SOLUZIONE ESAME DI STATO 2014/2015
INFORMATICA

©Giselda De Vita 2015

La traccia di esame, pubblicata dal Ministero, si trova al seguente link: esame 2015
Di seguito si riporta la soluzione della prima parte del compito.

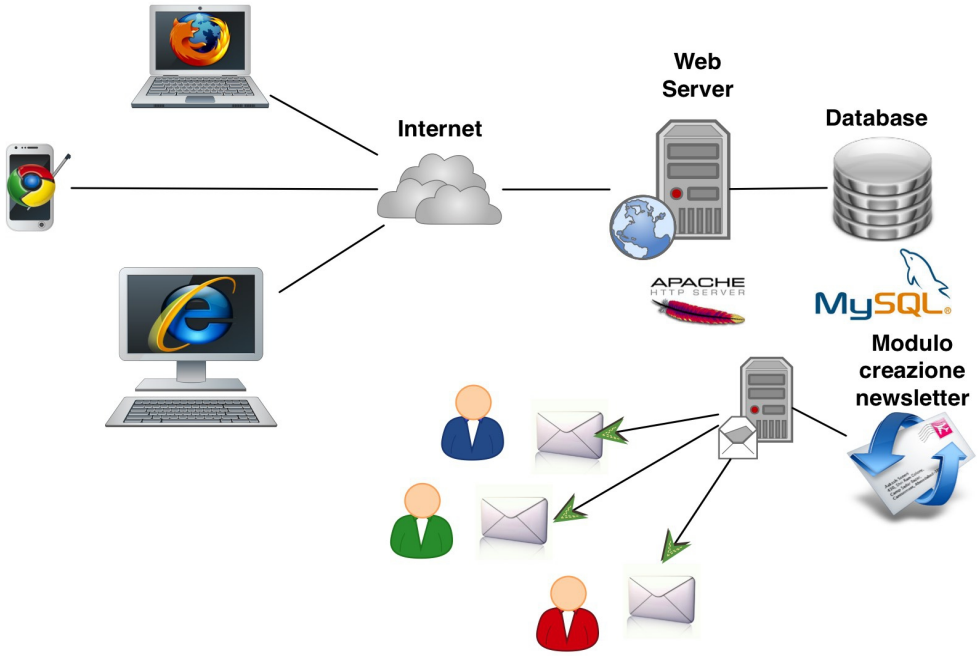
1 Analisi

Un'analisi della realtà di riferimento individuando le possibili soluzioni e scelga quella che a suo motivato giudizio è la più idonea a rispondere alle specifiche indicate

Quest'anno, viene illustrata una '*web-comunity per condividere dati e commenti relativi a eventi dal vivo di diverse categorie, ad esempio concerti, spettacoli teatrali, balletti, ecc. che si svolgono in Italia.'* [cit. da traccia Ministeriale]

La traccia è sufficientemente chiara e presenta solo qualche piccola ambiguità su cui si dovranno prendere delle decisioni. Si descrive un sistema da cui si evincono due entità principali gli **utenti** e gli **eventi**, su questo punto credo che pochi studenti abbiano avuto dei dubbi.

Essendo una 'web' community, ovviamente si tratta di dover realizzare un sito web su cui degli utenti si registrano.
Il Sistema informativo proposto dovrà essere realizzato come applicazione web e realizzare un'applicazione Web si traduce nel mettere su di un WEB SERVER delle pagine fruibili da un browser (siano esse html, php, asp, jsp etc.)
L' architettura di riferimento più economica, solida, che risponde agli standard di mercato ed al limitato numero di utenti da gestire, è quella disegnata nella figura che segue: APACHE, linguaggio PHP e Base Dati MySQL.



