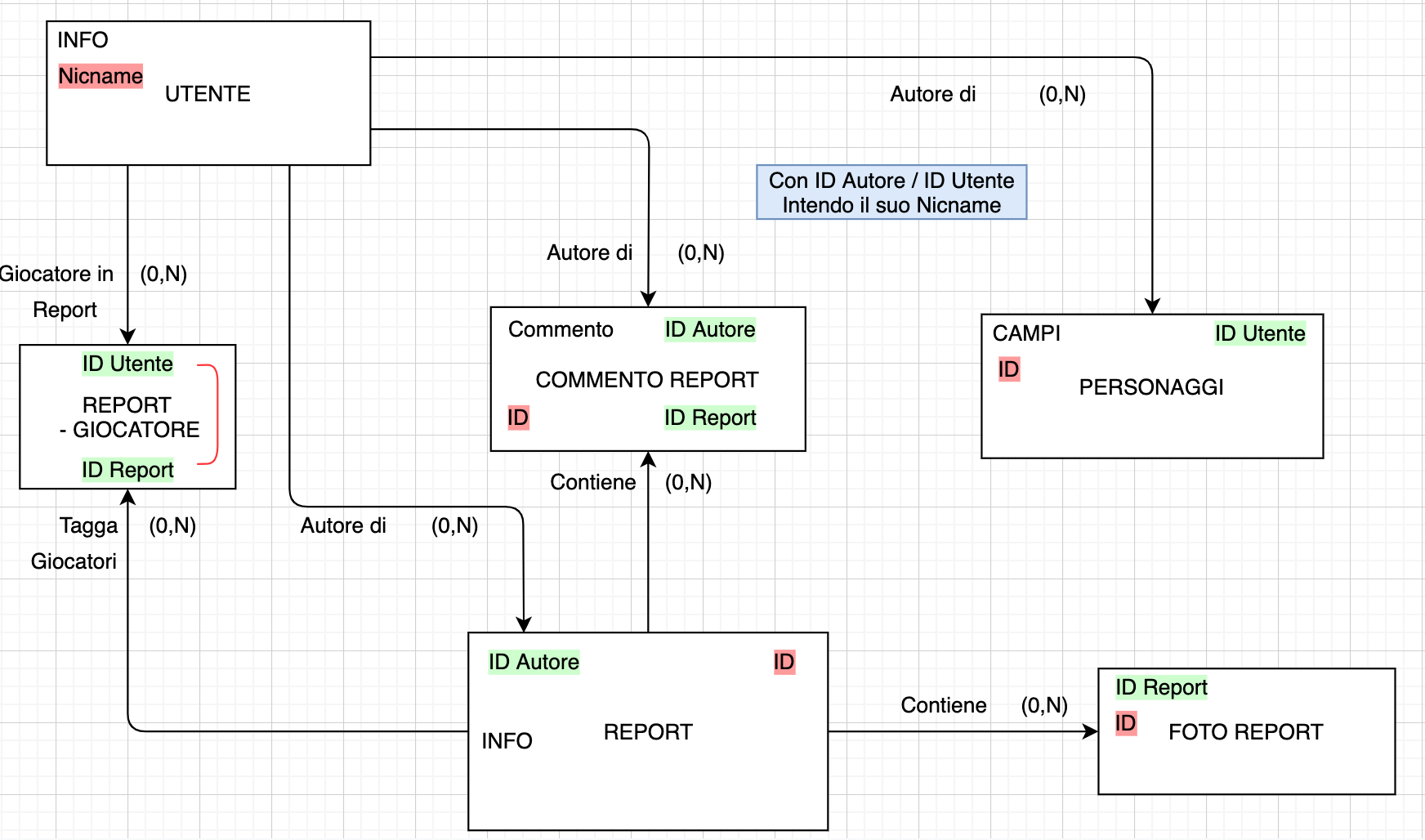
Database

**ATTENZIONE**: aggiungere il campo **Nicname** a Utente, è quello l’identificativo univoco, è ciò che ci consentirà di avere “user” come possibile campo di accesso invece che la mail.

