UNIVERSITÀ DEGLI STUDI DI FERRARA
INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE
BASI DI DATI

Realizzazione Database per Social Network

Azzolini Damiano - Bertagnon Alessandro

UNIVERSITÀ
DEGLI STUDI
DI FERRARA
- EX LABORE FRUCTUS -

INDICE

ELENCO DELLE FIGURE

CAPITOLO 1

MINIMONDO

1.1 DESCRIZIONE

Il progetto di basa sulla realizzazione di una applicazione web per la gestione di una clinica privata. Alla piattaforma possono accedere 5 tipi di utente:

- paziente
- medico
- infermiere
- impiegato
- amministratore

La clinica in questione eroga diversi tipi di **prestazioni** ai suoi utenti, ad esempio: visite specialistiche, esami diagnostici, day surgery e terapie. Ogni prestazione può essere effettuata da uno o più membri dello **staff** (a seconda della complessità) in una delle **sale** della clinica. Al termine di ogni prestazione il medico compila un **referto** corrispondete alla prestazione appena effettuata. Il sistema deve anche gestire i **farmaci** assunti dagli utenti e utilizzati durante le prestazioni. Per motivi di organizzazione interna ogni membro del personale e ogni sala afferisce a uno specifico **reparto** della clinica. Più in dettaglio:

L'utente **PAZIENTE** potrà:

- Registrarsi sulla piattaforma e fare il login. Modificare il proprio profilo.
- Aggiungere/rimuovere i farmaci che assume regolarmente.
- Visionare le prestazioni effettuate con i referti corrispondenti.

L'utente **MEDICO** potrà:

- Fare il login sulla piattaforma e visionare il proprio profilo.
- Visionare le schede personali dei pazienti (compresi i farmaci assunti).
- Visionare le prestazioni e i relativi referti.
- Aggiungere/Modificare/Cancellare i referti delle prestazioni a cui ha preso parte.
- Aggiungere/Rimuovere i farmaci utilizzati nelle prestazioni a cui ha preso parte.
- Aggiungere personale alle prestazioni che gli sono state assegnate.

L'utente **INFERMIERE** potrà:

- Fare il login sulla piattaforma e visionare il proprio profilo.
- Visionare le schede personali dei pazienti (compresi i farmaci assunti).
- Visionare le prestazioni assegnate.
- Aggiungere/Rimuovere i farmaci utilizzati nelle prestazioni alle quali ha preso parte.

L'utente **IMPIEGATO** potrà:

- Fare il login sulla piattaforma e visionare il proprio profilo.
- Prenotare le prestazioni per i pazienti associando ad esse i medici che dovranno effettuarle.
- Visualizzare lo storico delle prestazioni effettuate dai pazienti (ma non i referti).
- Gestire il personale:
  - Modificare lo stipendio dei vari membri dello staff.
  - Modificare il reparto di appartenenza.
- Aggiungere/Modificare/Cancellare le tipologie di prestazioni.
- Aggiungere/Modificare/Cancellare i farmaci nella lista della farmacia.

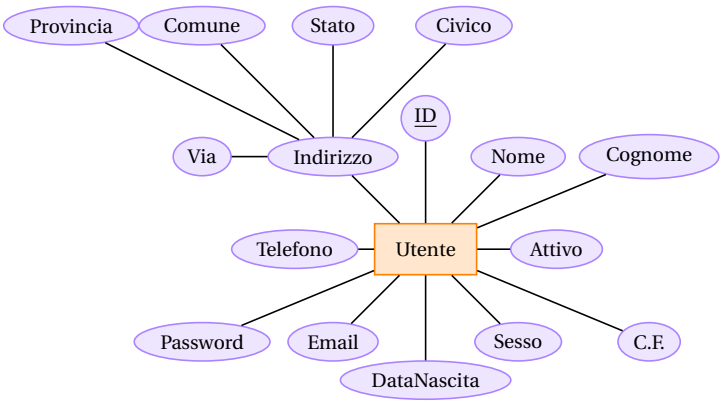
L'utente **AMMINISTRATORE** potrà:

- Fare il login sulla piattaforma e visionare il proprio profilo.
- Fare tutto quello che fanno gli utenti precedenti.
- Aggiungere/Modificare/Cancellare gli utenti Staff della clinica.
- Aggiungere/Modificare/Cancellare le sale della clinica.
- Aggiungere/Modificare/Cancellare i reparti della clinica.
- Gestire tutta la base utenti.
- Aggiungere/Rimuovere i singoli ruoli (permessi) agli utenti.