Il sistema, oltre al sito web, avrà un modulo dedicato alla estrazioni dalla base dati di alcune informazioni che verrano organizzate in mail, poi inviate ai vari utenti iscritti alla web community.
La traccia è molto chiara sul fatto che le **newsletter** sono personalizzate per ogni singolo utente.
A parte la considerazione che si dovrà prevedere un modulo dedicato alla creazione delle newsletter, all'interno della progettazione di tutta la web community, non c'è molto altro da dire visto che la traccia non chiede nulla sulla implementazione di questa parte.

Ci si deve soffermare, invece, sulla creazione della base dati, cuore della web community.
Ogni candidato, in questo primo punto di analisi, ha la possibilità di giustificare le decisioni che prenderà in seguito sul disegno della base dati. Per i più preparati potrebbe essere un'occasione per mostrare una capacità di approfondimento e di vaglio di varie soluzioni che potrebbe comportare una maggiore valutazione dell'elaborato.

Una breve considerazione sulla possibilità degli utenti anonimi di effettuare solo delle query sulla base dati della community, mentre gli utenti registrati possono inserire sia gli eventi che i commenti. Questa diversità verrà gestita a livello applicativo facendo registrare gli utenti con una login ed una password e, tenendo una sessione aperta, gli verrano mostrati i menù per inserire commenti, votare o inserire eventi.
Potremo anche, dimostrando di conoscere anche la sicurezza a livello database, pensare di definire due tipologie di USER a livello database (un ANONYMOUS e un REGISTERED) associati a dei GRANT (permessi) diversi sulla base dati, i primi potranno solo interrogare (GRANT SELECT) gli altri potranno sia inserire che modificare i dati della base dati (INSERT E UPDATE - GRANT ALL). Probabilmente pochi studenti avranno affrontato questa complicazione aggiunta dalla traccia.

2 Modello concettuale

uno schema concettuale della base di dati;

Il cuore della progettazione di una base dati. Nel momento in cui si è sviluppato un modello concettuale coerente e completo, il resto della traccia viene di conseguenza a cascata.

Da una prima lettura della traccia due entità cardine appaiono chiare:

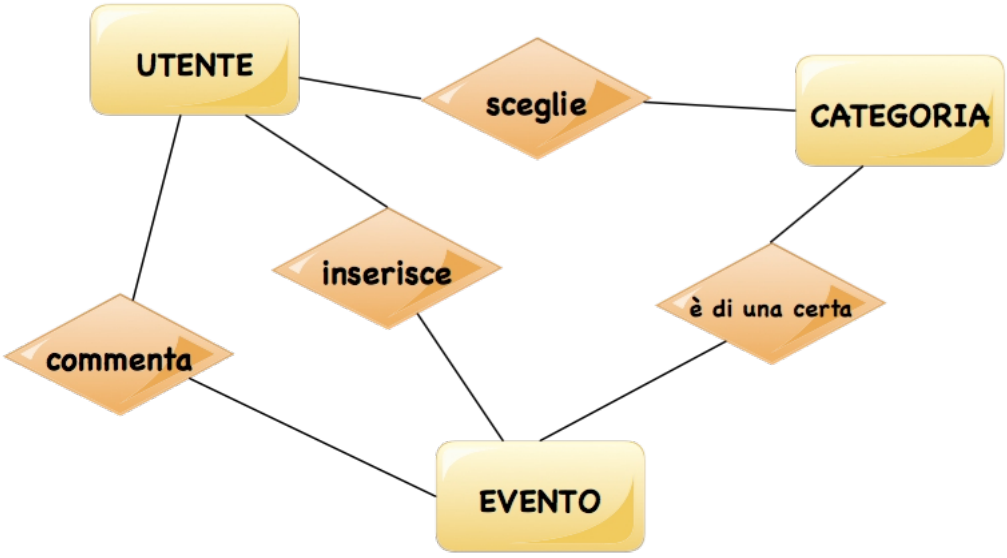
Su queste due entità non ci sono dubbi. Tutta la web community ruota intorno a queste due entità per cui dobbiamo ora decidere come legarle e eventuali entità di contorno.

