PROGETTO LINKEDIN

OBBIETTIVO

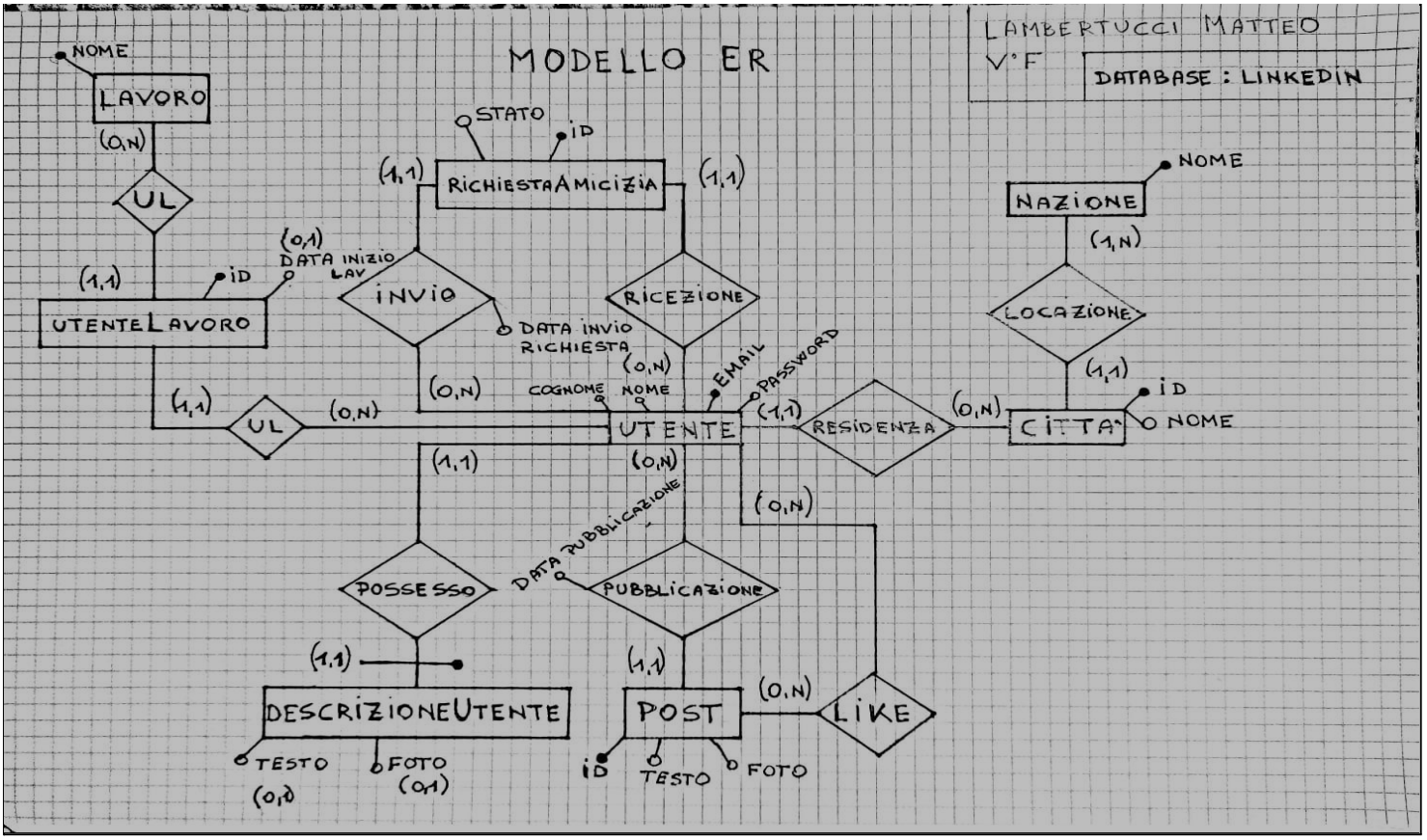
Si richiede la realizzazione di un progetto informatico che abbia come riferimento il portale “Linkedin” (http://www.linkedin.com). L’obiettivo è ricreare le seguenti funzionalità del portale: • Registrazione • Creazione scheda profilo • Richiesta contatto altri utenti • Visualizzazione scheda profilo propria e altri utenti • Visualizzazione scheda profilo altri utenti.

PROGETTAZIONE CONCETTUALE

Modello ER

Database: Linkedin.

Entità principale: Utente.



Dizionario dei Dati

ENTITA’	DESCRIZIONE	ATTRIBUTI	IDENTIFICATIVO
Utente	Utente di Linkedin.	email: stringa, password: stringa, nome: stringa, cognome: stringa	Id: intero > 0
Citta’	Citta di residenza dell’utente.	Id: intero > 0, nome: stringa.	Id: intero > 0
Nazione	Nazionalità dell’Utente.	nome: stringa	Id: intero > 0
Lavoro	Lavoro svolto dall’Utente	nome: stringa	nome: stringa
UtenteLavoro	Collegamento tra Utente e Lavoro.	Id: intero > 0, dataInizioLavoro: data	Id: intero > 0
DescrizioneUtente	Profilo personale dell’Utente.	testo: stringa, foto: stringa.	Utente (E)
Post	Post pubblicato dall’Utente.	Id: intero > 0, testo: stringa, foto:stringa	Id: intero > 0

RichiestaAmicizia	Richiesta di amicizia inviata da un utente ad un altro.	Id:intero > 0,
		stato: ENUM {
		“Sospesa”,
		“Accettata”
		}
			Id: intero > 0

RELAZIONE	DESCRIZIONE	ATTRIBUTI	ENTITÀ' COINVOLTE
Pubblicazione	Pubblicazione del Post.	DataPubblicazione: dataOra	Utente (0, N). Post (1,1).
Residenza	Residenza cittadina dell'utente.		Città (0, N). Utente(1, 1).
Locazione	Locazione nazionale della Citta.	.	Nazione(1, N). Citta(1, 1).
Possesso	Possesso della Descrizione Utente		Utente (1,1). DescrizioneUtente (1, 1).
Invio	Invio di una richiesta di amicizia da un Utente all'altro.	DataInvio :dataOra	Utente (0, N). RichiestaAmicizia (1, 1).
Ricezione	Ricezione di una richiesta di amicizia da parte di un Utente.		RichiestaAmicizia (1, 1). Utente (0, N).
UL	Collegamente tra Utente e Lavoro.	.	Utente (0, N). UtenteLavoro (1, 1).
LU	Collegamente tra Lavoro e Utente.		UtenteLavoro (1, 1). Lavoro (0, N).

