

UniConnect

[Descrizione](#)

[Struttura del Database](#)

[Relationship - DBMS](#)

[CREATED_BY](#)

[FOLLOWS](#)

[Back-End Mapping](#)

[Post - Entity \(ORM\)](#)

[Student - Entity \(ORM\)](#)

[Neo4J Browser - DBMS](#)

[Student - Graph Table](#)

[Student - Record](#)

[Post - Graph Table](#)

[Post - Record](#)

[Features \(Primarie \)](#)

[Update Biography](#)

[Update Post](#)

[Tecnologie Utilizzate](#)

[Sviluppi Futuri](#)

[- Autore](#)

Descrizione

UniConnect è un'applicazione mobile e web progettata per gli studenti universitari per facilitare la comunicazione e l'interazione tra di loro.

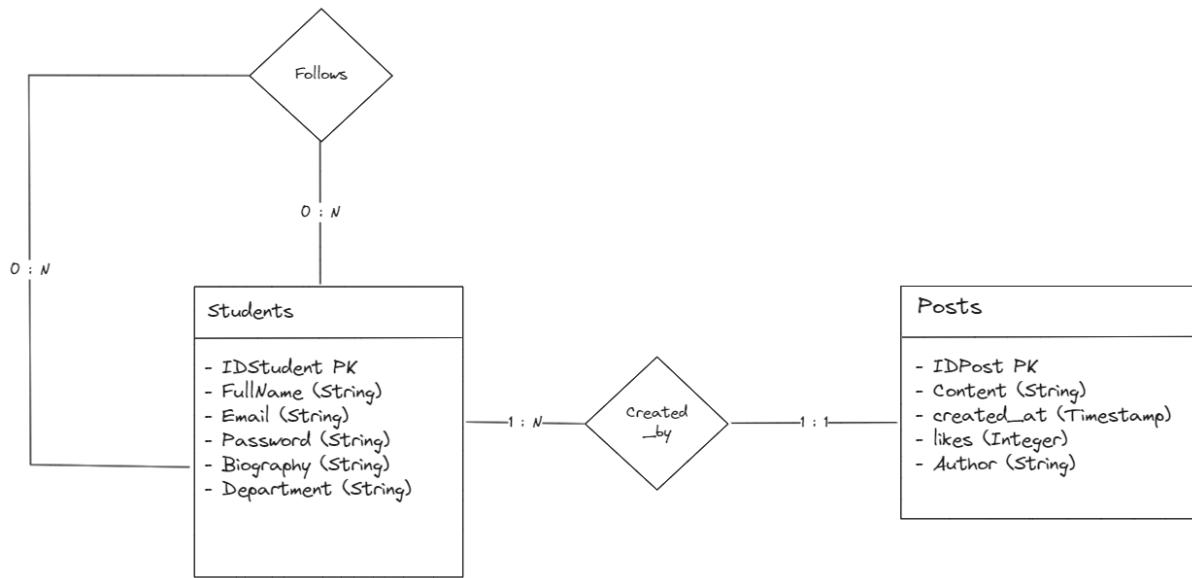
L'applicazione offre funzionalità di social media, permettendo agli studenti di creare profili, connettersi con altri studenti, condividere informazioni e conoscenze.

UniConnect mira a creare un ambiente virtuale in cui gli studenti possano sentirsi coinvolti e integrati nella vita universitari



CD-ROM Struttura del Database CD-ROM

Il database che viene presentato ha la seguente struttura :