L'entità CATEGORIA
Ad una prima lettura della traccia, CATEGORIA appare come un semplice attributo dell'EVENTO ma poi, considerando che l'utente, in fase di registrazione, dovrà scegliere tra le categorie di preferenza, su cui ricevere le newsletter, si configura come una vera e propria entità legata sia all'UTENTE che all'EVENTO.

Modello concettuale numero 1

Per semplificare al massimo prendendo la traccia alla lettera, dobbiamo assumere che un utente può inserire un solo commento e un solo voto! Per questo motivo tutto il nostro diagramma concettuale si traduce nell'inserire due relazioni tra UTENTE, EVENTO e poi unire CATEGORIA alle due entità.

Su queste relazioni ci sono pochi dubbi, l'utente da una parte può inserire l'evento e dall'altra parte può commentare inserendo un post con voto sull'evento.



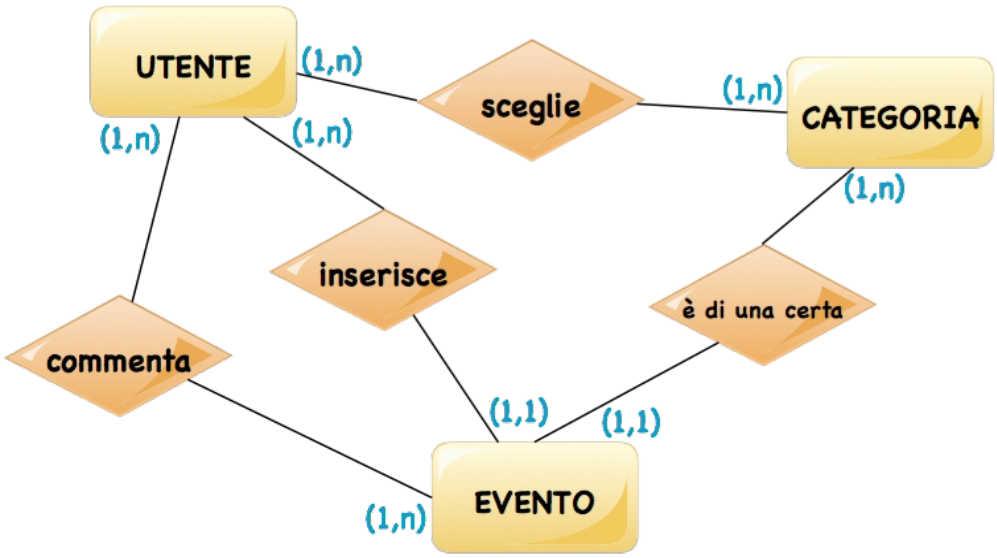
Ragioniamo sulle cardinalità e sugli attributi:

- Un UTENTE può inserire più di un EVENTO – un singole EVENTO viene inserito da un solo UTENTE (**uno** a **molti**)
- Un UTENTE può commentare più di un EVENTO – un EVENTO può essere commentato da tanti UTENTI (**molti** a **molti**)

- Un UTENTE può scegliere una o più CATEGORIE – una CATEGORIA può essere scelta da più UTENTI (**molti** a **molti**)
- Un EVENTO appartiene ad una CATEGORIA – una CATEGORIA può essere associata a più EVENTI (**uno** a **molti**)

Ricapitolando:

- la relazione INSERISCE è 1 a n (**uno** a **molti**)
- la relazione COMMENTA è n a n (**molti** a **molti**)
- la relazione SCEGLIE è 1 a n (**molti** a **molti**)
- la relazione E' DI UNA CERTA è 1 a n (**uno** a **molti**)



