**UNIVERSITÀ DEGLI STUDI DI NAPOLI “FEDERICO II”**

**DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELLE TECNOLOGIE DELL’INFORMAZIONE**

Immagine che contiene schizzo, cerchio, simbolo, fronte

Descrizione generata automaticamente

**CORSO DI LAUREA IN INFORMATICA**

**INSEGNAMENTO DI BASI DI DATI**

Progettazione di una base di dati per lo sviluppo

del Social Network “UninaSocialGroup”

Autori: Docente:

Vincenzo De Rosa N86004524 Prof. Mara Sangiovanni

Luigi Bove N86004760

Krystian Catewicz N86004708

Anno Accademico 2023/2024

**Indice**

[1.Descrizione del Progetto 5](#_Toc156635587)

[1.1 Traccia Assegnata 5](#_Toc156635588)

[1.2 Analisi del problema e scelte effettuate 5](#_Toc156635589)

[2. Progettazione Concettuale 7](#_Toc156635590)

[2.1 Modello UML 7](#_Toc156635591)

[2.2 Modello Entità-Relazioni 8](#_Toc156635592)

[3 Ristrutturazione 9](#_Toc156635593)

[3.1.1 Gestione delle Classi di Associazione 9](#_Toc156635594)

[3.1.2 Gestione delle Gerarchie di Specializzazione 9](#_Toc156635595)

[3.1.3 Gestione degli identificativi 9](#_Toc156635596)

[3.2 Modello UML ristrutturato 10](#_Toc156635597)

[3.3 Dizionario delle Classi 11](#_Toc156635598)

[3.4 Dizionario delle Associazioni 13](#_Toc156635599)

[3.5 Dizionario dei Vincoli 15](#_Toc156635600)

[4. Modello Logico 17](#_Toc156635601)

[5. Schema Fisico 19](#_Toc156635602)

[5.1 Definizione tabelle in SQL 19](#_Toc156635603)

[5.1.1 Definizione tabella Utente 19](#_Toc156635604)

[5.1.2 Definizione tabella Gruppo 19](#_Toc156635605)

[5.1.3 Definizione tabella Richiesta 20](#_Toc156635606)

[5.1.4 Definizione tabella Post 20](#_Toc156635607)

[5.1.5 Definizione tabella Interazione 21](#_Toc156635608)

[5.1.6 Definizione tabella Notifica 21](#_Toc156635609)

[5.1.7 Definizione tabella Iscrizione 22](#_Toc156635610)

[5.1.8 Definizione tabella Amministra 22](#_Toc156635611)

[5.2 Trigger e Procedure SQL 23](#_Toc156635612)

[5.2.1 Definizione vincolo “noTestoInLike” 23](#_Toc156635613)

[5.2.2 Definizione vincolo “coerenzaDataPost” 24](#_Toc156635614)

[5.2.3 Definizione vincolo “coerenzaDataInterazione” 25](#_Toc156635615)

[5.2.4 Definizione Vincolo “coerenzaDataNotificaPost 26](#_Toc156635616)

[5.2.5 Definizione Vincolo “coerenzaDataNotificaInterazione” 27](#_Toc156635617)

[5.2.6 Definizione Vincolo “coerenzaDataRichiesta” 28](#_Toc156635618)

[5.2.7 Definizione Vincolo “esitoTrueRichiesta” 29](#_Toc156635619)

[5.2.8 Definizione Vincolo “generaNotificaAccesso” 30](#_Toc156635620)

[5.2.9 Definizione Vincolo “generaNotificaPost” 31](#_Toc156635621)

[5.2.10 Definizione Vincolo “generaNotificaInterazione” 32](#_Toc156635622)

# 1.Descrizione del Progetto

## 1.1 Traccia Assegnata

UninaSocialGroup è un social network che offre agli utenti la possibilità di creare e partecipare a gruppi tematici, descritti da tag (es: sport, fantasy, anime, ecc.). Una volta accettati dal creatore del gruppo gli utenti possono condividere i loro pensieri, esperienze o interessi relativi alla categoria del gruppo pubblicando contenuti come foto e\o testo. Gli altri membri iscritti al gruppo hanno la possibilità di interagire con i contenuti attraverso commenti e “like”. All’inserimento di un nuovo contenuto tutti gli iscritti al gruppo ricevono una notifica. Si utilizzino le proprie conoscenze del dominio per definire dettagli non specificati nella traccia.

Aggiungere la possibilità di gestire le notifiche, un utente riceve una notifica ogni volta che:

• un utente accede ad un gruppo di cui si è il creatore;

• un utente ha creato un nuovo contenuto ad un gruppo di cui si è amministratori;

• un utente ha interagito ad un contenuto di cui si è autori.