ATTRIBUTO	DESCRIZIONE	DOMINIO	COSTRUTTO
Email	Email (univoca) dell'Utente.	Stringa, numero caratteri (variabile): 2 - 35.	Utente (E)
Password	Password dell' Utente.	Stringa, numero caratteri (fisso): 60.	Utente (E)
Nome	Utente: Nome anagrafico dell'Utente. Citta: Nome della Città (non univoco). Nazione: Nome della Nazione. Lavoro: Nome del Lavoro	Stringa,numero caratteri (variabile): 2 - 50.	Utente (E). Citta (E). Nazione (E). Lavoro (E).
Cognome	Cognome anagrafico dell' Utente.	Stringa,numero caratteri (variabile): 2 - 50.	Utente (E)
DataPubblicazione	Data di pubblicazione del Post.	DataOra	Pubblicazione (R)
DataInvioRichiesta	Data di Invio della richiesta di Amicizia.	DataOra	Invio (R)
DataInizioLavoro	Data di inizio Lavoro di un Utente.	Data	UtenteLavoro (E)
Id	Codice Identificativo Numerico.	Intero > 0.	Post (E). Città (E). UtenteLavoro (E). RichiestaAmicizia (E).
Foto	Post: Nome immagine (file salvato nel FileSystem) del Post di un Utente. DescrizioneUtente: Nome immagine (file salvato nel FileSystem) della foto di Profilo di un Utente.	Stringa, numero caratteri (variabile): 4 - 25.	Post (E). DescrizioneUtente(E).

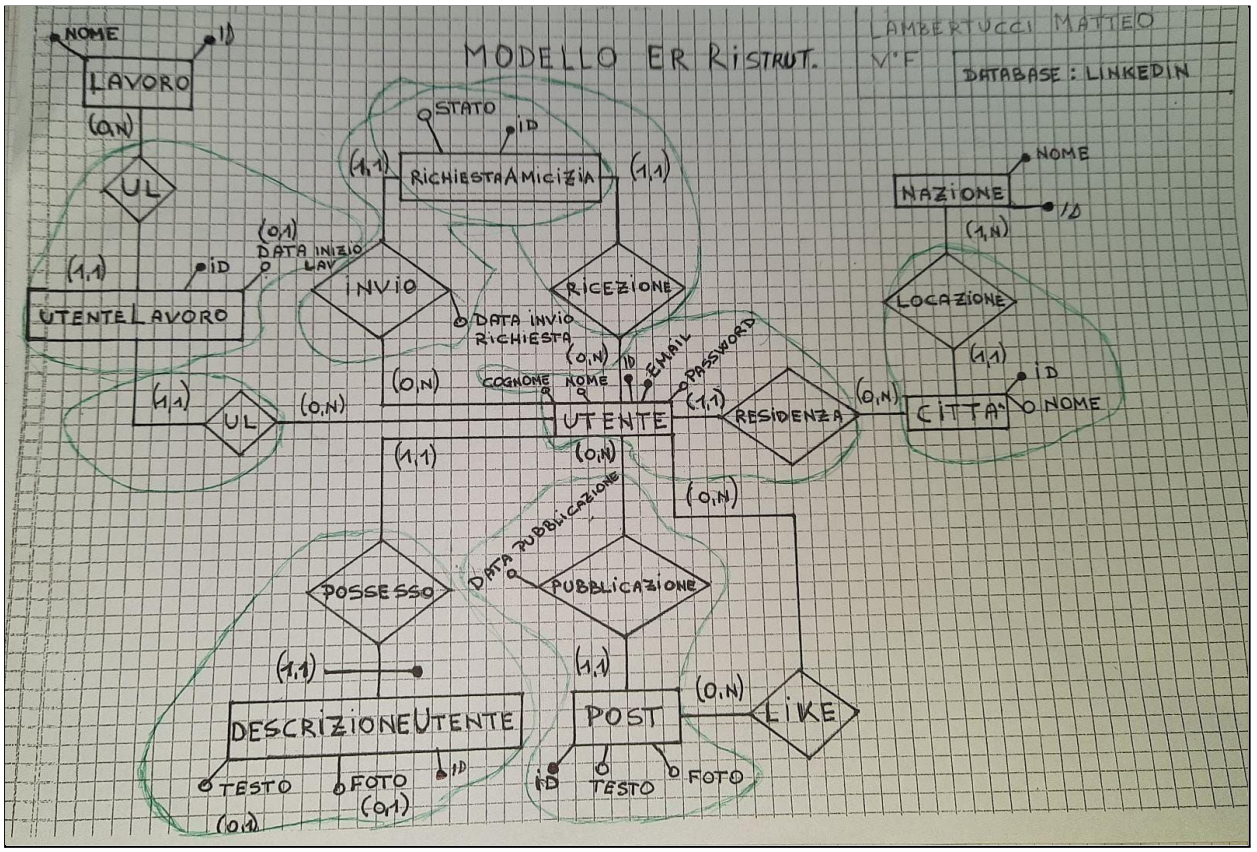
Testo	Testo di un Post o della biografia della Descrizione di un Utente .	Post: Stringa, numero caratteri: (variabile): 2 - 50. DescrizioneUtente: Stringa, Numero caratteri:(variabile): 2-255.	Post (E). DescrizioneUtente(E).