Ragioniamo sugli attributi degli utenti (chiamati in genere membri) leggendo dalla traccia su: '*i membri stessi della community, che si registrano sul sito fornendo un nickname, nome, cognome, indirizzo di e-mail e scegliendo una o più categorie di eventi*'
In questa frase abbiamo trovato i primi 4 attributi ma anche la **chiave primaria**. Sappiamo bene che su una community si deve scegliere un nickname univoco, infatti se proviamo a registrarci su un sito web con un nick name già utilizzato ci viene restituito un messaggio d'errore e ci viene chiesto di scegliere un altro nickname.
Leggendo la frase sulla newsletter: '*Ogni membro iscritto riceve periodicamente per posta elettronica una newsletter, emessa automaticamente dal sistema, che riporta gli eventi delle categorie da lui scelte, che si svolgeranno nella settimana seguente nel territorio provinciale dell'utente*' capiamo che c'è anche un ulteriore attributo in cui viene indicata la provincia di residenza dell'utente.
Essendo degli utenti che dovranno registrarsi su di un sito web, appare ragionevole inserire anche l'attributo password.

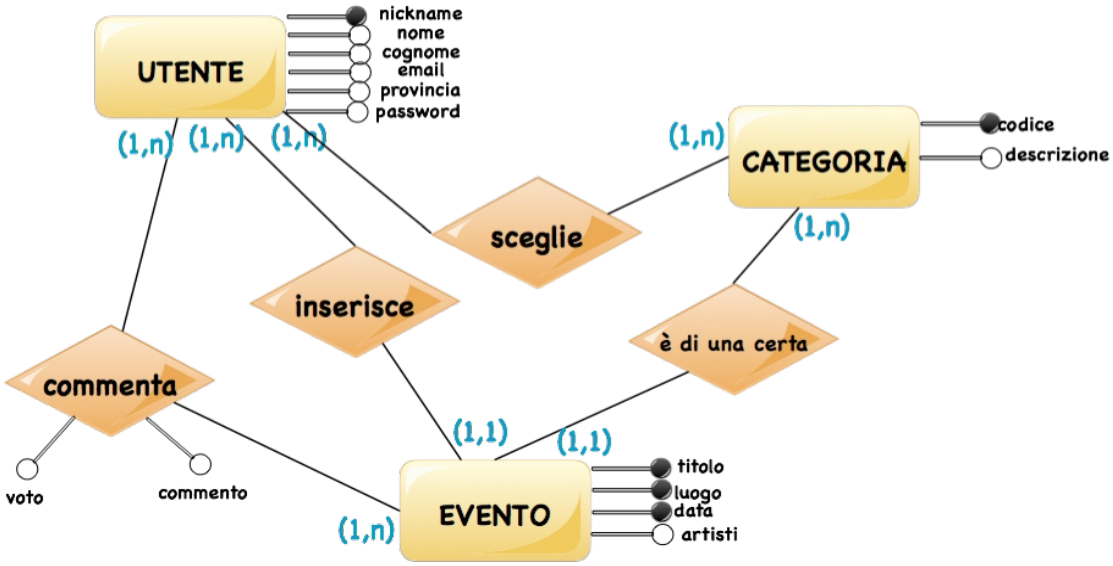
Ragioniamo ora sugli attributi dell'EVENTO. Leggendo la traccia: *'i dati di un nuovo evento, per il quale occorre specificare categoria, luogo di svolgimento, data, titolo dell'evento e artisti coinvolti,'*

Abbiamo trovato 5 attributi per l'evento, non è chiarissima ancora la chiave primaria (magari potrebbe essere il titolo + data + luogo) ma non occorre preoccuparci troppo, sappiamo già che nel livello logico-fisico assegneremo un ID univoco all'evento.
Leggendo solo questa frase e considerando solo EVENTO, CATEGORIA appare come un semplice attributo.
Sicuramente molti studenti hanno pensato alla CATEGORIA come ad un'entità ed è corretto, pensando al collegamento che la categoria ha con l'UTENTE.
Se fosse stata solo legata all'EVENTO, poteva essere pensato anche come un semplice attributo tipizzato dell'EVENTO di tipo ENUM o SET (per MySQL), ovvero una stringa che può assumere solo determinati valori predefiniti come :'concerti', 'spettacoli teatrali', 'balletti'. Se qualche studente avesse avuto questa visione, probabilmente poi ha risolto in maniera diversa la registrazione dell'utente e la sua scelta delle categorie di interesse, che poi era utile solo ai fini delle newsletter.