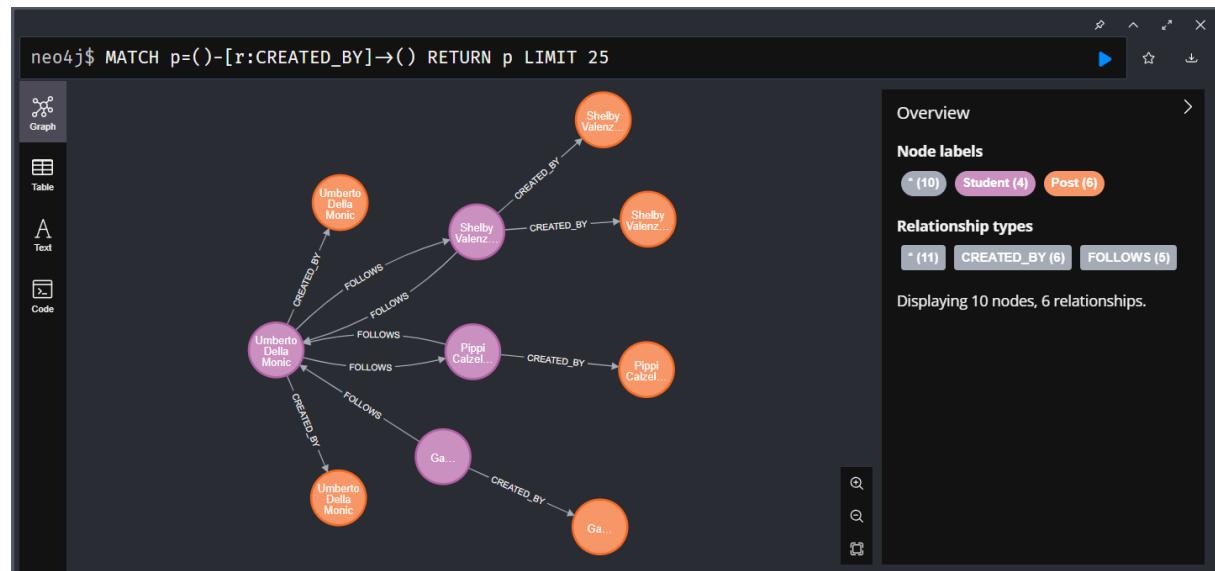
Grazie alla tecnologia di Spring e al mapping ORM possiamo evitare vari ***mismatch impedance*** nella gestione dei dati ad oggetti, infatti i seguenti codici sottostanti rappresentano come queste tabelle del diagramma ER siano state mappate in maniera efficace all'interno della tecnologia Back-End proposta

🔥 Relationship - DBMS

Le varie relazioni che possiamo incontrare sono queste :

CREATED_BY

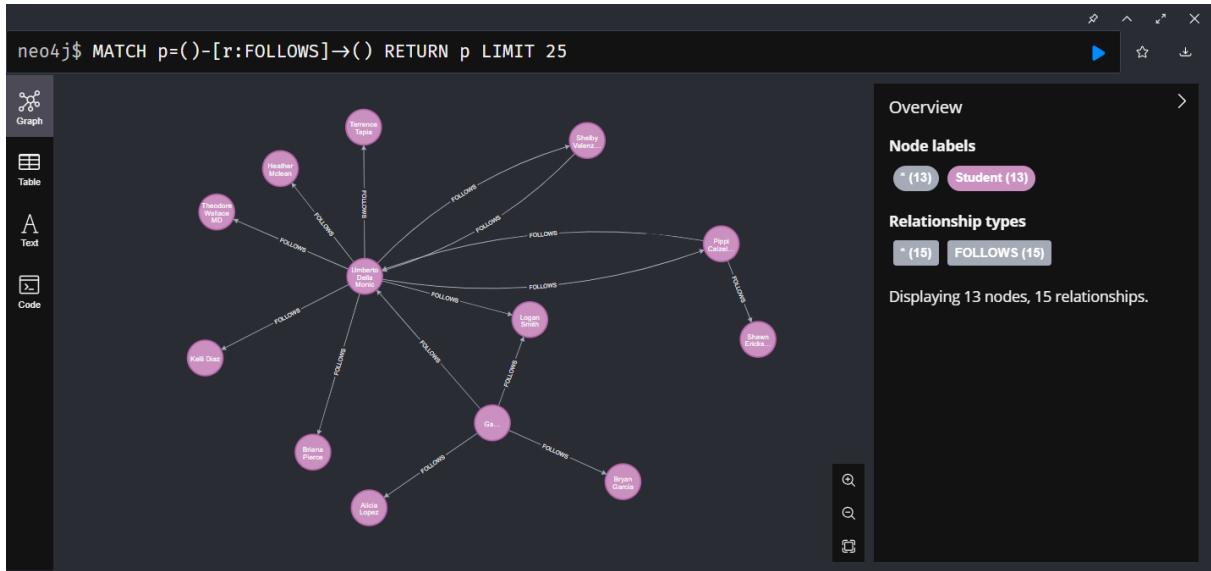
La relazione ***CREATED_BY*** mette in relazione uno studente con la creazione di un post



FOLLOW

La relazione FOLLOW mette in relazione uno studente con un altro studente permettendo di seguire le sue attività sul social network

