,50),
('Veste da contadino',4,5,1,'veste che offre protezione dal sole
cocente.',0,5);
```

Popolamento della tabella ASSEGNA:

```
INSERT INTO ASSEGNA VALUES
(1,3),
(2,4),
(3,1),
(4,2);
```

Popolamento della tabella GENERA:

```
INSERT INTO GENERA VALUES
(1,1),
(2,4);
```

QUERY DI MODIFICA

- *Modificare numero di telefono del dipendente con codice fiscale 'CF12' indicando che l'attuale numero di cellulare è '+398764223443':*

```
update dipendente  
set cellulare = '+398764223443' where cf = 'CF12';
```

- *Modificare numero dei partecipanti in seguito ad ampliamento del locale fino ad un massimo di 40 per tutti gli eventi ospitati in Via dei mille 99, San Giuliano PI:*

```
update evento  
set limite_partecipanti = 40 where indirizzo_location = 'Via  
dei mille 99, San Giuliano PI';
```

- *Eliminare oggetto 'cotta di maglia' al personaggio con codice personaggio '8':*

```
delete from oggetto where nome = 'cotta di maglia' and cod_pg  
= '8';
```

QUERY DI INTERROGAZIONE

- *Selezionare giocatori che non hanno mai partecipato ad un evento:*

```
select * from giocatore where cf not in (select cf from partecipa);
```

- *Selezionare il personaggio con il maggior numero di oggetti nell'inventario:*

```
select cod_pg from oggetto group by cod_pg having count(*) >= all (select count(*) from oggetto group by cod_pg) ;
```

- *Selezionare la missione completata che ha dato piu punti esperienza:*

```
select * from missione where fallita = false and guadagno_exp >= (select max(guadagno_exp) from missione where fallita = false);
```

- *Mostrare l'indirizzo della location di tutti gli eventi che fanno parte di campagne oneshot:*

```
select titolo, indirizzo_location from evento where nome_campagna in (select nome_campagna from campagna where oneshoot = true);
```

- *Selezionare le abilità del personaggio con codice personaggio '1':*

```
select * from abilità where cod_pg = '1';
```

- *Selezionare tutte le abilità che possono essere acquisite solo se prima se ne possiede l'abilità padre ordinate per costo di punti esperienza:*

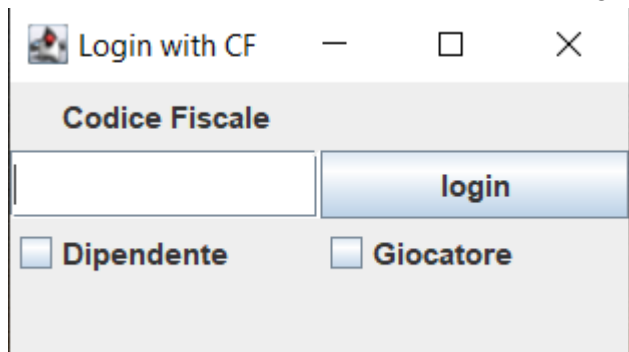
```
select * from abilita where abilita_padre != 'null' order by costo_exp;
```

- *Mostrare a quali eventi con quale NPC hanno partecipato i dipendente con stipendio > 1100 €:*