Stato	Stato della Richiesta di Amicizia.	ENUM {
		“Sospesa”,
		“Accettata”
		}
			RichiestaAmicizia (E)

REGOLE DI VINCOLO
1) La data di Inizio del Lavoro deve essere precedente o uguale alla data attuale.
2) La data di Invio della Richiesta di amicizia deve essere precedente o uguale alla data attuale.
3) La data di pubblicazione del Post deve essere precedente o uguale alla data attuale.
4) Se è presente una data di inizio Lavoro, l'Utente deve svolgere almeno un Lavoro.
5) Se non è presente la data di inizio di Lavoro, l'Utente non deve svolgere nessun Lavoro.
6) Il nome del Lavoro deve essere il nome di un Lavoro esistente.
7) Il nome della Nazione deve essere il nome di una Nazione esistente nel Mondo.
8) Il nome della Città deve essere il nome di una Città esistente nella sua Nazione di riferimento.
9) L'Email dell'Utente deve rispettare le norme della forme delle Email, in più deve essere esistente.
10) Il Nome dell'Utente deve rispettare le norme anagrafiche della Nazione di riferimento.
11) Il Cognome dell'Utente deve rispettare le norme anagrafiche della Nazione di riferimento.
12) La foto del Post deve essere il nome di un File presente nel Fyle System.
13) La foto della DescrizioneUtente (profilo personale) deve essere il nome di un File presente nel Fyle System.
14) La data di pubblicazione del Post deve essere antecedente alla data di registrazione dell'Utente.
15) Un Utente non può né' inviare né' ricevere richiesta di amicizia da se stesso.
16) La data di Invio della Richiesta di amicizia deve essere antecedente alla data di registrazione dell'Utente mittente / ricevente.
17) La data di inizio Lavoro deve essere antecedente alla data di nascita dell'Utente.
18) La data di pubblicazione del Post deve essere antecedente alla data di nascita dell'Utente.

PROGETTAZIONE LOGICA

Modello ER ristrutturato



Cambiamenti:

1. **Per quanto riguarda la scelta degli identificatori principali, per le entità Utente, Nazione, Lavoro e DescrizioneUtente, ho scelto i codici (id).**

2. **Non ci sono generalizzazioni.**

3. **Non ci sono attributi multivalore.**

4. **Non ho trovato ridondanze.**

Inglobamenti:

- **Utente (1, 1) [Residenza] Citta (0, N) = Utente ingloba Citta.**
- **Utente (0, N) [Pubblicazione] Post (1, 1) = Post ingloba Utente.**
- **Utente (1, 1) [Possesso] DescrizioneUtente (1, 1) = DescrizioneUtente (ha la chiave esterna) ingloba Utente.**
- **Utente (0, N) [Like] Post (0, N) = Like prende le chiavi di entrambe le Entità coinvolte nella relazione.**
- **Utente (0, N) [Invio] RichiestaAmicizia (1, 1) = RichiestaAmicizia ingloba Utente.**
- **Utente (0, N) [Ricezione] RichiestaAmicizia (1, 1) = RichiestaAmicizia ingloba Utente.**
- **Utente (0, N) [UL] UtenteLavoro (1, 1) = UtenteLavoro ingloba Utente.**
- **Lavoro (0, N) [LU] UtenteLavoro (1, 1) = UtenteLavoro ingloba Lavoro.**
- **Città (1, 1) [Locazione] Nazione (1, N) = Città ingloba Nazione.**

Modello Relazionale

Lavoro (id, nome**);**

Nazione (id, nome**);**

Citta (id, nome, nazione**)**
FOREIGN KEY**(**nazione**)** REFERENCES **Nazione(**id**);**

Utente (id, email, password, nome, cognome, citta**)**
FOREIGN KEY**(**citta**)** REFERENCES **Citta(**id**);**

UtenteLavoro(utente**,** lavoro**,** dataInizioLavoro**)**
FOREIGN KEY**(**utente**)** REFERENCES **Utente(**id**)**
FOREIGN KEY**(**lavoro**)** REFERENCES **Lavoro(**id**);**

Post (id, testo, foto, dataPubblicazione, utente**)**
FOREIGN KEY**(**utente**)** REFERENCES **Utente(**id**);**

DescrizioneUtente (id, testo, foto, utente)
FOREIGN KEY**(**utente**)** REFERENCES **Utente(**id**);**

RichiestaAmicizia (id, utenteMittente, utenteRicevente, stato, dataInvio)
FOREIGN KEY**(**utenteMittente**)** REFERENCES **Utente(**id**)**
FOREIGN KEY**(**utenteRicevente**)** REFERENCES **Utente(**id**);**

MiPiace (id, post, utente)
FOREIGN KEY**(**utente**)** REFERENCES **Utente(**id**)**
FOREIGN KEY**(**post**)** REFERENCES **Post (**id**);**

Vincoli di Integrità

Vincoli di inclusione:

Id Citta Citta (Id);

Id Utente Utente (Id) ;

Id Post Post (Id) ;

Id DescrizioneUtente DescrizioneUtente(Id) ;

Id UtenteLavoro UtenteLavoro (Id) ;

Id RichiestaAmicizia RichiestaAmicizia (Id) ;

Id RichiestaAmicizia RichiestaAmicizia (Id) ;

Id UtenteLavoro UtenteLavoro (Id) ;

Vincoli di Dominio:

Id > 0;

Email: Stringa, numero caratteri (variabile): 2 - 35;

Password: Stringa, numero caratteri (fisso): 60;

Nome: Stringa, numero caratteri (variabile): 2 - 50;

Cognome: Stringa, numero caratteri (variabile): 2 - 50;

Foto: Stringa, numero caratteri (variabile): 4 - 25;

Testo: Stringa, numero caratteri (variabile): 2 - 255;