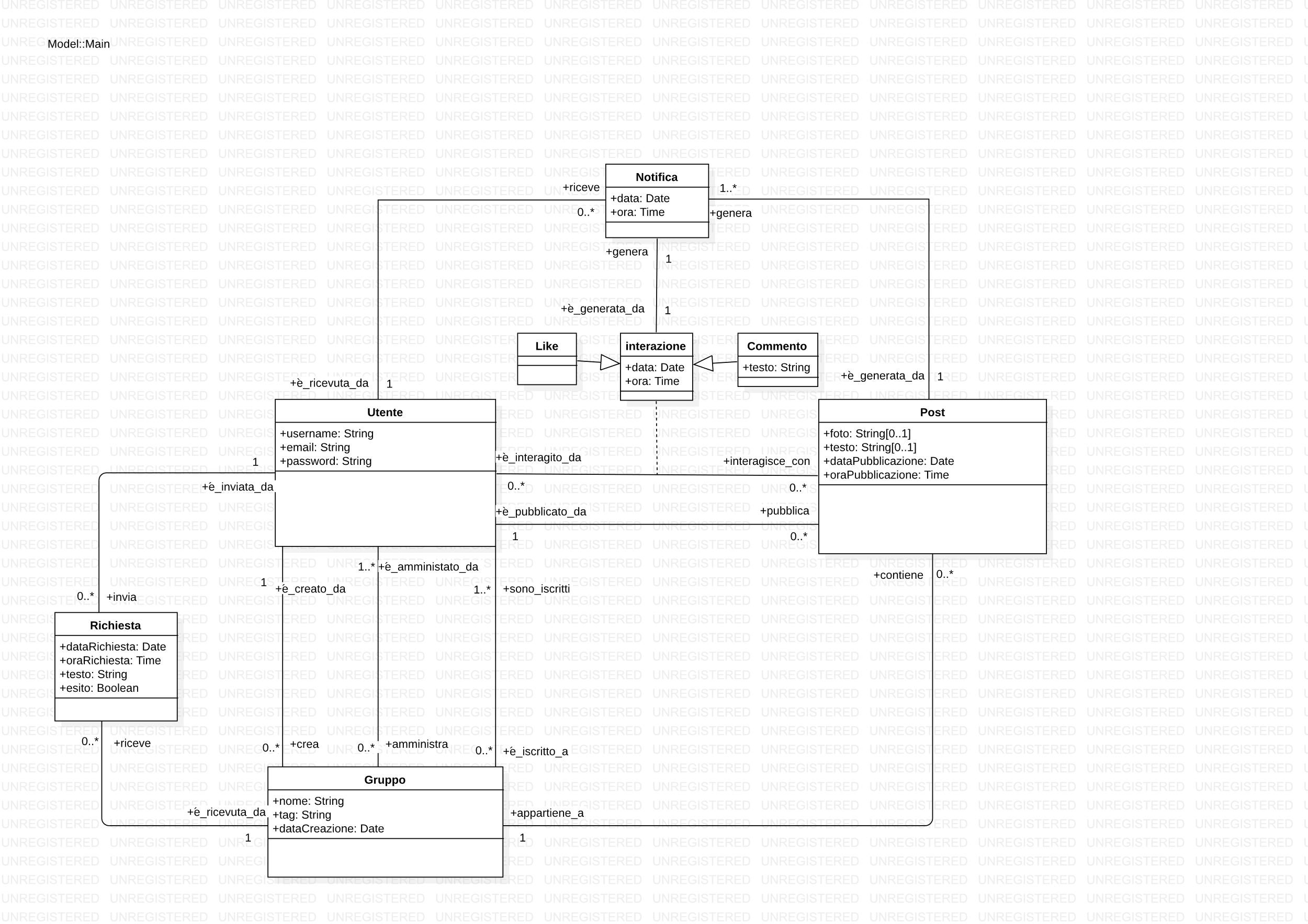
## 1.2 Analisi del problema e scelte effettuate

Si è scelto di modellare le seguenti entità del Minimondo descritto dalla traccia:

* **Utente**: è l’attore principale del dominio da realizzare. Di esso si tiene in considerazione l’**username**, l’**e-mail** e la **password.** Si è deciso di non creare delle specializzazioni di utente (per differenziare tra creatore, amministratore e membro di un gruppo) ma di arrivare a tali informazioni attraverso le relazioni che collegano utente all’entità gruppo.
* **Gruppo**: viene creato e gestito da utenti. Diversi utenti possono iscriversi ad un gruppo e un utente può iscriversi a più gruppi. Le informazioni memorizzate di un gruppo sono: **nome, tag** (tema principale del gruppo) e **data creazione**.
* **Post:** è l’entità che raccoglie i contenuti condivisi da un utente all’interno di un gruppo. Essi possono essere testuali o contenere un’immagine. Le informazioni memorizzate di un post sono: **foto, testo, data** e **ora di pubblicazione.**
* **Notifica:** viene generata dai post pubblicati in un gruppo oppure dalle interazioni di un utente (like o commenti) con un post. Di una notifica si tiene traccia delle seguenti informazioni: **data** ed **ora.**
* **Interazione:** entità che contiene le informazioni di un’interazione di un utente ad un post. Viene specializzata in altre due entità: **like** e **commento**. Le informazioni memorizzate sono: **data, ora** e **testo** (quest’ultima contenuta solo nella specializzazione commento)
* **Richiesta:** viene inviata da utenti ad un gruppo per richiedere la partecipazione ad esso. Le informazioni memorizzate sono: **data** e **ora** in cui viene effettuata la richiesta, **testo** che indica le motivazioni della richiesta ed **esito** (che di default avrà valore falso)

# 2. Progettazione Concettuale

## 2.1 Modello UML



## 2.2 Modello Entità-Relazioni



# 3 Ristrutturazione

### 3.1.1 Gestione delle Classi di Associazione

L’unica Classe di Associazione presente nello schema concettuale è l’associazione “interazione”. L’associazione interazione è stata quindi rimodellata in un’entità, di cui è stata ristrutturata anche la specializzazione (come viene descritto in seguito).