Sulla relazione 'COMMENTA' appare chiaro che ci siano almeno due attributi: '*sia scrivendo un post con un commento ed un voto (da 1 a 5) su un evento*'

<u>L'entità POST</u>
Molti studenti avranno inserito nel diagramma concettuale l'entità POST.
Non lo ritengo un errore, sia perché normalmente si è portati a non disegnare gli attributi su una relazione ma a far diventare una relazione con attributi una entità, ma anche per il dubbio sulla cardinalità di questo POST.
La traccia sembra fissare la possibilità di un utente ad inserire **un** solo voto ed **un** solo commento, per cui va bene lasciarlo come relazione. Però se sul voto possiamo essere d'accordo (noi utenti possiamo modificare un voto dato ma non inserirne più di uno), sul commento appare inverosimile. Perché non posso rispondere ad un commento di un altro utente o inserire un commento prima e dopo l'evento?
Capisco la scelta di chi ha deciso questa restrizione nella traccia, pensare a più commenti e un solo voto complica parecchio la progettazione, si dovrebbero dividere dal voto i POST che, in questo caso, diventerebbero a tutti gli effetti un'altra entità.

Diagramma E-R finale prima soluzione:



Possibile Modello concettuale numero 2
La soluzione precedente a cui si aggiunge l'entità POST al posto della relazione COMMENTA.
Se si pensa che si possano inserire più POST e un solo VOTO, allora anche VOTO diventerà o una relazione VOTA con attributo VOTO o una vera e propria entità.

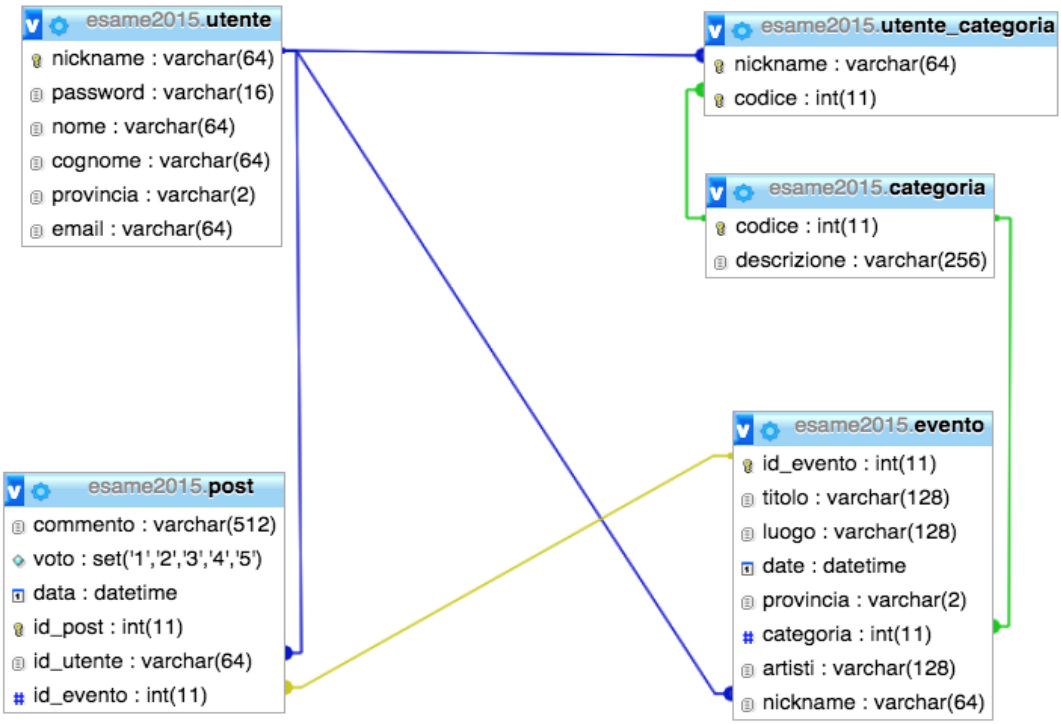
Possibile Modello concettuale numero 3
Aggiunta delle entità non fondamentali, ma che non rappresentano un errore, come: ARTISTA e LUOGO.