PROGETTAZIONE FISICA

Mapping tipi di dato: Modello ER - MySQL

ATTRIBUTO	MODELLO ER	MYSQL
Email	Stringa numero caratteri (variabile): 2 - 45.	VARCHAR(50) NOT NULL UNIQUE CHECK(CHAR_LENGTH(email) > 2)
Password	Stringa numero caratteri (fisso): 60.	CHAR(60) NOT NULL CHECK(CHAR_LENGTH(password) > 2);
Nome	Stringa numero caratteri (variabile): 2 - 50.	VARCHAR(50) NOT NULL CHECK(CHAR_LENGTH(nome) > 2);
Cognome	Stringa numero caratteri (variabile): 2 - 50.	VARCHAR(50) NOT NULL CHECK (CHAR_LENGTH(cognome) > 2);
Id	Intero > 0	INT(10) UNSIGNED AUTO_INCREMENT PRIMARY KEY;
Foto	Stringa numero caratteri (variabile): 4 - 25.	Post: VARCHAR(25) NOT NULL CHECK (CHAR_LENGTH(foto) > 2); DescrizioneUtente: VARCHAR(25);
Testo	Stringa numero caratteri (variabile): 2 - 255.	Post: VARCHAR(255) NOT NULL CHECK CHAR_LENGTH(testo) > 2); DescrizioneUtente: VARCHAR(255)
DataInizioLavoro	Data	DATE
DataInvioRichiesta	DataOra	TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP();
DataPubblicazione	DataOra	TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP()
Stato	ENUM { “Sospesa”, “Accettata” }	ENUM (“Sospesa”, “Accettata”) NOT NULL DEFAULT “Sospesa”

RDBMS scelto: MySQL.

Query di esempio: (ricordarsi prima di selezionare il database tramite “USE”)

```sql
-- QUERY DI ESEMPIO --

SET @n = 15;

SELECT
    p.id,
    p.utente AS utente_id,
    p.foto,
    p.testo,
    DATE_FORMAT(p.created_at,'%Y-%m-%d %H:%i') AS dataPubblicazione,
    CONCAT(u.nome, ' ', u.cognome) AS utente,
    CONCAT(l.nome, ' presso ', c.nome, ', ', n.nome, '.') AS lavoroPresso,
    u.email AS utenteEmail,
    COUNT(mp.id) AS miPiace
FROM
    Post p
        LEFT JOIN MiPiace mp ON p.id = mp.post
        JOIN Utente u ON p.utente = u.id
        JOIN UtenteLavoro ul ON ul.utente = u.id
        JOIN Lavoro l ON ul.lavoro = l.id
        JOIN Citta c ON u.citta = c.id
        JOIN Nazione n ON c.nazione = n.id
WHERE
        Nazione <> 'Finlandia' AND
        p.utente < (SELECT @n)
GROUP BY
    p.id
ORDER BY
    p.created_at DESC
LIMIT 25;
```

Esempio creazione Indice fisico

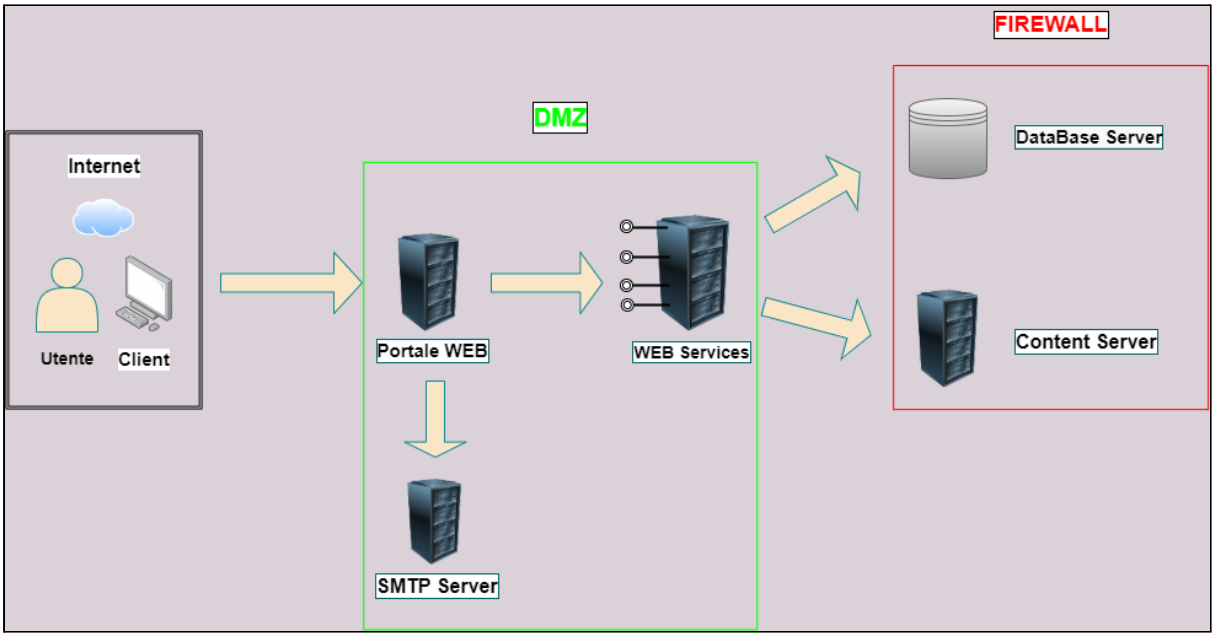
```sql
CREATE INDEX user_account ON Utente(email, password) USING BTREE;
```

[Codice SQL per replicare il DataBase (clicca)]

ARCHITETTURA DEL SISTEMA INFORMATICO

Lista “componenti”

1. DATABASE SERVER.
2. 2 WEB SERVER.
3. CONTENT SERVER.
4. SMTP SERVER.



Spiegazione architettura sistema informatico

- **Il DataBase Server (cuore della mia applicazione) sarà utilizzato per la gestione (storage, interrogazioni, aggiornamenti, eliminazioni...) dei dati relativi agli Utenti del mio sito WEB, dunque salvo le loro credenziali, informazioni anagrafiche, i loro post, like, amicizie, caratteristiche dei loro profili personali ed il ruolo che ricoprono nel mondo del lavoro.**
  Nel mio caso utilizzo un RDBMS, MySQL. Tale "componente" sarà coperto dal Firewall.