### 3.1.2 Gestione delle Gerarchie di Specializzazione

L’unica Gerarchia di Specializzazione è quella definita su “Interazione”, le cui specializzazioni sono “Like” e “Commento”. La specializzazione di interazione è stata eliminata accorpando le specializzazioni "Like" e "Commento" nella classe generale "Interazione". Per poter distinguere tra interazione di tipo Like e interazione di tipo Commento è stata definita un’enumerazione “enum\_interazione”. Il motivo di tale scelta è dovuto al fatto che le specializzazioni aggiungono poche informazioni alla classe generale e di conseguenza adottare questa strategia risulta essere poco pesante in fase di progettazione e definizione di nuovi vincoli. Inoltre essendo la specializzazione di tipo Totale/Disgiunto sono necessari solo due tipi nell’enumerazione.

Vengono definiti inoltre i seguenti vincoli (che vengono trattati nel dettaglio nel Dizionario dei Vincoli):

-Interazione DEVE essere Like O Commento. (Totale/Disgiunta)

-Se interazione è Like allora l'attributo testo è NULL.

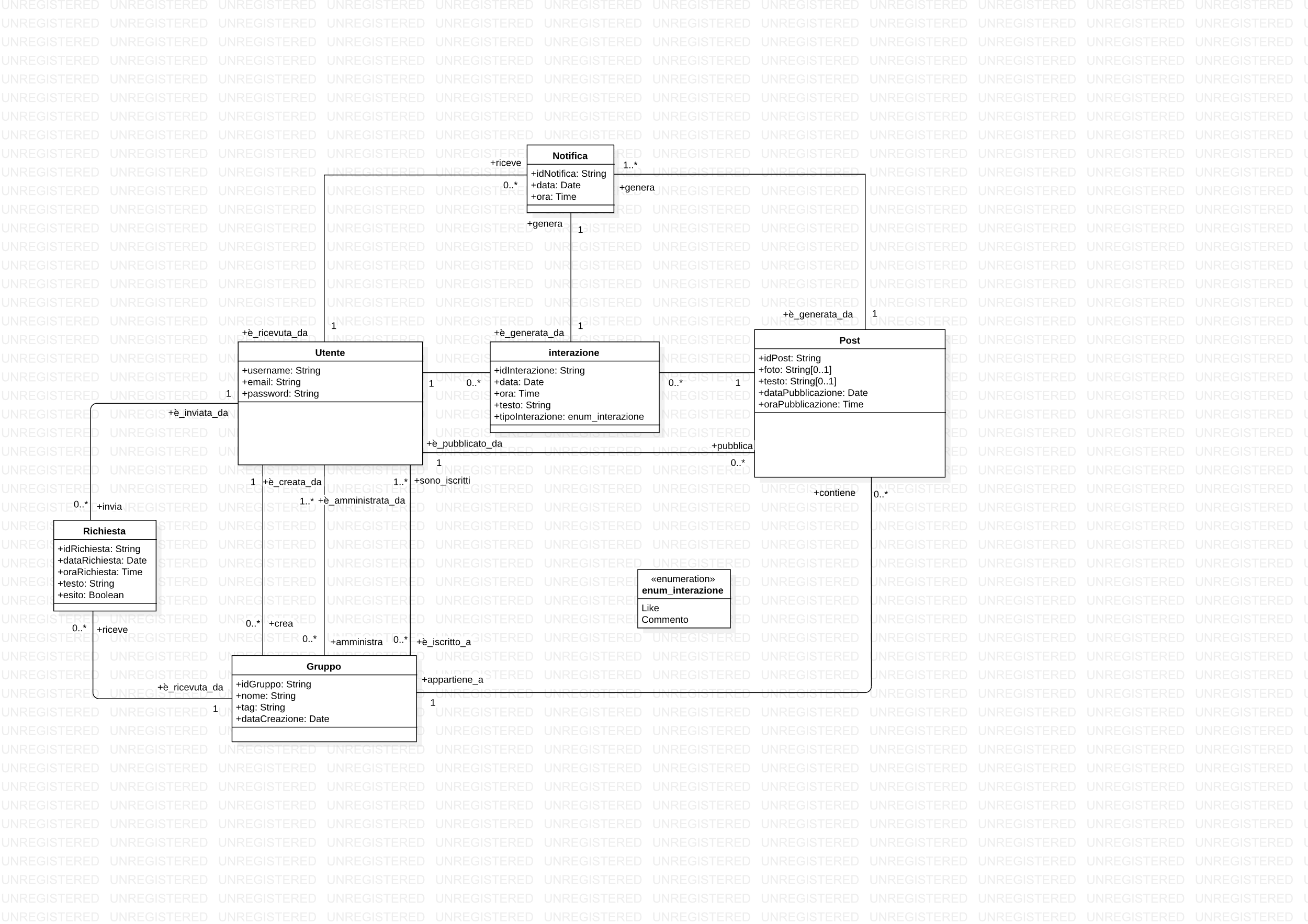
### 3.1.3 Gestione degli identificativi

Per tutte le entità, ad eccezione di “Utente”, sono state definite chiavi primarie surrogate per poter identificare univocamente le istanze delle entità. Questa scelta permette di ricercare un determinato record in maniera più immediata e semplice.