=> Quando modifico le informazioni utente o quando faccio un Iscrizione il sistema deve verificare che sia univoco

Classe: Interfaccia DB

Schede Personaggio

**DatiPers** = {Nome + classe+ .. +traits + ideals +..} Evito di dare l’ ID\_Pers perchè presumo di avercelo essendo passato necessariamente dalla sua card

DatiPers getPersonaggioDati( ID\_pers )

Void setPersonaggioDati( ID\_pers , DatiPers )

Per le modifiche su un determinato Personaggio (gli attributi vengono interessati sempre assieme)

Void addPersonaggio( DatiPers )

Realizzato con aggiungi tupla con ID\_pers generato + setPersonaggioDati( ID\_pers, DatiPers)

Array<DatiPcorti> getPersonaggiDatiCards ( Nic\_user )

**DatiPcorti** = {ID\_pers + Nome + Razza + Classe + Allineamento} . Mi restituisce un array dei personaggi posseduti da user, per creare le cards, nell’area Personale

Void deletePersonaggio( ID\_pers )

Attenzione a eleminare tutti i riferimenti

Report Sessione

**DatiRepo** = { {Nic\_autore+img\_autore}+titolo+sottotitolo+contenuto+{ {Nic\_giocatore1+img\_giocatore1}+{Nic\_giocatore2+img\_giocatore2} +..} + { IDUrl\_img1+IDUrl\_img2+…} + isEsplorabile }.

Evito di ritrasmettere l’ ID\_Repo perchè presumo di avercelo essendo passato necessariamente dalla sua card.

DatiRepo getRepoDati( ID\_repo )

Array<DatiRCorti> getRepoDatiCards( {IDs\_repo} )

Per ogni ID in {IDs\_repo} Comporre un DatiRCorti da mettere in 1 Array

**DatiRCorti** = { **ID\_Repo** + titolo + sottotitolo + numero\_di\_giocatori + isEsplorabile + {Nic\_autore+img\_autore} }

Il # di giocatori è un valore **ricavato** con un altra quary o dalla stessa

Questo metodo verrà spesso usato subito dopo aver chiamato una delle query per trovare gli ID dei report che mi interssano o anche direttamente con un ID specifico (x ricerche da casella di ricerca)

**QUARY X RICAVARE LE CARDS GIUSTE**

{IDs\_repo}+NumP getIDsRepo(enum<TypeDashB>, Nic\_User, i )

Se non specificato l’ordine dei ID è: data creazione Decrescente

**TypeDashB** indica il tipo di dashboard (MyDashbor, ImPlayer, Esplora)

**NumP** indica il numero di pagine che possono essere create con il numero di cards persenti nella data dashboard, tenendo conto che una dashboard ha max 5 card => se ho 7 cards return 2 , se ho 0 cards return 0.

**Parametro “i”** : Restituisce i report che vanno dalla posizione 5\*i alla pos. 5\*i+4 , serve perchè le varie dashboard hanno solo 5 report a botta.

**ATTENZIONE**: Da tenere presente che se TypeDashB = Esplora => Nic\_User può essere NULL.

Void setRepoDati( ID\_repo , DatiRepo )

Nella prima istruzione va eseguito in serie : ID\_img addImmagine(file\_i) per le N immagini che si sono aggiunte

Void addReport( DatiRepo )

Realizzato con aggiungi tupla con ID\_repo generato + setRepoDati( ID\_repo, DatiRepo) e tutte le altre cose iniziali

Void setEsplorabile( ID\_repo)

Nega il valore booleano Dell attributo isEsplorabile del report

Void deleteRepo( ID\_repo )

Attenzione andare a cancellare anche i riferimenti nelle altre entità

Commenti

**DatiCommenti** = { ID\_Commento+Commento+{Nic\_autore+img\_autore} }

Array<{DatiCommenti}> getCommenti(ID\_repo)

Void addCommento(Nic\_Autore, Commento, ID\_Report)

Void deleteCommento(ID\_Commento)

Utenti

**DatiAnagrUtente** = { **Nicname**+img+{altre\_info\_personali} }

Bool verificaEsistenzaNic(Nicname)

Per la creazione serve un Nic Univoco

Void creaUtente(DatiAnagrUtente,Password)

DatiAnagrUtente getUtente(Nicname)

Void setDatiAnagrUtente (DatiAnagrUtente)

Bool verificaPassword(Nicname, pswdProva)

La password così resta sempre all’interno del DB

Void setPassword(nuovaPassword)

Immagini Report

{ID\_img} getMediaGallery ( ID\_Repo )

ID\_img addImmagine( file ) -> genera ID

Void deleteImmagine( ID )

File getVisualImmagine ( ID\_img)

Altri Metodi

Connection OpenConnection()

Void CloseConnection()