1.2 Entità

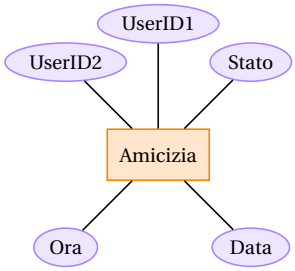
Di seguito vengono analizzate tutte le entità presenti nel database:

1.2.1 Utente



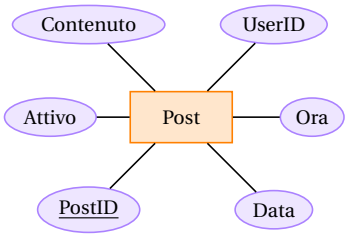
- *Nome*: nome dell'utente.
- *Cognome*: cognome dell'utente.
- *UserID*: identificativo numerico dell'utente e chiave primaria.
- *Password*: password dell'account.
- *Email*: email dell'utente. Nel databse non possono esistere due utenti diversi con la stessa mail.
- *DataNascita*: data di nascita dell'utente.
- *Attivo*: flag booleano, solitamente fissato a 0, 1 se l'utente si è cancellato dal social network.

1.2.2 AMICIZIA



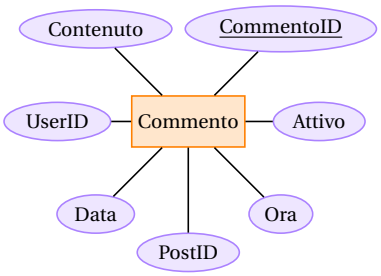
- *UserID1*: rappresenta l'identificativo dell'utente che richiede l'amicizia, chiave esterna.
- *UserID2*: rappresenta l'identificativo dell'utente che riceve la richiesta di amicizia, chiave esterna.
- *timestamp*: timestamp della richiesta dell'amicizia.
- *Stato*: variabile intera che rappresenta se l'amicizia è stata richiesta, accettata, in sospeso o rifiutata.

1.2.3 POST



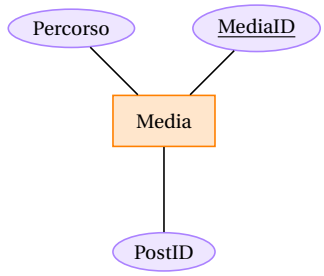
- *PostID1*: rappresenta l'identificativo del post, funge da chiave primaria.
- *Contenuto*: campo di testo che rappresenta il contenuto del post.
- *Data*: data di pubblicazione del post.
- *Ora*: ora di pubblicazione del post.
- *UserID*: chiave esterna, identificativo dell'utente autore del post.
- *Attivo*: flag booleano che indica se il post è attivo (1) o cancellato (0).

1.2.4 COMMENTO



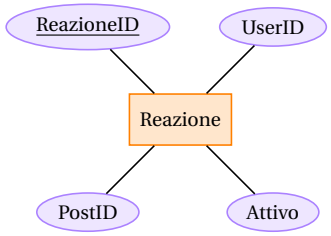
- *CommentoID*: identificativo numerico del commento e chiave primaria.
- *Contenuto*: contenuto del commento.
- *UserID*: identificativo dell'utente autore del commento, chiave esterna.
- *Data*: data del commento.
- *Ora*: ora del commento.
- *PostID*: identificativo numerico del post commentato, chiave esterna.
- *Attivo*: flag booleano che indica se il commento è attivo (1) o cancellato (0).

1.2.5 MEDIA



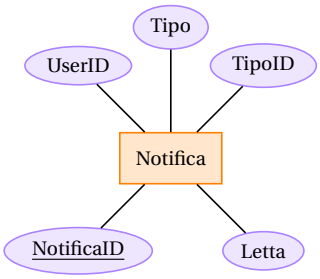
- *MediaID*: identificativo del media (foto, video, ...), chiave primaria.
- *Percorso*: link al media.
- *PostID*: identificativo del post nel quale il media viene inserito.

1.2.6 REAZIONE



- *ReazioneID*: identificativo numerico della reazione.
- *UserID*: identificativo dell'utente che mette la reazione.
- *PostID*: identificativo del post sul quale viene messa una reazione.
- *Attivo*: flag booleano che indica se ila reazione è attiva (1) o è stata cancellata (0).