La relazione può essere Bidirezionale, poiché uno Studente può seguire o non seguire un altro studente



Back-End Mapping

Post - Entity (ORM)

```

import lombok.Getter;
import lombok.Setter;
import org.springframework.data.neo4j.core.schema.*;
import java.util.UUID;

@Node("Post") 12 usages ▲ Umberto Della Monica
@Getter
@Setter
public class PostEntity {

    @Id @GeneratedValue(GeneratedValue.UuidGenerator.class)
    private UUID ID;

    @Property("author")
    private String author;

    @Property("content")
    private String content;
    @Property("created_at")
    private String createdAt;

    private int likes;

    @Relationship(type = "CREATED_BY", direction = Relationship.Direction.INCOMING)
    private StudentEntity student;
}

```

Student - Entity (ORM)

```

@Getters 32 usages ▲ Umberto Della Monica
@Setter
@Node("Student")
public class StudentEntity {

    // ID dello studente
    @Id @GeneratedValue(GeneratedValue.UuidGenerator.class)
    private UUID ID;

    @Property("fullName")
    private String fullName;

    @Property("email")
    private String email;

    private String passwordHash;

    private String biography;

    @Enumerated(EnumType.STRING)
    private DepartementUnisa departementUnisa;

    @Relationship(type = "CREATED_BY", direction = Relationship.Direction.OUTGOING)
    private List<PostEntity> posts;

    @Relationship(type = "FOLLOWS", direction = Relationship.Direction.OUTGOING)
    private List<StudentEntity> following;
}

```

Neo4J Browser - DBMS

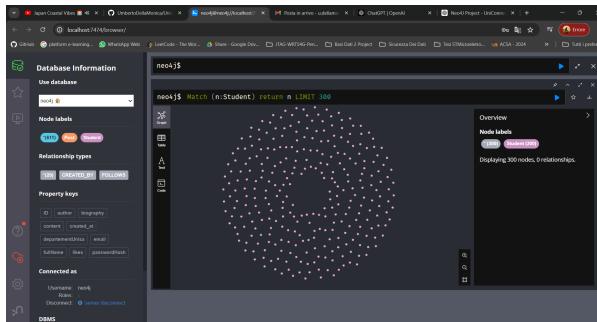
La gestione del database a grafi ci è data dal DBMS di Neo4J che viene fornito:

- Sia come immagine Docker da cui poter creare un container con esecuzione isolata del codice
- Sia come software per la gestione locale del Database

Offrendo anche una User Experience efficiente e intuitiva

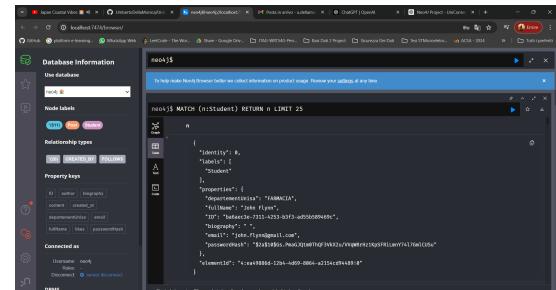
- Vi è anche la possibilità di interagire con il database scrivendo le query nel linguaggio CQL

Student - Graph Table



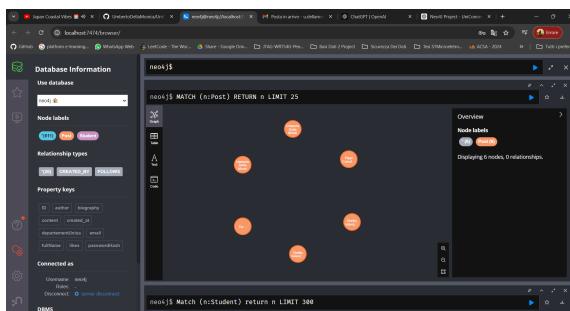
The screenshot shows the Neo4j Browser interface. On the left, there's a sidebar with 'Database Information' and 'Cypher' sections. The main area displays a circular graph of nodes, with a tooltip indicating 'Displaying 300 nodes, 0 relationships'. The Cypher query at the bottom is: 'neok\$ MATCH (n:Student) RETURN n LIMIT 300'.