Per i punti successivi del compito, porteremo avanti il **modello concettuale numero 1**, che preferisco per compattezza, semplicità e soprattutto rende lo svolgimento delle query molto agevole.
Spesso, nella progettazione reale della base dati, si tende a non normalizzare all'eccesso per avere una maggiore performance sull'esecuzione delle query.

3 MODELLO LOGICO

uno schema logico della base di dati

Creato il modello concettuale, il modello logico ne consegue.



Le relazioni vengono tradotte. Le relazioni uno a molti portano le foreign key nelle tabelle a molteplicità n (EVENTO ha la FK verso CATEGORIA e l'UTENTE che lo ha inserito).
Le relazioni **molti a molti** si traducono aggiungendo due tabelle, una che traduce la relazione COMMENTA che chiameremo proprio POST e una tabella UTENTE_CATEGORIA.

4 MODELLO FISICO

la definizione in linguaggio SQL di un sottoinsieme delle relazioni della base di dati in cui siano presenti alcune di quelle che contengono vincoli di integrità referenziale e/o vincoli di dominio, laddove presenti

Quest'anno, essendo presente la seconda parte, viene chiesto solo un sottoinsieme di linguaggio DDL per la creazione di sole due tabelle, legate tra di loro tramite Foreign key.

Di seguito riporto tutto lo script SQL di creazione del database mysql, che serve per la parte seguente di implementazione, ma sarebbero bastate due tabelle.

```
CREATE TABLE IF NOT EXISTS `categoria` (
  `codice` int(11) NOT NULL AUTO_INCREMENT,
  `descrizione` varchar(256) NOT NULL,
  PRIMARY KEY (`codice`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
CREATE TABLE IF NOT EXISTS `evento` (
  `id_evento` int(11) NOT NULL AUTO_INCREMENT,
  `titolo` varchar(128) NOT NULL,
  `luogo` varchar(128) NOT NULL,
  `data` datetime NOT NULL,
  `provincia` varchar(2) NOT NULL,
  `categoria` int(11) NOT NULL,
  `artisti` varchar(128) NOT NULL,
  `nickname` varchar(64) NOT NULL,
  PRIMARY KEY (`id_evento`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=4 ;
CREATE TABLE IF NOT EXISTS `post` (
  `commento` varchar(512) NOT NULL,
  `voto` set('1','2','3','4','5') NOT NULL,
  `data` datetime NOT NULL,
  `id_post` int(11) NOT NULL AUTO_INCREMENT,
  `nickname` varchar(64) NOT NULL,
  `id_evento` int(11) NOT NULL,
  PRIMARY KEY (`id_post`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;
CREATE TABLE IF NOT EXISTS `utente` (
  `nickname` varchar(64) NOT NULL,
  `password` varchar(16) NOT NULL,
  `nome` varchar(64) NOT NULL,
  `cognome` varchar(64) NOT NULL,
  `provincia` varchar(2) NOT NULL,
  `email` varchar(64) NOT NULL,
  PRIMARY KEY (`nickname`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
CREATE TABLE IF NOT EXISTS `utente_categoria` (
  `nickname` varchar(64) NOT NULL,
  `codice` int(11) NOT NULL,
  PRIMARY KEY (`nickname`,`codice`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Constraints for table `evento`
ALTER TABLE `evento`
  ADD CONSTRAINT `evento_ibfk_2` FOREIGN KEY (`categoria`) REFERENCES `categoria`
(`codice`),
  ADD CONSTRAINT `evento_ibfk_1` FOREIGN KEY (`nickname`) REFERENCES `utente`
(`nickname`);
-- Constraints for table `post`
ALTER TABLE `post`
  ADD CONSTRAINT `post_ibfk_2` FOREIGN KEY (`id_evento`) REFERENCES `evento`
(`id_evento`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `post_ibfk_3` FOREIGN KEY (`nickname`) REFERENCES `utente` (`nickname`);
-- Constraints for table `utente_categoria`
ALTER TABLE `utente_categoria`
  ADD CONSTRAINT `utente_categoria_ibfk_1` FOREIGN KEY (`nickname`) REFERENCES `utente`
(`nickname`),
  ADD CONSTRAINT `utente_categoria_ibfk_2` FOREIGN KEY (`codice`) REFERENCES `categoria`
(`codice`);
```