1.2.7 NOTIFICA



- *NotiificaID*: identificativo numerico della notifica.
- *Tipo*:
- *TipoID*:
- *UserID*:
- *Letta*:

1.2.8 GRUPPO

Entità da aggiungere

1.3 RELAZIONI

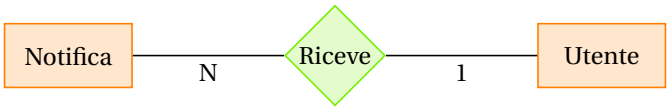


Figura 1.1: Un utente riceve N notifiche ma una determinata notifica è ricevuta da un solo utente.

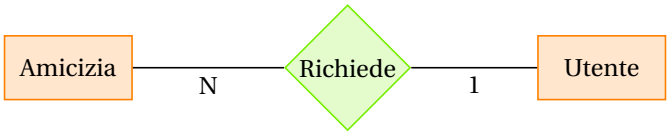


Figura 1.2: Un utente riceve N richieste di amicizia ma una determinata richiesta è ricevuta da un solo utente.

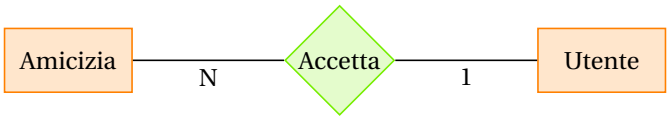


Figura 1.3: Un utente accetta N richieste di amicizia ma una determinata richiesta è accettata da un solo utente.

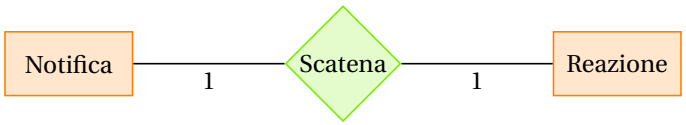


Figura 1.4: Una reazione scatena una notifica e una determinata notifica è scatenata da una reazione.

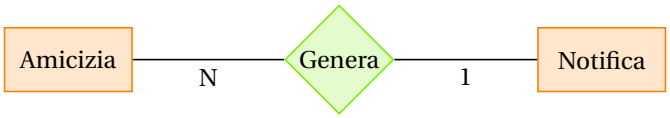


Figura 1.5: Una richiesta di amicizia genera una notifica e una determinata notifica e generata da una richiesta.

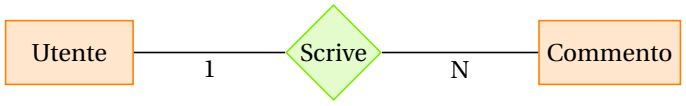


Figura 1.6: Un utente scrive n commenti ma un determinato commento può essere scritto solamente da un utente.

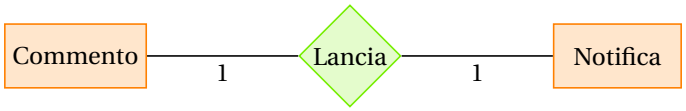


Figura 1.7: Un commento lancia una notifica e una determinata notifica può essere lanciata da un solo commento.

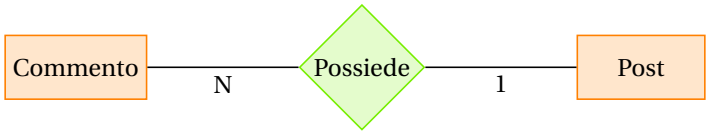


Figura 1.8: Un determinato commento può essere fatto solamente su un post ma un post può contenere n commenti.

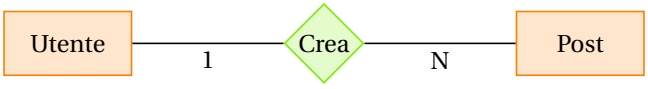


Figura 1.9: Un utente crea n post ma un determinato post è scritto solamente da un utente.

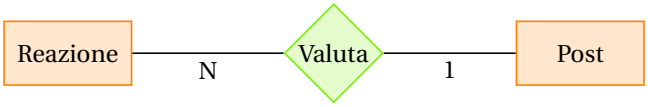


Figura 1.10: Una reazione valuta un solo post ma un post può essere valutato da più reazioni.

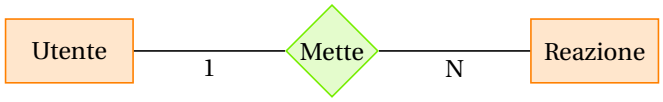


Figura 1.11: Un utente può mettere n reazioni ma una reazione può essere messa da un solo utente.