È stata adottata la convenzione secondo la quale una chiave primaria è definita come idNomeEntità.

Per l’entità “Utente” è stata scelta come chiave primaria l’attributo “username”.

## 3.2 Modello UML ristrutturato



## 3.3 Dizionario delle Classi

|  |  |  |
| --- | --- | --- |
| **Classe** | **Descrizione** | **Attributi** |
| Utente | Descrive tutti gli utenti | -**username**(String): indica l’username dell’utente (chiave primaria)  -**email**(String): indica l’email associata all’account dell’utente  -**password**(String): indica la password associata all’account dell’utente |
| Interazione | Descrive le interazioni di ogni utente con i post di altri utenti | -**idInterazione**(String): chiave primaria che identifica ciascuna interazione  -**data**(Date): indica la data in cui è stata fatta l’interazione  -**ora**(Time): indica l’ora in cui è stata fatta l’interazione  -**testo**(String): testo identificativo dell’interazione  **-idInterazione** (enum\_interazione): indica il tipo di interazione fatta |
| Notifica | Descrive la notifica generata da un’interazione | -**idNotifica**(String): chiave primaria che identifica ciascuna notifica  -**data**(Date): indica la data in cui è stata generata la notifica  -**ora**(Time): indica l’ora in cui è stata generata la notifica |
| Post | Descrive i post pubblicati da ogni utente | -**idPost**(String): chiave primaria che identifica ciascun post  **-foto**(String[0..1]): indica un’eventuale foto caricata nel post  **-testo**( String[0..1]): indica un eventuale testo aggiungo al post  **-dataPubblicazione**(Date): indica la data di pubblicazione del post  **-oraPubblicazione**(Time): indica l’ora di pubblicazione del post |
| Richiesta | Descrive per ogni utente le richieste di partecipazione inviate ai vari gruppi | -**idRichiesta**(String): chiave primaria che identifica ciascuna richiesta  -**dataRichiesta**(Date): indica la data in cui è stata inviata la richiesta  -**oraRichiesta**(Time): indica l’ora in cui è stata inviata la richiesta  -**testo**(String): indica un testo aggiunto dall’utente alla richiesta inviata  -**esito**(Boolean): indica l’esito della richiesta inviata |
| Gruppo | Descrive ogni gruppo creato dove possono accedere utenti | -**idGruppo**(String): chiave primaria che identifica ciascun gruppo  -**nome**(String): indica il nome del gruppo  -**tag**(String): indica i tag rappresentativo del gruppo  -**dataCreazione**(Date): indica la data in cui è stato creato il gruppo |
| Enum\_interazione | Descrive il tipo di interazione che è stata prodotta | -**Like**  **-Commento** |

## 3.4 Dizionario delle Associazioni

|  |  |  |
| --- | --- | --- |
| Associazione | Descrizione | Classi Coinvolte |
| Crea | Esprime la creazione di uno o più gruppi da parte di un utente | **Utente[1]** ruolo (**crea):** indica l’utente che crea il gruppo  **Gruppo[0..\*]** ruolo (**è\_creato\_da):**  indica i gruppi che vengono creati dall’utente |
| Amministra | Esprime l’amministrazione di un gruppo da parte di un utente | **Utente[0..\*]** ruolo (**amministra):** indica l’utente che amministrano il gruppo  **Gruppo[0..\*]** ruolo (**è amministrato da):**  indica i gruppi che vengono amministrati dall’utente |
| Iscrizione | Esprime l’iscrizione di un utente ad un gruppo | **Utente[0..\*]** ruolo (**è iscritto a):** indica l’utente che è iscritto ad un gruppo  **Gruppo[0..\*]** ruolo (**sono\_iscritti**): indica il gruppo a cui è iscritto l’utente |
| Invia | Esprime l’invio di una richiesta da parte di un utente | **Utente[1]** ruolo (**Invia**): indica l’utente che invia la richiesta  **Richiesta[0..\*]** ruolo (**è inviata da):** indica la richiesta che è stata inviata dall’utente |
| Riceve | Esprime la ricezione di una richiesta di un gruppo | **Richiesta[0..\*]** ruolo (**è ricevuta da**): indica la richiesta ricevuta dal gruppo  **Gruppo[1]** ruolo (**riceve**): indica il gruppo che riceve la richiesta |
| Appartiene | Esprime l’appartenenza di un post ad un gruppo | **Gruppo[1]** ruolo (**contiene**): indica il gruppo contenente il post  **Post[0..\*]** ruolo (**appartiene a**): indica il post che appartiene al gruppo |
| Pubblica | Esprime la pubblicazione di un post da parte di un utente | **Utente[1]** ruolo (**pubblica**): indica l’utente che pubblica un post  **Post[0..\*]** ruolo (**è pubblicato da**): indica il post che è pubblicato da un utente |
| Riceve | Esprime la ricezione di una notifica da parte di un utente | **Utente[1]** ruolo (**riceve**): indica l’utente che riceve la notifica  **Notifica[0..\*]** ruolo (**è ricevuta da**): indica la notifica che viene ricevuta dall’utente |
| Interagisce | Esprime il rilascio di un’interazione da parte dell’utente | **Utente[1]:** ruolo (**interagisce**): indica l’utente che rilascia l’interazione  **Interazione[0..\*]**  ruolo (**è rilasciata da**): indica l’interazione che viene rilasciata dall’utente |
| Appartiene | Esprime l’appartenenza di un’interazione ad un post | **Interazione[0..\*]:** ruolo (**appartiene a**) indica l’interazione che appartiene al post  **Post[1]** ruolo (**contiene**): indica il post a cui appartiene l’interazione |
| Genera | Esprime la generazione di una notifica da parte di un’interazione | **Interazione[1]** ruolo (**genera**): indica l’interazione che genera la notifica  **Notifica[1]** ruolo (**è generata da**): indica la notifica che viene generata dall’interazione |
| Genera | Esprime la generazione di una notifica quando viene pubblicato un post | **Notifica[1..\*]** ruolo (**è generata da**): indica la notifica che viene generata dal post  **Post[1]** ruolo (**genera**): indica il post che genera la notifica |