Student - Record



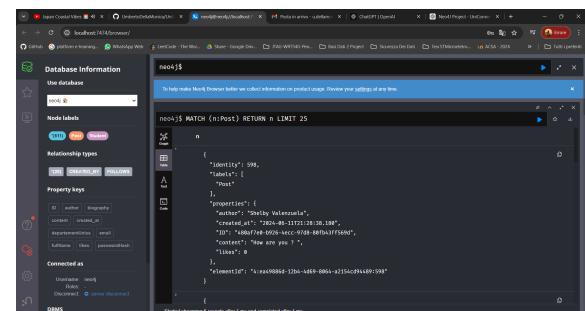
The screenshot shows the Neo4j Browser interface. The main area displays a list of student records. One record is expanded, showing its properties: 'identity': 8, 'label': 'Student', 'properties': { 'name': 'JANNAKA', 'fName': 'John Flynn', 'ID': '2020-08-11123-23130-1079-00700004646', 'major': 'Computer Science', 'email': 'john.flynn@email.com', 'password': 'password123', 'passwordHash': '\$2b\$10\$uXuQZmHgP3hA3zIvqgpe2tqg5hUlam74lhwL0ts' }, 'elementID': '4ca9886d-12ba-4d97-0064-42154c9d4481'. The Cypher query at the bottom is: 'neok\$ MATCH (n:Student) RETURN n LIMIT 25'.

Post - Graph Table



The screenshot shows the Neo4j Browser interface. On the left, there's a sidebar with 'Database Information' and 'Cypher' sections. The main area displays a graph of post nodes, with a tooltip indicating 'Displaying 6 nodes, 0 relationships'. The Cypher query at the bottom is: 'neok\$ MATCH (n:Post) RETURN n LIMIT 25'.

Post - Record



The screenshot shows the Neo4j Browser interface. The main area displays a list of post records. One record is expanded, showing its properties: 'identity': 998, 'label': 'Post', 'properties': { 'content': 'Dobby Valentine', 'createdAt': '2020-08-11123-23130-1079-00700004646', 'ID': '4ca9886d-12ba-4d97-0064-42154c9d4481', 'context': 'How are you?', 'likes': 0, 'tags': [] }, 'elementID': '4ca9886d-12ba-4d97-0064-42154c9d4481'. The Cypher query at the bottom is: 'neok\$ MATCH (n:Post) RETURN n LIMIT 25'.

🔍 Features (Primarie) 🔎

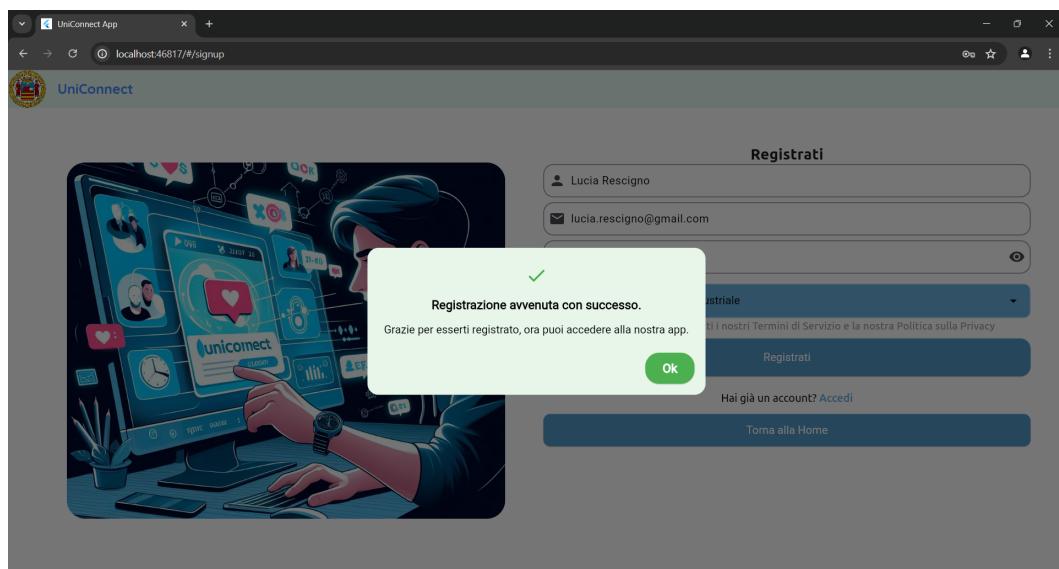
L'applicazione presenta queste funzionalità :

1. **Registrazione** - Ogni studente può essere registrato all'interno della piattaforma e accedere al sito