```
select d.cf,d.nome,d.cognome, n.nome, i.titolo from dipendente d, interpreta i, npc n where i.cf = d.cf and n.cod_npc = i.cod_npc and d.stipendio>1100;
```

JDBC

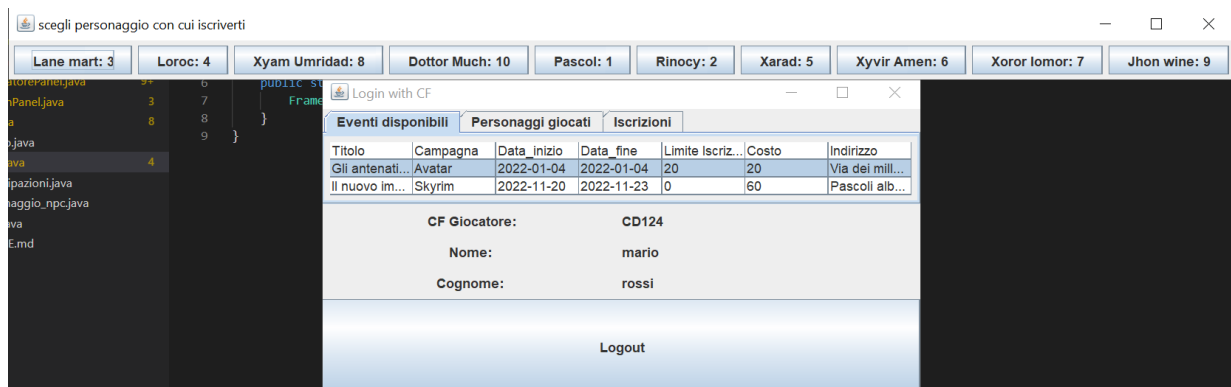
Come prima cosa è necessario eseguire il login nella seguente schermata, inserendo il Codice Fiscale e selezionando il ruolo dell'utente, se giocatore oppure dipendente.



A screenshot of a Java Swing window titled "Login with CF". It contains a text field for "Codice Fiscale", a "login" button, and two radio buttons labeled "Dipendente" and "Giocatore".

Se un giocatore decide di accedere come tale ed il CF è presente nel database il login avviene con successo, il giocatore ha la possibilità di visualizzare: gli eventi disponibili, i personaggi con cui ha giocato e le iscrizioni agli eventi.

Selezionando un evento in "Eventi disponibili" è possibile iscriversi al relativo evento selezionando il personaggio con il quale si ha intenzione di giocare, di cui è possibile vedere il nome ed il codice identificativo in alto nella schermata qui riportata. Per completare l'iscrizione ad un evento è necessario selezionare uno dei personaggi presenti, una volta fatto ciò il giocatore è iscritto all'evento ed è possibile visualizzare l'iscrizione nella finestra "Iscrizioni" (dopo aver effettuato un Logout e nuovamente un Login).



A screenshot of the main application window. At the top, there's a bar with buttons for various characters: Lane mart: 3, Loroc: 4, Xyam Umridad: 8, Dottor Much: 10, Pascol: 1, Rinocy: 2, Xarad: 5, Xyvir Amen: 6, Xoror Iomor: 7, Jhon wine: 9. Below this is a "scegli personaggio con cui iscriverti" section. The main area has three tabs: "Eventi disponibili", "Personaggi giocati", and "Iscrizioni". The "Eventi disponibili" tab is active, showing a table with columns: Titolo, Campagna, Data_inizio, Data_fine, Limite Iscriz..., Costo, and Indirizzo. The table contains two rows of event data. Below the table, there's a section for the selected player: "CF Giocatore: CD124", "Nome: mario", and "Cognome: rossi". At the bottom is a "Logout" button.

Titolo	Campagna	Data_inizio	Data_fine	Limite Iscriz...	Costo	Indirizzo
Gli antenati...	Avatar	2022-01-04	2022-01-04	20	20	Via dei mill...
Il nuovo im...	Skyrim	2022-11-20	2022-11-23	0	60	Pascoli alb...

Se un dipendente decide di accedere come tale ed il CF è presente nel database il login avviene con successo, il dipendente ha la possibilità di visualizzare: tutti gli eventi (passati e futuri) e gli NPC da lui interpretati.

Il dipendente ha la possibilità di aggiungere un nuovo evento selezionando l'apposito tasto, così facendo può inserire tutti i dati del nuovo evento nella finestra di creazione dell'evento, facendo attenzione a non violare il vincolo di ForeignKey per il quale il nome della campagna deve corrispondere al nome di una campagna già esistente, se i dati sono corretti è possibile visualizzare l'evento inserito (dopo aver effettuato un Logout e nuovamente un Login) nella finestra "Eventi".

Eventi		NPC interpretati				
Titolo	Campagna	Data_inizio	Data_fine	Limite iscriz...	Costo	Indirizzo
Gli antenati...	Avatar	2022-01-04	2022-01-04	20	20	Via dei mill...
Il nuovo im...	Skyrim	2022-11-20	2022-11-23	0	60	Pascoli alb...
Aggiungi nuovo evento						
CF Giocatore:		CF34				
Nome:		luca				
Cognome:		longagnani				
Logout						

Titolo:	<input type="text"/>
Campagna:	<input type="text"/>
Data Inizio:	AAAA-MMGG
Data Fine:	AAAA-MMGG
Limite Iscritti:	<input type="text"/>
Costo:	<input type="text"/>
Indirizzo:	<input type="text"/>
aggiungi	

QUERY DI INTERROGAZIONE

JDBC:

- Selezione tupla di un giocatore/dipendente dato il cf al login dell'applicazione (a seconda del login selezionato):

```
SELECT * from giocatore where cf = 'CF1' ; -- CF1 è un codice fiscale di
esempio tra i dati reali
```

```
SELECT * from dipendente where cf = 'CF34' ;
```

- Selezione degli eventi disponibili che si terranno per un giocatore, selezione di tutti gli eventi per un dipendente:

```
SELECT * from evento where data_inizio > current_date order
by data_inizio;
```

```
SELECT * from evento
```

- Selezione dei personaggi giocati per giocatore , npc interpretati per dipendente:

```
SELECT * from personaggio pe , partecipa pa where
pe.cod_pg=pa.cod_pg and cf =
'CF1' ;
```

```
SELECT * from npc n , interpreta i where n.cod_npc = i.cod_npc
and cf = 'CF34' ;
```

- *Inserimento nuovo evento nel pannello dipendente:*

```
insert into evento values ('titolo','campagna',  
to_date('data_inizio'),to_date('data_fine'),lim_part,costo,'in  
dirizzo');
```

- *Iscrizione ad un evento (per giocatore, partecipando come un certo personaggio)
chiamando la procedura iscrizione():*

```
SELECT iscrizione('cf','evento',cod_pg);
```

CODICE JAVA PER UTILIZZO JDBC:

(dopo aver importato i driver giusti per postgresql come archivio jar)

```
Class.forName("org.postgresql.Driver");  
c = DriverManager.getConnection(url,usr,psw); //oggetto  
connessione al db tramite url del db (postgresql) username e password di tale db  
  
st = this.c.createStatement(); // oggetto statement che permette  
esecuzione sql  
String query = "una certa query";  
  
// per query di interrogazione come di seguito  
ResultSet res = st.executeQuery(query); //tuple risultanti salvate in  
ResultSet  
while(res.next()) { ... } // chiamando next viene restituita  
prossima tupla  
res.close();  
//per query di modifica  
st.executeUpdate(query);
```