## 3.5 Dizionario dei Vincoli

|  |  |  |
| --- | --- | --- |
| Nome vincolo | Tipo Vincolo | Descrizione |
| noTestoInLike | N-pla | Se **tipoInterazione** è **like** allora **testo** è NULL |
| uniqueNomeGruppo | Intrarelazionale | Il **nome** di un **Gruppo** è univoco |
| uniqueEmail | Intrarelazionale | L’**email** di un **utente** è univoca |
| postNonVuoto | Intrarelazionale | In **post** gli attributi **foto** e **testo** non possono essere entrambi NULL. |
| coerenzaDataPost | Interrelazionale | La **data di pubblicazione** di un **post** deve essere successiva alla **data creazione** del **gruppo** in cui è contenuto il post |
| coerenzaDataInterazione | Interrelazionale | La **data** e l’**ora** di un’ interazione devono essere successive alla **data pubblicazione** e all’**ora pubblicazione** di un post |
| coerenzaDataNotificaPost | Interrelazionale | La **data** e l’**ora** di una **notifica** generata da un **post** devono essere uguali a **data** e **ora** di pubblicazione del post generante. |
| coerenzaDataNotificaIntera-  zione | Interrelazionale | La **data** e l’**ora** di una **notifica** generata da un’**interazione** devono essere uguali a **data** e **ora** di creazione dell’interazione generante. |
| coerenzaDataRichiesta | Interrelazionale | La **data** e l’**ora** di una **richiesta** ad un **gruppo** devono essere successive alla **data** e all’**ora** di creazione del gruppo |
| dominioTipoInterazione | Dominio | L’attributo **tipoInterazione** di un’**interazione** deve essere **like** o **commento** |
| esitoTrueRichiesta | Interrelazionale | Quando l’**esito** di una **richiesta** viene aggiornato a true viene creata una nuova istanza di iscrizione tra l’**utente** e il **gruppo** della richiesta |
| generaNotificaAccesso | Interrelazionale | Quando un nuovo utente accede ad un gruppo viene generata una nuova istanza di **notifica** per il creatore del gruppo |
| generaNotificaPost | Interrelazionale | Quando viene inserito un post viene generata una nuova istanza di **notifica** per gli amministratori del gruppo e agli iscritti del gruppo. |
| generaNotificaInterazione | Interrelazionale | Quando viene inserita un’**interazione** viene generata una nuova istanza di **notifica** per l’autore del post |

# 4. Modello Logico

Utente (username,email,password)

*Chiavi esterne*:

Gruppo (idGruppo ,nome,tag,dataCreazione,FK usernameCreatore)

*Chiavi esterne*: FK usernameCreatore ↪ Utente.username

*Chiavi esterne*: FK\_ Username ↪Utente.username FK idGruppo ↪ Gruppo.idGruppo

Richiesta (idRichiesta, dataRichiesta, oraRichiesta, testo, esito, FK username,FK idGruppo)

Post (idPost, foto, testo, dataPubblicazione, oraPubblicazione, FK idGruppo, FK username)

*Chiavi esterne*: FK idGruppo ↪ Gruppo.idGruppo FK username ↪ Utente.username

*Chiavi esterne*: FK username ↪ Utente.username FK idPost ↪ Post.idPost

FK idInterazione↪Interazione.idIterazione

Notifica(idNotifica,data,ora, FK username, FK idPost, FK idInterazione )

*Chiavi esterne*: FK Username ↪ Utente.username FK idPost ↪ Post.idPost

Interazione(idInterazione, data, ora, testo, tipoInterazione, FK username, FK idPost)

*Chiavi esterne*: FK username ↪ Utente.username FK idGruppo ↪ Gruppo.idGruppo

Amministra(FK username, FK idGruppo)

Iscrizione(FK username, FK IdGruppo)

*Chiavi esterne*: FK username ↪ Utente.username FK idGruppo ↪ Gruppo.idGruppo

# 5. Schema Fisico

## 5.1 Definizione tabelle in SQL

### 5.1.1 Definizione tabella Utente

CREATE TABLE Utente (

username VARCHAR(255),

email VARCHAR(255),

password VARCHAR(255)

);

ALTER TABLE utente ADD CONSTRAINT "pkUtente" PRIMARY KEY (username);

ALTER TABLE utente ADD CONSTRAINT uniqueEmail UNIQUE(email);