5 Le query SQL

le seguenti interrogazioni espresse in linguaggio SQL:

a. elenco degli eventi già svolti, in ordine alfabetico di provincia

```
SELECT *
FROM evento
WHERE data<NOW()
ORDER BY provincia
```

b. elenco dei membri che non hanno mai inserito un commento

```
SELECT nickname
FROM utente
WHERE nickname NOT
IN (
SELECT nickname
FROM post
)
```

c. per ogni evento il voto medio ottenuto in ordine di categoria e titolo

```
SELECT AVG( voto ) , categoria, titolo
FROM post, evento
WHERE evento.id_evento = post.id_evento
GROUP BY categoria, titolo
```

d.i dati dell'utente che ha registrato il maggior numero di eventi

```
SELECT MAX( cont ) , nickname
FROM (
SELECT COUNT( * ) AS cont, nickname as nickname
FROM post
GROUP BY nickname ORDER BY cont DESC) as contapost
```

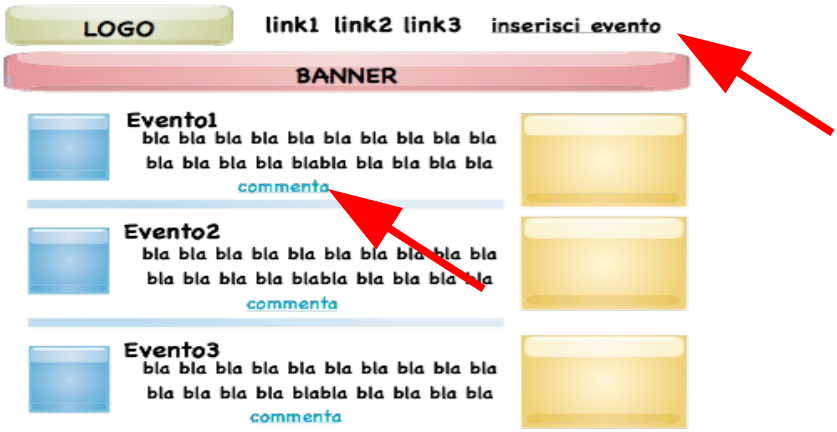
6 L'interfaccia utente e un pezzo di codice

il progetto della pagina dell'interfaccia WEB che permetta ad un utente registrato di svolgere le operazioni specificate

La traccia chiede di indicare l'interfaccia HTML che si potrebbe presentare ad un utente registrato alla community, che quindi ha la possibilità di inserire o di commentare un evento.

L'importante di questi punti molto tecnici, che solo nei compiti di informatica scritti si eseguono senza l'ausilio di un PC, personalmente credo sia l'impostazione corretta, ovvero sapere di massima com'è strutturata una pagina HTML.