- **I WEB Server (portale WEB e WEB Services) saranno utilizzati per gestire le richieste HTTP / HTTPS dei Client (generalmente WEB browser). Le risposte HTTP / HTTPS servite saranno le pagine WEB del mio sito (a seconda della richiesta), dunque utilizzo tale componente anche per l'hosting del mio Sito.**
  Nel mio caso, come WEB Server, utilizzo NGINX.

  N.B. = Entrambi i WEB Server, per essere raggiungibili dall'Utente saranno presenti nella DMZ.

- **Il Content Server sarà utilizzato per lo storage delle immagini di Profilo e dei Post degli Utenti del mio sito WEB, in modo da non salvarle direttamente nel DataBase, evitando il rischio di "sovraccaricarlo". Tale "componente" sarà coperto dal Firewall.**

- **L'SMTP Server sarà utilizzato per inviare Email agli utenti iscritti del mio sito WEB, soprattutto nel caso che uno di essi si dimentichi la password per accedere al sito stesso. Tale "componente" non sarà coperto dal Firewall, infatti sarà presente nella DMZ.**

Funzionamento

L’Utente tramite il WEB browser (HTTP / HTTPS Client) della sua macchina effettua una richiesta HTTP / HTTPS al Portale WEB, il quale, subito dopo comunica con il successivo WEB Server che, a seconda della richiesta, “interagisce” col DataBase Server e/o Content Server, per poi inviare la relativa risposta HTTP / HTTPS al primo WEB Server.
Il primo WEB Server invia tale risposta HTTP / HTTPS al Client (WEB browser) che la interpreta, rendendola visibile all’Utente.
Il Portale WEB può anche comunicare direttamente con l’SMTP Server (presente anch'esso nella DMZ).

DESCRIZIONE E GUIDA DEL SITO WEB

La prima pagina web visualizzata sarà la Home page, raggiungibile all'indirizzo radice ('/').
Attraverso la Home page è possibile raggiungere la pagina di registrazione e / o di login.
Nella pagina di registrazione è possibile inserire nel database nuovi utenti, che dovranno inserire valori come email (univoca), password (hashing delle password salvate nel database), nome, cognome, ruolo ricoperto nel mondo del lavoro e Città di residenza.
Una volta che l'utente si è registrato, sarà reindirizzato nella pagina di Login (ovviamente ci si può andare anche senza essersi registrati) e inserire le proprie credenziali (email e password), e se presenti nel database, avverrà un reindirizzamento in un'area ('/ feed') in cui può visualizzare i suoi post più quelli dei suoi amici e mettere mi piace ad ogni post. In questa pagina è possibile anche pubblicare nuovi post.
Sempre in quest'area, possiamo trovare una barra di navigazione orizzontale, grazie alla quale possiamo raggiungere diverse pagine ed eseguire diverse azioni, come ad esempio:

- Accesso alla propria scheda profilo, modificarla, visualizzare i propri post, mettere mi piace ad essi ed accettare / rifiutare eventuali richieste di collegamento esterne.

- Cercare altri utenti e visualizzare (cliccando) le schede dei loro profili, visualizzare i loro post e connettersi con loro.

- Logout.

- Accesso alla Home page e all'area principale ('/ feed').

Nella pagina di login, l'utente può anche recuperare la sua password, inserendo la sua email e ed una nuova password.

FUNZIONALITÀ' DI LOGIN, LOGOUT E REGISTRAZIONE

Frontend

Registrazione: (file: registrazione/index.blade.php,
pagina contenente il Form della Registrazione)