### 5.1.2 Definizione tabella Gruppo

CREATE TABLE Gruppo (

"idGruppo" VARCHAR(255),

nome VARCHAR(255),

tag VARCHAR(255),

"dataCreazione" DATE,

"usernameCreatore" VARCHAR(255)

);

ALTER TABLE gruppo ADD CONSTRAINT "pkGruppo" PRIMARY KEY ("idGruppo");

ALTER TABLE gruppo ADD CONSTRAINT "fkUtenteCreatore" FOREIGN KEY ("usernameCreatore") REFERENCES utente(username) ON DELETE CASCADE;

ALTER TABLE Gruppo ADD CONSTRAINT uniqueNomeGruppo UNIQUE(nome);

### 5.1.3 Definizione tabella Richiesta

CREATE TABLE Richiesta (

"idRichiesta" VARCHAR(255),

"dataRichiesta" DATE,

"oraRichiesta" TIME,

testo VARCHAR(255),

esito BOOLEAN DEFAULT false,

username VARCHAR(255),

"idGruppo" VARCHAR(255)

);

ALTER TABLE richiesta ADD CONSTRAINT "pkRichiesta" PRIMARY KEY ("idRichiesta");

ALTER TABLE richiesta ADD CONSTRAINT "fkUsername" FOREIGN KEY ("username") REFERENCES utente(username) ON DELETE CASCADE;

ALTER TABLE richiesta ADD CONSTRAINT "fkIdGruppo" FOREIGN KEY ("idGruppo") REFERENCES gruppo("idGruppo") ON DELETE CASCADE;

### 5.1.4 Definizione tabella Post

CREATE TABLE Post (

"idPost" VARCHAR(255),

foto VARCHAR(255),

testo VARCHAR(255),

"dataPubblicazione" DATE,

"oraPubblicazione" TIME,

"idGruppo" VARCHAR(255),

username VARCHAR(255)

);

ALTER TABLE post ADD CONSTRAINT "pkPost" PRIMARY KEY ("idPost");

ALTER TABLE post ADD CONSTRAINT "fkUsername" FOREIGN KEY ("username") REFERENCES utente(username) ON DELETE CASCADE;

ALTER TABLE post ADD CONSTRAINT "fkIdGruppo" FOREIGN KEY ("idGruppo") REFERENCES gruppo("idGruppo") ON DELETE CASCADE;

ALTER TABLE post ADD CONSTRAINT "postNonVuoto"

CHECK ((foto IS NOT NULL OR testo IS NOT NULL) AND NOT (foto IS NULL AND testo IS NULL));

### 5.1.5 Definizione tabella Interazione

CREATE TABLE Interazione (

"idInterazione" VARCHAR(255),

data DATE,

ora TIME,

testo VARCHAR(255),

"tipoInterazione" VARCHAR(255),

username VARCHAR(255),

"idPost" VARCHAR(255)

);

ALTER TABLE interazione ADD CONSTRAINT "pkInterazione" PRIMARY KEY ("idInterazione");

ALTER TABLE interazione ADD CONSTRAINT "fkUsername" FOREIGN KEY ("username") REFERENCES utente(username) ON DELETE CASCADE;

ALTER TABLE interazione ADD CONSTRAINT "fkIdPost" FOREIGN KEY ("idPost") REFERENCES post("idPost") ON DELETE CASCADE ;

ALTER TABLE interazione ADD CONSTRAINT dominioTipoInterazione CHECK ("tipoInterazione" IN ('like','commento'));

### 5.1.6 Definizione tabella Notifica

CREATE TABLE Notifica (

"idNotifica" VARCHAR(255),

data DATE,

ora TIME,

username VARCHAR(255),

"idPost" VARCHAR(255),

"idInterazione" VARCHAR(255)

);

ALTER TABLE notifica ADD CONSTRAINT "pkNotifica" PRIMARY KEY ("idNotifica");

ALTER TABLE notifica ADD CONSTRAINT "fkUsername" FOREIGN KEY ("username") REFERENCES utente(username) ON DELETE CASCADE;

ALTER TABLE notifica ADD CONSTRAINT "fkIdPost" FOREIGN KEY ("idPost") REFERENCES post("idPost") ON DELETE CASCADE;

ALTER TABLE notifica ADD CONSTRAINT "fkIdInterazione" FOREIGN KEY ("idInterazione") REFERENCES interazione("idInterazione") ON DELETE CASCADE;

### 5.1.7 Definizione tabella Iscrizione

CREATE TABLE Iscrizione (

username VARCHAR(255),

"idGruppo" VARCHAR(255)

);

ALTER TABLE iscrizione ADD CONSTRAINT "fkUsername" FOREIGN KEY(username) REFERENCES utente(username) ON DELETE CASCADE;

ALTER TABLE iscrizione ADD CONSTRAINT "fkIdGruppo" FOREIGN KEY("idGruppo") REFERENCES gruppo("idGruppo") ON DELETE CASCADE;

### 5.1.8 Definizione tabella Amministra

CREATE TABLE Amministra (

username VARCHAR(255),

"idGruppo" VARCHAR(255)

);

ALTER TABLE amministra ADD CONSTRAINT "fkUsername" FOREIGN KEY(username) REFERENCES utente(username) ON DELETE CASCADE;