Il processo di registrazione prevede l'immisione dei dati dell'utente possibilmente :

- Nome e Cognome
- Email - formato @gmail.com
- Password - con vincolo di lunghezza composizione
- Dipartimento di Appartenenza - Lo studente inserisce il suo dipartimento di appartenenza



La Query che viene utilizzata è :

```
/**  
 * Register Student - Registra uno studente all'interno del database  
 * @param ID  
 * @param fullName  
 * @param email  
 * @param passwordHash  
 * @param biography  
 * @param departementUnisa  
 */  
@Query("CREATE (s:Student { " +  no usages  new *  
    "ID: $ID, " +  
    "fullName: $fullName, " +  
    "email: $email, " +  
    "passwordHash: $passwordHash, " +  
    "biography: $biography, " +  
    "departementUnisa: $departementUnisa }) " +  
    "RETURN s")  
StudentEntity createStudent(UUID ID, String fullName, String email,  
                           String passwordHash, String biography, String departementUnisa);
```

2. **Login** - Ogni studente, una volta registrato può loggarsi all'interno di questa applicazione



Il processo di Login viene eseguito secondo questi passi :

- L'utente inserisce le sue credenziali di accesso : (Email e Password)
- Il sistema effettua un controllo su quest'ultime
- Se le credenziali sono corrette allora il sistema manda l'utente alla sua home page

Two side-by-side screenshots of a computer monitor displaying the Uni-Connect login page. Both screenshots show a background illustration of a young man holding a smartphone. The left screenshot shows the initial login form with fields for 'Email' (containing 'ponzelli.pietro@gmail.com') and 'Password', and a dropdown menu set to 'Dipartimento di Farmacia'. Below these are buttons for 'Login', 'Non hai un account? Registrati', and 'Torna alla Home'. The right screenshot shows the same form after a successful login attempt. A modal window is displayed in the center, containing a green checkmark icon, the text 'Autenticazione avvenuta con successo.', and the instruction 'Premi Ok per accedere essere indirizzo all'area utente.' with an 'Ok' button. The rest of the page remains the same as the left screenshot.

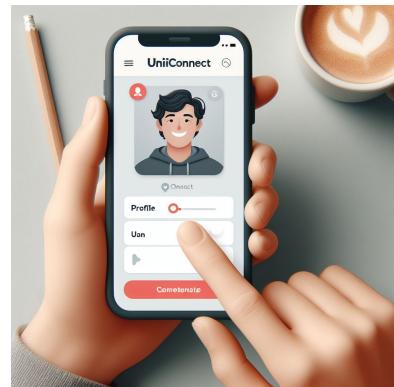
The screenshot shows the UniConnect App interface. On the left, a sidebar for 'Umberto Della Monica' includes a profile picture, name, and links for Profile, Le mie Foto, Followers, and Seguiti. The main area has a search bar 'Search students'. A central post from 'Pippi Calzelunghé' says 'A cosa stai pensando?' with options to share publicly or privately. Below it is a post from 'Shelby Valenzuela' asking 'How are you?'. On the right, a sidebar titled 'Persone connesse' lists 'Pippi Calzelunghé' and 'Shelby Valenzuela'.

La query che viene utilizzata per poter ottenere i dati dello studente viene rappresentata dalla ricerca di quest'ultimo mediante la sua email che deve essere unica all'interno di tutto il sito :

```
/**
 * RetrieveByEmail - Metodo che recupera i Dati dello studente esistente
 * @param email dello studente
 * @return Uno Studente singolo
 */
@Query("MATCH (s:Student) WHERE s.email = $email RETURN s") 2 usages  ↳ Umberto Della Monica
Optional<StudentEntity> retrieveByEmail(String email);
```

3. Aggiornamento / Cancellazione

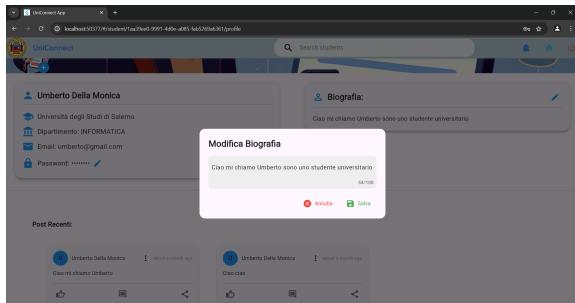
del Profilo - Ogni studente può aggiornare o cancellare il proprio profilo su UniConnect