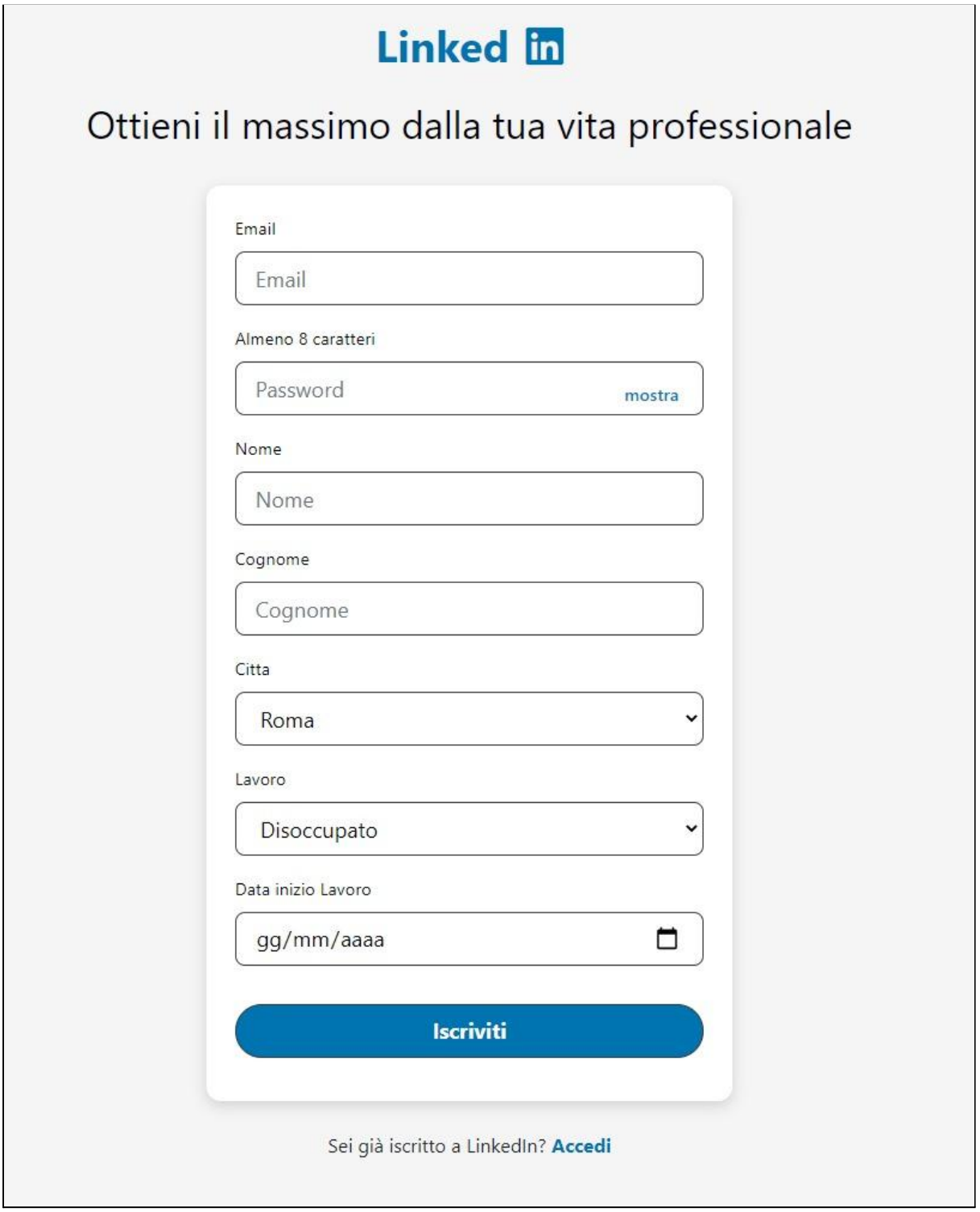
```
<!DOCTYPE html>

@php
    $selectors = selectors();
@endphp

<html lang="{{ $selectors['lang'] }}" dir="{{ $selectors['dir'] }}">
<head>
    <x-head title="Iscriviti" />
    <link rel="stylesheet" type="text/css" href="/css/registrazione/index.css" />
```

```
</head>
<body>
<div class="{{ $selectors['container'] }}">
   <div class="{{ $selectors['col'] }}4">
      <div class="{{ $selectors['row'] }}">
         <x-noscript />
         <div class="{{ $selectors['col'] }}">
            <x-title />
            <div class="{{ $selectors['row'] }} mt-3">
               <h5 id="subtitle">
                  Ottieni il massimo dalla tua vita professionale
               </h5>
            </div>
         </div>
         <div class="col-xs-11 col-sm-8 col-md-6 col-lg-5 col-xl-4">
            <div class="{{ $selectors['row'] }} mt-4">
               <div id="form-card" class="{{ $selectors['col'] }} p-4">
                  <form method="POST" action="{{ $selectors['action']
                                                }}/registrazione" id="profile-form">
                     @csrf
                     <x-errors />
                     <div class="{{ $selectors['col'] }}">
                        <div class="row">
                           <x-email label="{{ ucfirst($selectors['email']) }}"/>
                        </div>
                     </div>
                     <div class="{{ $selectors['col'] }}3">
                        <div class="row">
                           <x-password label="Almeno {{ $selectors['passLen'] }}
                                              caratteri"/>
                        </div>
                     </div>
                     <div class="{{ $selectors['col'] }}3">
                        <div class="row">
                           <x-text label="nome" />
                        </div>
                     </div>
                     <div class="{{ $selectors['col'] }}3">
                        <div class="row">
                           <x-text label="cognome" />
                        </div>
                     </div>
                     <div class="{{ $selectors['col'] }}3">
                        <div class="row">
                           <label for="{{ $selectors['select2'] }}">
                              {{ ucfirst($selectors['select2']) }}
                           </label>
                           <select
```

```
                        class="{{ $selectors['input'] }}"
                        name="{{ $selectors['select2'] }}"
                        required>
                            @component('components.option', ['data' => $citta])
                            @endcomponent
                        </select>
                    </div>
                </div>
                <div class="{{ $selectors['col'] }}3">
                    <div class="row">
                        <label for="{{ $selectors['select1'] }}">
                            {{ ucfirst($selectors['select1']) }}
                        </label>
                        <select
                            class="{{ $selectors['input'] }}"
                            id="{{ $selectors['select1']}}"
                            name="{{ $selectors['select1'] }}">
                        @component('components.option', [
                          'data' => $lavori,
                          'selected' => 'Disoccupato'
                          ])
                        @endcomponent
                        </select>
                        </div>
                </div>
                <div class="{{ $selectors['col'] }}3">
                    <div class="row">
                        <x-date />
                    </div>
                </div>
                <x-submit text="Iscriviti" mt="{{ 4 }}" />
            </form>
        </div>
    </div>
</div>
<x-footer />
</div>
</div>
<script src="{{ asset('js/app.js') }}"></script>
<script type="text/javascript" src="js/login/index.js"></script>
<script type="text/javascript" src="js/registrazione/index.js"></script>
<x-alert
    msg="{{ $msg }}"
    ref="{{ $ref }}"
/>
</body>
</html>
```


Linked
Ottieni il massimo dalla tua vita professionale
Email
Email
Almeno 8 caratteri
Password
mostra
Nome
Nome
Cognome
Cognome
Citta
Roma
Lavoro
Disoccupato
Data inizio Lavoro
gg/mm/aaaa
Iscriviti
Sei già iscritto a LinkedIn? Accedi