Gli studenti potrebbero aver disegnato, una interfaccia classica da web community:



Nel nostro caso, in uno dei link c'è la possibilità di inserire un nuovo evento, e in un altro la possibilità di commentarlo.
Per dare dimostrazione di conoscere l'HTML, si poteva impostare una semplice form di inserimento dell'evento:

Il codice HTML della seguente FORM:

```
<html>
<head>
   <title>Inserisci Evento</title>
</head>
<body>
<p> <b>Inserisci Evento</b></p>
<form action="inserisci_evento.php" method="POST" name="form">
<table width="180" border="0">
 <tr>
   <td><label>Titolo</label> </td>
   <td><input name="titolo" id="nome" type="text" size="54" maxlength="54" /> </td>
 </tr>
 <tr>
   <td><label>Categoria</label> </td> <td>
 <select name="categoria">
 <option value="1">Balletto</option>
 <option value="2" selected>Concerto</option>
 <option value="3">Spettacolo teatrale</option></td>
</select>
 <tr>
   <td><label>Data</label> </td>
   <td><input name="giorno" id="giorno" type="text" size="2" maxlength="2" />/<input
name="mese" id="mese" type="text" size="2" maxlength="2" />/<input name="anno" id="anno"
type="text" size="4" maxlength="4" /> </td>
 </tr>
 <tr>
   <td><label>Ora</label> </td>
   <td><input name="ora" id="ora" type="text" size="5" maxlength="5" /> </td>
 </tr>
 <tr>
   <td><label>Luogo</label> </td>
   <td><input name="luogo" id="luogo" type="text" size="35" maxlength="35" /> </td>
 </tr>
 <tr>
   <td><label>Provincia</label> </td>
   <td><input name="provincia" id="provincia" type="text" size="2" maxlength="2"/></td>
 </tr>
 <tr>
   <td><label>Artisti</label> </td>
   <td><input name="artisti" id="artisti" type="text" size="58"  /></td>
 </tr>
 <tr >
   <td colspan="2" align="center"><br><input name="submit" type="submit" value="Inserisci
Evento" /> <input name="reset" type="reset" value="RESET" /></td>
   </tr>
</table>
</form>
</body>
</html>
```

7 Implementazione

la codifica in un linguaggio a scelta di un segmento significativo dell'applicazione Web che consente l'interazione con la base di dati.

Alla fine della prima parte, viene richiesto un pezzo di codice che implementa una delle funzioni richieste. Il candidato a questo punto può scegliere il linguaggio di programmazione e anche cosa implementare tra le visualizzazioni degli eventi, gli inserimenti dei commenti o degli eventi stessi.

Di seguito, l'implementazione in PHP dell'Inserimento dell'evento.

Notare bene le prime righe per impostare la sessione di un utente precedentemente registrato con login e password:

```
<?php session_start();
$utenteregistrato = $_SESSION['login_user']; ?>
<html>
      <head>
           Inserimento Evento <br>
      </head>
      <body>
<?php
   $titolo=$_POST[titolo];
   $data=$_POST[anno]."-".$_POST[mese]."-".$_POST[giorno];
   $ora=$_POST[ora].":00";
   $luogo=$_POST[luogo];
   $provincia=$_POST[provincia];
   $artisti=$_POST[artisti];
   $categoria=$_POST[categoria];

   $db=mysql_connect("localhost","root","root");

    if(!$db)
      echo "no connection";
   else
   {
      mysql_select_db("provola");

      $sql = "INSERT INTO `evento`(`id_evento`, `titolo`, `luogo`, `data`, `provincia`,
`categoria`, `artisti`, `nickname`) VALUES
(null,'".$titolo."','".$luogo."',CAST('".$data." ".$ora."' AS DATE),
'".$provincia."',".$categoria.",'".$artisti."','".$utenteregistrato:"')";

   if(mysql_query($sql)){
       echo "<br>Nuovo Evento inserito correttamente! " ;
    }
   else
    {
     echo "<br>Errore: "  . mysql_error($db);
    }
   mysql_close($db);
 }
      ?>
      </body>
   </html>
```