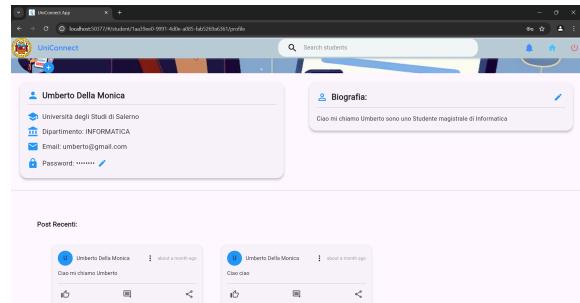
Il processo di aggiornamento può essere fatto per diverse sezioni :

- Biografia
- Post

Update Biography



Aggiornamento Biografia



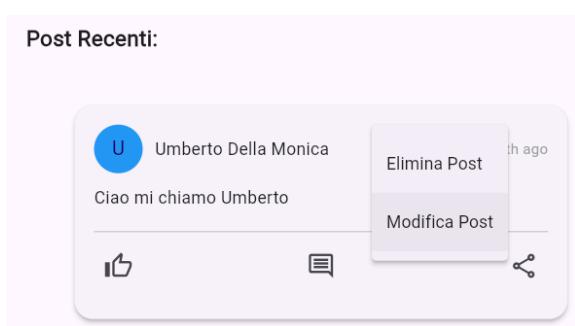
Aggiornamento effettuato della biografia

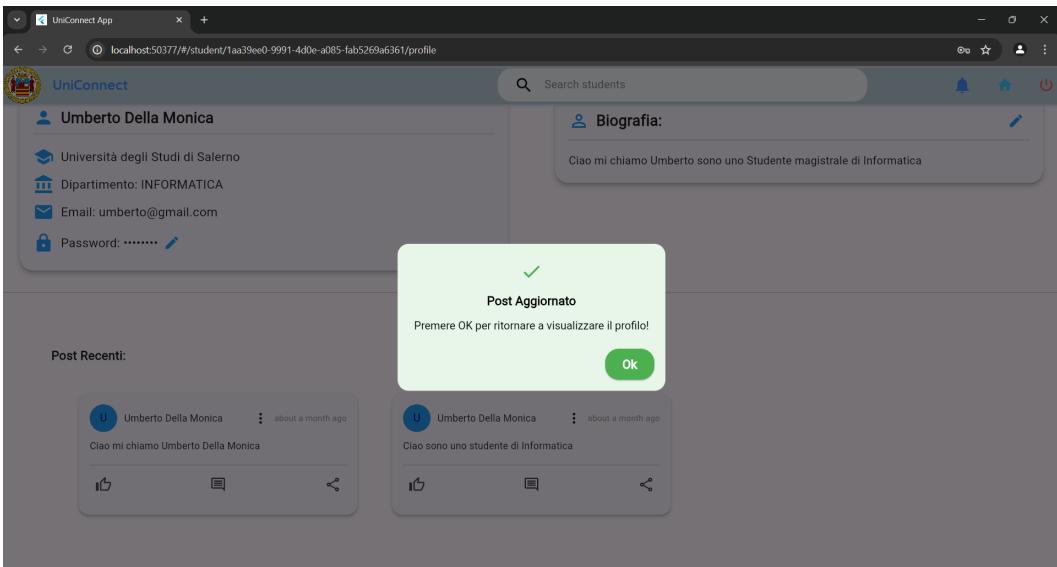
La query che viene adoperata per l'aggiornamento dei dati dello studente è :

```
/*
 * Update Student - Aggiorna i dati dello studente una volta che è stato ricercato tramite il suo ID
 * Aggiorna tutti i suoi dati di seguito :
 * - ID
 * - fullName
 * - email
 * - password
 * - biography
 * - departement
 * @param ID
 * @param fullName
 * @param email
 * @param passwordHash
 * @param biography
 * @param departementUnisa
 */
@Query("MATCH (s:Student {ID: $ID}) " + no usages new *
        "SET s.fullName = $fullName, " +
        "s.email = $email, " +
        "s.passwordHash = $passwordHash, " +
        "s.biography = $biography, " +
        "s.departementUnisa = $departementUnisa " +
        "RETURN s")
StudentEntity updateStudent(UUID ID, String fullName, String email,
                           String passwordHash, String biography, String departementUnisa);
```