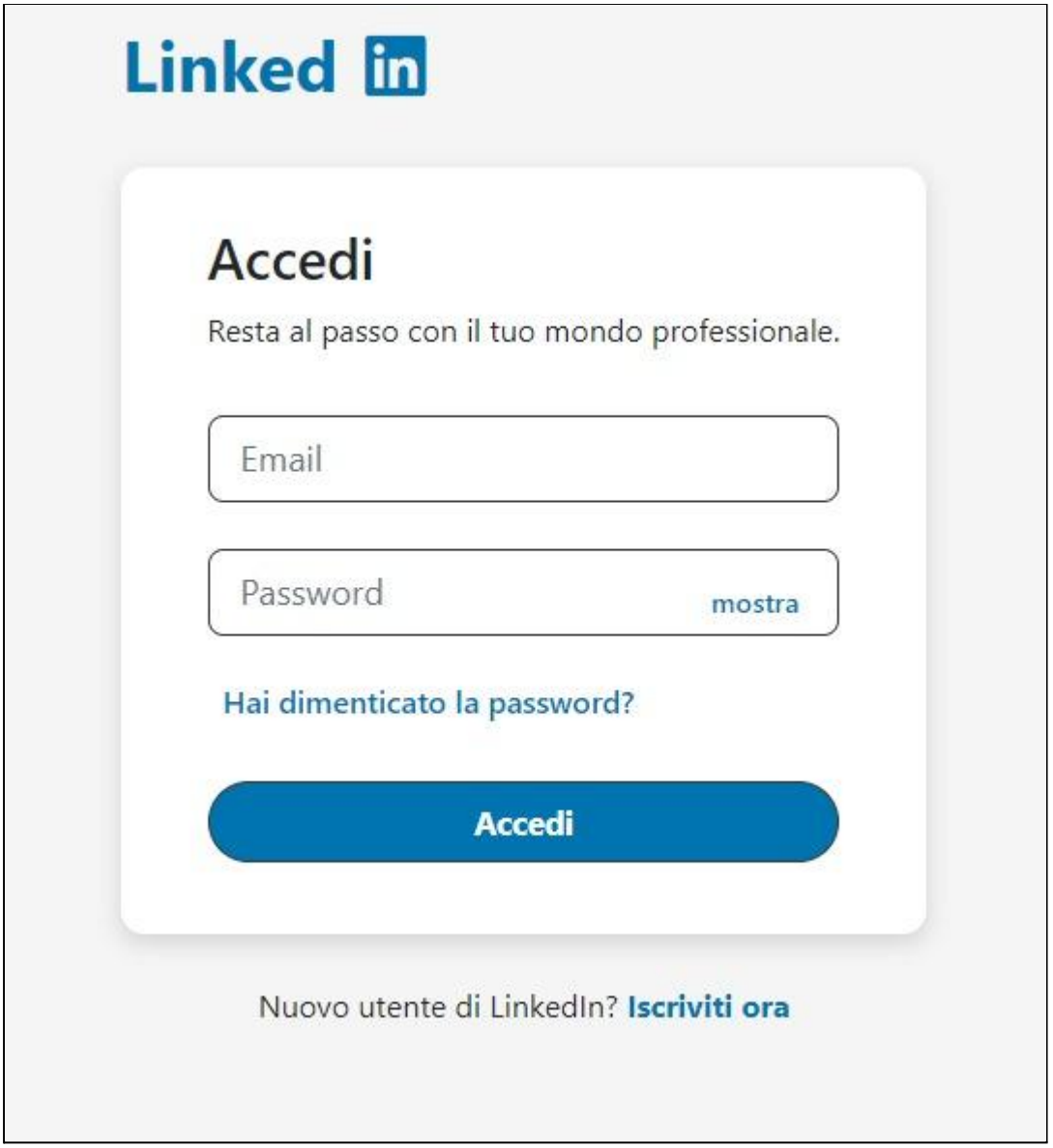
Login: (file: login/index.blade.php
pagina contenente il Form del Login)

```
<!DOCTYPE html>

@php
   $selectors = selectors();
@endphp

<html lang="{{ $selectors['lang'] }}" dir="{{ $selectors['dir'] }}">
<head>
   <x-head title="Accesso" />
   <link rel="stylesheet" type="text/css" href="/css/login/index.css" />
</head>
<body>
<div class="{{ $selectors['container'] }}">
   <div class="{{ $selectors['col'] }}4">
      <div class="{{ $selectors['row'] }}">
         <x-noscript />
         <div class="col-xs-12 col-sm-10 col-md-8 col-lg-6 col-xl-4">
            <div class="{{ $selectors['col'] }}">
               <x-title row="justify-content-xl-start" />
            </div>
            <div class="{{ $selectors['row'] }} mt-4">
               <div id="form-card" class="p-5">
                  <div class="{{ $selectors['col'] }}" id="header">
                     <div class="row">
                        <h2>Accedi</h2>
                     </div>
                     <div class="row">
                        <p>Resta al passo con il tuo mondo professionale.</p>
                     </div>
                  </div>
                  <form method="POST" action="/feed">
                     @csrf
                     <x-errors />
                     <div class="{{ $selectors['col'] }}">
                        <div class="row">
                           <x-email />
                        </div>
                     </div>
                     <div class="{{ $selectors['col'] }}4">
                        <div class="row">
```

```
                        <x-password />
                    </div>
                </div>
                <div class="{{ $selectors['col'] }}4 ml-2">
                    <div class="row">
                        <h6 id="passwordDimenticata" class="primaryTXT">
                            Hai dimenticato la password?
                        </h6>
                    </div>
                </div>
                <x-submit text="Accedi" mt="{{ 3 }}" />
            </form>
        </div>
        <x-footer login="{{ true }}" />
    </div>
</div>
</div>
</div>
</div>
<script src="{{ asset('js/app.js') }}"></script>
<script type="text/javascript" src="{{ asset('js/login/index.js') }}"></script>
<x-alert
    msg="{{ $msg }}"
    ref="{{ $ref }}"
/>
</body>
</html>
```


Linked in
Accedi
Resta al passo con il tuo mondo professionale.
Email
Password
mostra
Hai dimenticato la password?
Accedi
Nuovo utente di LinkedIn? Iscriviti ora

Backend

Nome File: app/Htpp/controllers/UtenteController.php (qui viene gestita la logica di business).