ALTER TABLE amministra ADD CONSTRAINT "fkIdGruppo" FOREIGN KEY("idGruppo") REFERENCES gruppo("idGruppo") ON DELETE CASCADE;

## 5.2 Trigger e Procedure SQL

### 5.2.1 Definizione vincolo “noTestoInLike”

CREATE OR REPLACE FUNCTION no\_testo\_function()

RETURNS trigger AS

$$

BEGIN

IF (NEW.”tipoInterazione” LIKE 'like') THEN

UPDATE interazione SET testo=NULL

WHERE "idInterazione"=NEW."idInterazione";

END IF;

RETURN NULL;

END;

$$

LANGUAGE 'plpgsql';

CREATE TRIGGER "noTestoInLike"

AFTER INSERT on interazione

FOR EACH ROW

EXECUTE FUNCTION no\_testo\_function();

### 5.2.2 Definizione vincolo “coerenzaDataPost”

CREATE OR REPLACE FUNCTION coerenza\_data\_post\_function()

RETURNS trigger AS

$$

DECLARE

dataCreazioneGruppo DATE;

BEGIN

SELECT “dataCreazione”

INTO dataCreazioneGruppo

FROM gruppo

WHERE idGruppo=NEW.idGruppo;

IF(NEW.dataPubblicazione<dataCreazioneGruppo) THEN

UPDATE post SET dataPubblicazione=CURRENT\_DATE, oraPubblicazione=CURRENT\_TIME

WHERE idPost=NEW.idPost;

END IF;

RETURN NULL;

END;

$$

LANGUAGE 'plpgsql';

CREATE TRIGGER "coerenzaDataPost"

AFTER INSERT on post

FOR EACH ROW

EXECUTE FUNCTION coerenza\_data\_post\_function();

### 5.2.3 Definizione vincolo “coerenzaDataInterazione”

CREATE OR REPLACE FUNCTION coerenza\_data\_interazione\_function()

RETURNS trigger AS

$$

DECLARE

dataPubblicazionePost DATE;

oraPubblicazionePost TIME;

BEGIN

SELECT "dataPubblicazione","oraPubblicazione"

INTO dataPubblicazionePost,oraPubblicazionePost

FROM post

WHERE "idPost"=NEW."idPost";

IF((NEW.data<dataPubblicazionePost) OR (NEW.data=dataPubblicazionePost AND NEW.ora<oraPubblicazionePost) ) THEN

UPDATE interazione SET data=CURRENT\_DATE, ora=CURRENT\_TIME

WHERE "idInterazione"=NEW."idInterazione";

END IF;

RETURN NULL;

END;

$$

LANGUAGE 'plpgsql';

CREATE TRIGGER "coerenzaDataInterazione"

AFTER INSERT ON interazione

FOR EACH ROW

EXECUTE FUNCTION coerenza\_data\_interazione\_function();

### 5.2.4 Definizione Vincolo “coerenzaDataNotificaPost

CREATE OR REPLACE FUNCTION coerenza\_data\_notificapost\_function()

RETURNS trigger AS

$$

DECLARE

dataPubblicazionePost DATE;

oraPubblicazionePost TIME;

BEGIN

SELECT "dataPubblicazione","oraPubblicazione"

INTO dataPubblicazionePost,oraPubblicazionePost

FROM post

WHERE "idPost"=NEW."idPost";

IF (NEW."idInterazione"=NULL) THEN

IF((NEW.data<dataPubblicazionePost) OR (NEW.data=dataPubblicazionePost AND NEW.ora<oraPubblicazionePost)) THEN

UPDATE notifica SET data=CURRENT\_DATE, ora=CURRENT\_TIME

WHERE "idNotifica"=NEW."idNotifica";

END IF;

END IF;

RETURN NULL;

END;

$$

LANGUAGE 'plpgsql';

CREATE TRIGGER "coerenzaDataNotificaPost"

AFTER INSERT ON notifica

FOR EACH ROW

EXECUTE FUNCTION coerenza\_data\_notificapost\_function();

### 5.2.5 Definizione Vincolo “coerenzaDataNotificaInterazione”

CREATE OR REPLACE FUNCTION coerenza\_data\_notificainterazione\_function()

RETURNS trigger AS

$$

DECLARE

dataInterazione DATE;

oraInterazione TIME;

BEGIN

SELECT data,ora

INTO dataInterazione,oraInterazione

FROM interazione

WHERE "idInterazione"=NEW."idInterazione";

IF (NEW."idInterazione"<>NULL) THEN

IF((NEW.data<dataInterazione) OR (NEW.data=dataInterazione AND NEW.ora<oraInterazione) ) THEN

UPDATE notifica SET data=CURRENT\_DATE, ora=CURRENT\_TIME

WHERE "idNotifica"=NEW."idNotifica";

END IF;

END IF;

RETURN NULL;

END;

$$

LANGUAGE 'plpgsql';

CREATE TRIGGER "coerenzaDataNotificaInterazione"

AFTER INSERT ON notifica

FOR EACH ROW

EXECUTE FUNCTION coerenza\_data\_notificainterazione\_function();

### 5.2.6 Definizione Vincolo “coerenzaDataRichiesta”

CREATE OR REPLACE FUNCTION coerenza\_data\_richiesta\_function()

RETURNS trigger AS

$$

DECLARE

dataCreazioneGruppo DATE;