Update Post

Il processo di aggiornamento del post segue questo procedimento :





La query che viene adottata per modificare tutte le informazioni di un post è :

```
@Query("MATCH (p:Post {ID: $ID}) " + no usages new *
      "SET p.content = $content " +
      "RETURN p")
PostEntity updatePostContent(UUID ID, String content);
```

4. **Search User** - Ogni studente può ricercare altre persone che sono all'interno della piattaforma e vedere i loro contenuti



Il processo di ricerca viene fatto appositamente all'interno di una pagina specifica dell'applicazione e lì, l'utente in realtime può digitare i nomi che intende ricercare come in questo caso :

The screenshot shows a web browser window titled "UniConnect App" with the URL "localhost:50377/#/student/1aa39ee0-9991-4d0e-a085-fab5269a6361/search". The main content area displays a list of student profiles. Each profile includes a small circular icon with a crest, the student's name, and their email address.

- Briana Pierce
briana.pierce@gmail.com
- Terrence Tapia
terrence.tapia@gmail.com
- James Pitts
james.pitts@gmail.com
- Christopher Pineda
christopher.pineda@gmail.com
- Pippi Calzelunghe
pippi@gmail.com
- Poncio Pilato
poncio.pilato@gmail.com

Ora per fare in modo che la ricerca avvenga con successo e in realtime, si adopera una visita del database a grafo dove si possono visualizzare i dati delle persone che hanno quelle lettere all'interno del nome

La query viene presentata in questo modo e fa ad effettuare un controllo sull'ID dello studente che richiede la ricerca delle persone

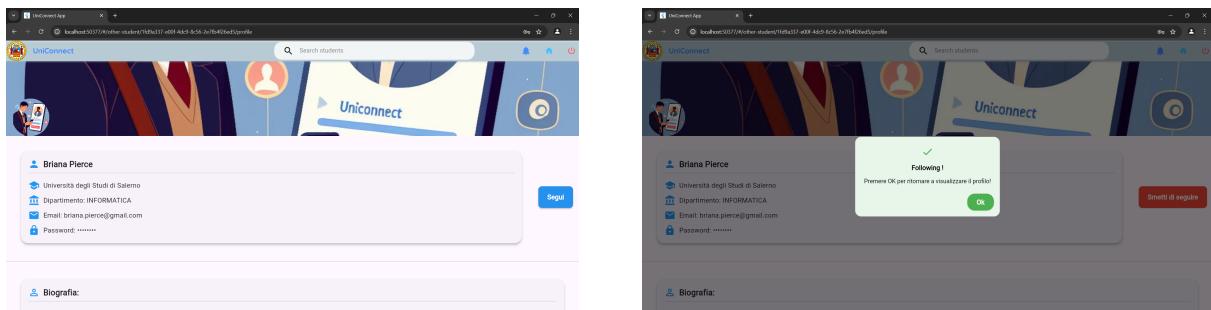
```
/**  
 * Retrieve all User with that name  
 * @param query contains the letter of that name  
 * @param IDStudent IDStudent ID of Student  
 */  
@Query("MATCH (s:Student) WHERE toLower(s.fullName) CONTAINS toLower($query) AND s.ID <> $IDStudent RETURN s") 1 usage  
List<StudentEntity> searchStudents(String query, UUID IDStudent);
```

5. **Follow** - Una volta che l'utente ha ricercato altri studenti può mettersi in contatto con esso tramite l'azione di follow all'interno della pagina social



Una volta che lo studente ha ricercato un suo coetaneo può arrivare a seguire questa determinata persona, premendo il tasto "Follow" sulla pagina in

questione



- Si effettua prima una verifica sul soggetto che si vuole seguire per verificare che non lo abbiamo già tra i seguiti
- Dopodichè si procede con il following della persona tramite la query che viene presentata successivamente :

```
/**  
 * Follow - Perform action where user follow the other user  
 * @param followerId ID of user logged  
 * @param followeeId ID of user that will be followed  
 * @return true if the relationship was created successfully, false otherwise  
 */  
@Query("MATCH (follower:Student {ID: $followerId}), (followee:Student {ID: $followeeId}) " +  
    "MERGE (follower)-[r:FOLLOWS]->(followee) " +  
    "RETURN CASE WHEN r IS NOT NULL THEN true ELSE false END AS result")  
Boolean followUser(UUID followerId, UUID followeeId);
```