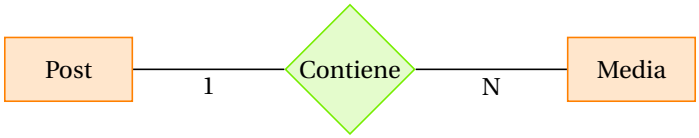


Figura 1.12: Un post contiene n media ma un media può essere in un solo post.

1.4 Schema ER Completo

Da inserire in una pagina nuova

CAPITOLO 2

NORMALIZZAZIONE

CAPITOLO 3

DA MODELLO ER A MODELLO RELAZIONALE

CAPITOLO 4

CODICE SQL

4.1 INTRODUZIONE

Come detto nella descrizione, l'intero progetto è stato sviluppato utilizzando il framework *laravel*. Le tabelle del database sono state create utilizzando le `migration`. Per ogni tabella del database è stato eseguito il comando `php artisan make:migration create_table_nomeTabella`. Questo comando genera una classe `migration` all'interno del file `create_table_nomeTabella.php` nella quale sono definiti i metodi `up()` e `down()`. All'interno di `up()` vengono inseriti tutti i comandi per la creazione delle tabelle. Generate le migrations per tutte le tabelle, il comando `php artisan migrate` traduce i comandi specificati nel metodo `up`, in comandi SQL per la creazione delle tabelle. Di seguito viene riportato il codice SQL generato dalle migrations.

4.2 CODICE

```
UTENTE
CREATE TABLE utente (
        utenteID INT AUTO_INCREMENT PRIMARY_KEY,
        nome VARCHAR(255) NOT NULL,
        cognome VARCHAR(255) NOT NULL,
        email VARCHAR(255) NOT NULL UNIQUE,
        password VARCHAR(255) NOT NULL,
        citta VARCHAR(255) NOT NULL,
        dataNascita DATE
);

NOTIFICA
CREATE TABLE notifica (
        notificaID INT AUTO_INCREMENT PRIMARY_KEY,
        utenteID VARCHAR(255) FOREGIN_KEY REFERENCES utente(utenteID),
        tipo VARCHAR(255),
        tipoID VARCHAR(255),
        letta BIT DEFAULT 0
);

AMICIZIA
CREATE TABLE amicizia (
        utenteID1 INT FOREGIN KEY REFERENCES utente(utenteID),
        utenteID2 INT FOREGIN_KEY REFERENCES utente(utenteID),
        timestamp TIMESTAMP,
```

```
        Stato VARCHAR(255) NOT_NULL DEFAULT 'sospesa'
);

COMMENTO
CRATE TABLE commento (
        commentoID INT AUTO_INCREMENT PRIMARY_KEY,
        utenteID INT FOREGIN_KEY REFERENCES utente(utenteID),
        postID INT FOREGIN_KEY REFERENCES post(postID),
        contenuto TEXT NOT_NULL,
        timestamp TIMESTAMP
);

POST
CREATE TABLE post (
        postID INT AUTO_INCREMENT PRIMARY_KEY,
        contenuto TETX NOT_NULL,
        timestamp TIMESTAMP,
        utenteID FOREGIN_KEY REFERENCES utente(utenteID)
);

MEDIA
CREATE TABLE media (
        mediaID INT AUTO_INCREMENT PRIMARY_KEY,
        postID INT FOREGIN_KEY REFERENCES post(PostID),
        percorso TEXT NOT_NULL
);

REAZIONE
CREATE TABLE rezione (
        reazioneID INT AUTO_INCREMENT PRIMARY_KEY,
        utenteID INT FOREGIN_KEY REFERENCES utente(utenteID),
        postID INT FOREGIN_KEY REFERENCES post(postID),
        flag BIT
);
```

Capitolo 5

Query

Capitolo 6

Interfaccia

BIBLIOGRAFIA

[1] Tetra Pak launches FSC cartons in China, *http://beta.nepcon.org/newsroom/tetra-pak-launches-fsc-cartons-china*, 10 Giugno 2010.

[2] Circular Economy, Sustainable Materials Management, and the Importance of KPIs: *https://sustainablepackaging.org/circular-economy-sustainable-materials-management-importance-kpis/*, 17 Maggio 2017.

[3] Marius Leibold, Gilbert J. B. Probst, Michael Gibbert, *Strategic Management in the Knowledge Economy: New Approaches and Business Applications*, John Wiley & Sons, 2007.