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Contracts\View\Factory;
use Illuminate\Contracts\View\View;
use Illuminate\Contracts\Foundation\Application;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Log;
use Illuminate\Support\Facades\Cookie;
use App\Models\Utente;
use App\Models\Lavoro;
use App\Models\Citta;


class UtenteController extends Controller {

   public function logout(Request $req): RedirectResponse {
     $req
        ->session()
        ->forget('utente');
     Log::warning('Finished User-Session.');
     return redirect()
        ->route('login');
  }
```

```
public function insert(Request $req): RedirectResponse {
     $req->validate([
        'email' => ['email', 'required', 'unique:utente','min:2',
'max:35'],
        'password' => ['required', 'min:8', 'max:8'],
        'nome' => ['required', 'min:3', 'max:45'],
        'cognome' => ['required', 'min:3', 'max:45'],
        'citta' => ['required'],
     ], [
        'email.required' => 'Email is Required.',
        'email.min' => 'Email almeno 2 caratteri.',
        'email.max' => 'Email massimo 35 caratteri.',
        'email.unique' => 'Utente già Registrato, è possible effettuare
                           il Login.',
        'password.required'  => 'Password is Required.',
        'password.min'  => 'Password con 8 caratteri.',
        'password.max'  => 'Password con 8 caratteri.',
        'nome.required' => 'Nome is Required.',
        'nome.min' => 'Nome almeno 3 caratteri.',
        'nome.max' => 'Nome massimo 45 caratteri.',
        'cognome.required' => 'Cognome is Required.',
        'cognome.min' => 'Cognome almeno 3 caratteri.',
        'cognome.max' => 'Cognome massimo 45 caratteri.',
     ]);
     $email = $req->email;
     if(isLogged($email))
        return redirect()
           ->route('login', ['msg' => 'log']);
     else {
        insertUtente($req);
        Log::debug('New User Interted.');
        return redirect()
           ->route('login', ['msg' => 'reg']);
     }
```

```
    }

    public function logResult(Request $req): Factory | View |
RedirectResponse | Application {
        $req->validate([
            'email' => ['email', 'required', 'min:2', 'max:35'],
            'password' => ['required', 'min:8', 'max:8'],
        ], [
            'email.email' => 'Inserisci Email valida',
            'email.required' => 'Email is Required.',
            'email.min' => 'Email almeno 2 caratteri.',
            'email.max' => 'Email massimo 35 caratteri.',
            'password.required'  => 'Password is Required.',
            'password.min'  => 'Password con 8 caratteri.',
        ]);
        $email = $req->email;
        $password = $req->password;
        $logged = isLogged($email, $password);
        if($logged) {
              $utente = Utente::all([
                  'id',
                  'email',
                  'password',
                  'nome',
                  'cognome',
              ])
                  ->where('email', $email)
                  ->first();
              $utente->password = $password;

             $req
                  ->session()
                  ->put('utente', $utente);
              Log::info("New User-Session started ($email)");
            $utente_id = $req->session()->get('utente')->id;
            return $this->feed($utente_id);
```

```
            } else
                return redirect()
                    ->route('login', ['msg' => 'not-reg']);
        }
        public function feed(int $utente_id): Factory | View | Application {
            return view('feed.index', [
                'posts' => getAllPosts($utente_id),
                'profile_id' => null
            ]);
        }
    }

    ?>
```

- **Nel metodo “insert” gestisco la logica relativa alla registrazione di un nuovo Utente prendendo i valori dai campi di un Form (metodo Post). Sono presenti controlli Client-side (required, min, max...) e Server side (tramite il metodo “validate” della classe Request), quest’ultimo se rileva problemi reindirizza l’Utente al Form mostrandogli gli errori commessi. Successivamente c’è un controllo sull’Email inserita nel Form, se tale Email è già presente nel DataBase(perchè deve essere univoca) l’Utente viene reindirizzato alla pagina di Login con un messaggio che gli indica che è già loggato, altrimenti, se l’Email non è presente nel DataBase, l’Utente viene reindirizzato sempre alla pagina di Login, ma con un messaggio che gli indica che la Registrazione (inserimento Utente nel DataBase) è avvenuta con successo e che può procedere con il Login.**

  N.B. = Nel DataBase le Password vengono memorizzate nella versione criptata (bcrypt).

- **Nel metodo “logResult” gestisco la logica relativa al Login di un Utente prendendo i valori da un Form (metodo Post).**
  Sono presenti controlli Client-side (required, min, max...) e Server side (tramite il metodo “validate” della classe Request), quest’ultimo se rileva problemi reindirizza l’Utente al Form mostrandogli gli errori

commessi. Successivamente c’è un controllo su Email e Password (nel Database viene salvata la parte criptata) inserite nel Form, se tale controllo da esito positivo l’Utente viene reindirizzato nella sua “area personale” ed alcuni dei suoi dati vengono memorizzati nella sessione corrente, altrimenti rimane fermo alla pagina di Login e gli viene mostrato un messaggio che gli indica che non è un utente registrato, ma può provvedere a farlo.

- Nel metodo “logout” semplicemente vengono cancellati dalla sessione corrente i dati dell’Utente precedentemente loggato, in più esso viene reindirizzato alla pagina di Login.

CARATTERISTICHE SITO WEB

- Portable, il ”motore” del sito WEB è stato testato in 3 diversi SO:
  - ❏ Windows.
  - ❏ Linux.
  - ❏ MacOs.
- Robusto (controlli Client-side, Server-side ed anche nel database tramite vincoli di CHECK e TRIGGER).
- Layout Responsive.
- Facile da trasportare da una macchina all'altra (sia database che applicazione).
- Sicuro
- Facile da utilizzare.
- Graficamente vivace.

TECNOLOGIE UTILIZZATE

- **PHP 8.0.6**
- **Laravel 8**
- **MySQL 8.0.25**
- **Bootstrap 4.6.0**
- **JavaScript ES11 (2020)**
- **JQuery 3.6.0**
- **SASS 1.15.2**
- **Composer 2.0.10**
- **NPM 7.14.0**
- **GIT 2.31.1, GITHUB**
- **Windows, Ubuntu.**
- **PHPstorm, Visual-studio Code.**

COMPONENTI LARAVEL UTILIZZATI

- **Route**
- **Controller**
- **Model**
- **Migration**
- **Seeder**
- **Blade (con componenti ed altro...)**
- **LiveWire**
- **MiddleWare**
- **Session**
- **Cookie**
- **Command**
- **Storage**
- **Log**
- **Test**
- **Hash**
- **DB**
- **Http**
- **Console**

[GitHub Repository (clicca)]

Studente: Matteo Lambertucci 5F. 29/05/2021.