6. **Pubblicare Post** - Ogni studente avrà la possibilità di poter pubblicare un post che contenga immagini o del testo



Ogni studente può pubblicare un post sul proprio profilo mediante la sezione apposita

The screenshot shows the UniConnect App interface. On the left, a sidebar for 'Umberto Della Monica' includes a profile picture, a blue circular placeholder, and links for 'Profile', 'Le mie Foto', 'Followers', and 'Seguiti'. The main area displays posts from other users: 'Pippi Calzelunghe' with the message 'Hi, my name is Pippi Calzelunghe' and 'Shelby Valenzuela' with the message 'How are you?'. A 'Personne connesse' sidebar lists 'Pippi Calzelunghe' and 'Shelby Valenzuela' with their respective profile pictures.

This screenshot shows a modal dialog for creating a new post. The input field contains the text 'Ciao, sono uno studente universitario che frequenta l'università di Salerno'. Below the input field are three small icons (green checkmark, red X, yellow smiley face) and a large blue 'Pubblica' button. To the right, a confirmation dialog box is displayed with a green checkmark icon, the text 'Post pubblicato', and a message 'Premere OK per ritornare a pubblicare post!', with an 'Ok' button at the bottom.

La query adibita alla creazione e pubblicazione del post risulta essere :

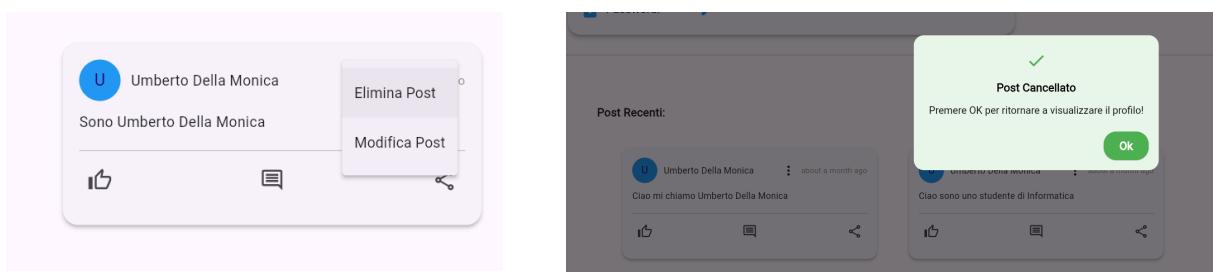
```

@Query("MATCH (student:Student {ID: $studentId}) " + no usages new *
      "CREATE (post:Post {ID: $postId, author: $author, content: $content, created_at: $createdAt, likes: $likes}) " +
      "CREATE (student)-[:CREATED_BY]->(post) " +
      "RETURN post")
PostEntity createPost(UUID studentId, UUID postId, String author, String content, String createdAt, int likes);
    
```

7. **Eliminazione Post** - Ogni studente avrà la possibilità di poter eliminare un post



Come per la cancellazione di un utente, possiamo fare in modo che il post venga cancellato



```
@Query("MATCH (post:Post {ID: $postId}) " +  
        "DETACH DELETE post")  
void deletePost(UUID postId);
```

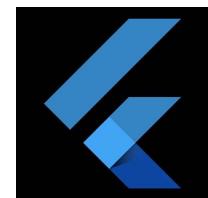
🔧 Tecnologie Utilizzate

L'applicazione di UniConnect viene sviluppata con tecnologie innovative e ricercate nel contesto aziendale.

Tecnologia	Descrizione
Back-End : Spring	Viene utilizzato il framework di Flask, utilizzato per creare microservizi in Java
Front-End : <i>Flutter</i>	Utilizziamo Flutter che ha come linguaggio di programmazione scritto in Dart e permette uno sviluppo multiplataforma - Mobile App - Web App - Desktop Application
Neo4J	Tecnologia innovativa che permette di memorizzare le informazioni mediante una struttura a grafi con linguaggio Query denominato Cipher



Spring Framework



flutter -Logo



Neo4J technology

🚀 Sviluppi Futuri 🚀

Nei sviluppi successivi l'applicazione avrà la possibilità di :

- **Chat** - Ogni studente potrà avviare una chat con altri studenti, potendosi scambiare delle informazioni necessarie sugli eventuali corsi



👤- Autore

Umberto Della Monica - Matricola : 0522501617