BEGIN

SELECT "dataCreazione"

INTO dataCreazioneGruppo

FROM gruppo

WHERE "idGruppo"=NEW."idGruppo";

IF(NEW."dataRichiesta"<dataCreazioneGruppo) THEN

UPDATE richiesta SET "dataRichiesta"=CURRENT\_DATE, "oraRichiesta"=CURRENT\_TIME

WHERE "idRichiesta"=NEW."idRichiesta";

END IF;

RETURN NULL;

END;

$$

LANGUAGE 'plpgsql';

CREATE TRIGGER "coerenzaDataRichiesta"

AFTER INSERT on richiesta

FOR EACH ROW

EXECUTE FUNCTION coerenza\_data\_richiesta\_function();

### 5.2.7 Definizione Vincolo “esitoTrueRichiesta”

CREATE OR REPLACE FUNCTION esito\_true\_richiesta\_function()

RETURNS trigger AS

$$

DECLARE

myIdGruppo gruppo."idGruppo"%TYPE;

myUsername utente."username"%TYPE;

BEGIN

SELECT "idGruppo","username"

INTO myIdGruppo,myUsername

FROM richiesta

WHERE "idRichiesta"=NEW."idRichiesta";

INSERT INTO iscrizione VALUES(myUsername,myIdGruppo);

RETURN NULL;

END;

$$

LANGUAGE 'plpgsql';

CREATE TRIGGER "esitoTrueRichiesta"

AFTER UPDATE OF esito ON richiesta

FOR EACH ROW

WHEN (NEW.esito = true)

EXECUTE FUNCTION esito\_true\_richiesta\_function();

### 5.2.8 Definizione Vincolo “generaNotificaAccesso”

CREATE OR REPLACE FUNCTION genera\_notificaaccesso\_function()

RETURNS trigger AS

$$

DECLARE

myUsernameCreatore utente."username"%TYPE;

myIdNotifica notifica."idNotifica"%TYPE;

BEGIN

SELECT (MAX("idNotifica" :: INTEGER)+1)::VARCHAR(255)

INTO myIdNotifica

FROM notifica;

SELECT "usernameCreatore"

INTO myUsernameCreatore

FROM gruppo

WHERE "idGruppo"=NEW."idGruppo";

INSERT INTO notifica VALUES (myIdNotifica,CURRENT\_DATE,CURRENT\_TIME, myUsernameCreatore, NULL, NULL);

RETURN NULL;

END;

$$

LANGUAGE 'plpgsql';

CREATE TRIGGER "generaNotificaAccesso"

AFTER INSERT on iscrizione

FOR EACH ROW

EXECUTE FUNCTION genera\_notificaaccesso\_function();

### 5.2.9 Definizione Vincolo “generaNotificaPost”

CREATE OR REPLACE FUNCTION genera\_notificapost\_function()

RETURNS trigger AS

$$

DECLARE

cursoreIscritti cursor FOR (SELECT username

FROM iscrizione

WHERE "idGruppo"=NEW."idGruppo" AND "username"<>NEW."username"

);

cursoreAmministratori cursor FOR (SELECT username

FROM amministra

WHERE "idGruppo"=NEW."idGruppo" AND "username"<>NEW."username"

);

myIdNotifica notifica."idNotifica"%TYPE;

BEGIN

FOR iscritto IN cursoreIscritti

LOOP

SELECT (MAX("idNotifica" :: INTEGER)+1)::VARCHAR(255)

INTO myIdNotifica

FROM notifica;

INSERT INTO notifica VALUES (myIdNotifica,CURRENT\_DATE,CURRENT\_TIME,iscritto.username,NEW."idPost",NULL);

END LOOP;

FOR amministratore IN cursoreAmministratori

LOOP

SELECT (MAX("idNotifica" :: INTEGER)+1)::VARCHAR(255)

INTO myIdNotifica

FROM notifica;

INSERT INTO notifica VALUES (myIdNotifica,CURRENT\_DATE,CURRENT\_TIME,amministratore.username,NEW."idPost",NULL);

END LOOP;

RETURN NULL;

END;

$$

LANGUAGE 'plpgsql';

CREATE TRIGGER "generaNotificaPost"

AFTER INSERT on post

FOR EACH ROW

EXECUTE FUNCTION genera\_notificapost\_function();

### 5.2.10 Definizione Vincolo “generaNotificaInterazione”

CREATE OR REPLACE FUNCTION genera\_notificainterazione\_function()

RETURNS trigger AS

$$

DECLARE

myUsernameAutore utente."username"%TYPE;

myIdNotifica notifica."idNotifica"%TYPE;

BEGIN

SELECT (MAX("idNotifica" :: INTEGER)+1)::VARCHAR(255)

INTO myIdNotifica

FROM notifica;

SELECT "username"

INTO myUsernameAutore

FROM post

WHERE "idPost"=NEW."idPost";

INSERT INTO notifica VALUES (myIdNotifica,CURRENT\_DATE,CURRENT\_TIME,myUsernameAutore,NULL,NEW."idInterazione");

RETURN NULL;

END;

$$

LANGUAGE 'plpgsql';

CREATE TRIGGER "generaNotificaInterazione"

AFTER INSERT on interazione

FOR EACH ROW

EXECUTE FUNCTION genera\_notificainterazione\_function();