DOCUMENTAZIONE TECNICA

DESCRIZIONE PROGETTO

Si vuole realizzare un'applicazione di basi di dati per la gestione di elezioni mediante web.
Ogni elezione e caratterizzata da una data di inizio delle votazioni, una data di ne e una breve descrizione. Un'elezione viene indetta per eleggere una carica. Una carica e caratterizzata da una durata in anni e da una descrizione che ne definisce le responsabilità. Un utente può decidere di candidarsi per un'elezione solo se rispetta i seguenti requisiti:

- non è già candidato per un'altra elezione in corso
- non detiene, al momento della votazione, un'altra carica ottenuta in una precedente elezione
- ha detenuto la carica collegata all'elezione al piu una volta (la base di dati dovrà garantire il rispetto di questo vincolo)

Il vincitore di un'elezione e il candidato che ha ottenuto il maggior numero di preferenze il quale inizia istantaneamente a detenere la carica associata all'elezione.
L'applicazione dovrà supportare funzionalità specifiche per le seguenti categorie di utenza.
Utente non registrato. Ha la possibilità di visionare le elezioni in corso e lo storico delle elezioni passate. In particolare il numero di votanti sarà sempre disponibile per la visualizzazione mentre la classica dei candidati sarà visualizzabile solo dopo la chiusura delle votazioni.
Utente registrato. Ha accesso al sistema mediante credenziali e può interagire con il processo elettorale. Gli utenti registrati possono essere elettori oppure candidati (senza perdere la qualifica di elettore).
Amministratore. Ha funzioni di supervisione dell'applicazione, può disattivare gli utenti, creare nuove cariche e indire elezioni per ciascuna di esse (decidendo data di inizio e ne del periodo elettorale).

Si richiede di implementare tutte le seguenti funzionalità all'interno della base di dati mediante la definizione di opportune strutture (i.e., domini, viste, asserzioni, funzioni, procedure, trigger).
Gestione di un'elezione. Quando un utente amministratore ha generato una nuova elezione, gli utenti registrati possono candidarsi no alla data di inizio delle votazioni. Una volta iniziate le operazioni di voto, ogni utente registrato potrà dare una e una sola preferenza ad uno dei candidati per quella specifica elezione. Il voto dovrà essere segreto (la base di dati dovrà garantire il rispetto di questo vincolo: per un'elezione sarà possibile stabilire se un utente ha votato ma la base di dati NON dovrà memorizzare la preferenza espressa).
Rinnovo automatico di cariche. Allo scadere di una carica, l'amministratore potrà invocare una procedura della base di dati che si occuperà di indire una nuova elezione per quella carica in maniera automatica utilizzando parametri predefiniti.
Visualizzazione dati storici. Al termine del periodo elettorale, ogni utente (registrato e non registrato) potrà visualizzare l'esito delle elezioni e dovranno essere mostrate alcune statistiche (e.g., numero di votanti, numero di schede bianche) oltre alla classica completa dei candidati con il numero di voti totali ottenuti da ciascuno di essi. Per migliorare le prestazioni dell'applicazione lo studente dovrà costruire opportune viste materializzate che verranno aggiornate mediante trigger.
Ricerca all'interno dello storico delle elezioni. Ogni utente (registrato e non) avrà la possibilità di accedere ad una specifica funzione di ricerca sulle elezioni passate. Dovràa essere possibile cercare le elezioni almeno secondo i seguenti criteri:

- per candidato: scelto un candidato, visualizzare tutte le elezioni alle quali ha partecipato e i risultati da lui ottenuti
- per carica: scelta una carica, visualizzare lo storico di tutti gli utenti che l'hanno detenuta ed

il numero di voti da essi ottenuti
- per periodo di votazione: scelto un periodo (caratterizzato da una data d'inizio e una data di fine) visualizzare tutte le elezioni che si sono svolte in tale periodo (con la possibilità di vedere il dettaglio dei risultati per ognuna di esse)

L'applicazione web si fa carico di tutte le funzioni di visualizzazione e di inserimento/modica dei dati. Ogni categoria d'utenza dispone di una propria interfaccia nella quale trovano posto le funzionalità ad essa dedicate. E' richiesto che l'inserimento dei dati sia quanto più possibile assistito dall'applicazione. Inoltre, l'applicazione web offre all'utente la possibilità di invocare l'esecuzione delle funzioni/procedure definite all'interno della base di dati (in base ai privilegi di ciascuna categoria d'utenza).

SCELTE DI PROGETTO

Un utente amministratore potrà usufruire di tutte le funzionalità di un utente elettore.

PROGETTAZIONE CONCETTUALE

Sono state individuate le seguenti entità:
Utente: Per gestire le informazioni relative agli utenti della piattaforma elettorale. Gli utenti possono essere Amministratori, Elettori o Candidati.
Gli attributi di utente sono:
- Nome: di tipo stringa per indicare il nome dell'utente
- Cognome: di tipo stringa per indicare il cognome dell'utente
- User: di tipo stringa. L'username dell'utente per poter accedere alla piattaforma. L'username deve essere unico all'interno della basi di dati. Username è anche chiave primaria di utente.
- Password: di tipo stringa. È la password che l'utente usa per accedere alla piattaforma.
- Admin: di tipo booleano. Se settato a 1, indica l'amministratore del sistema.
- Banned: di tipo booleano. Indica se l'utente è attivo (1) oppure se è disattivato (0)
- Cand: di tipo booleano. Indica se l'utente può candidarsi (1) oppure se è solo un elettore (0)

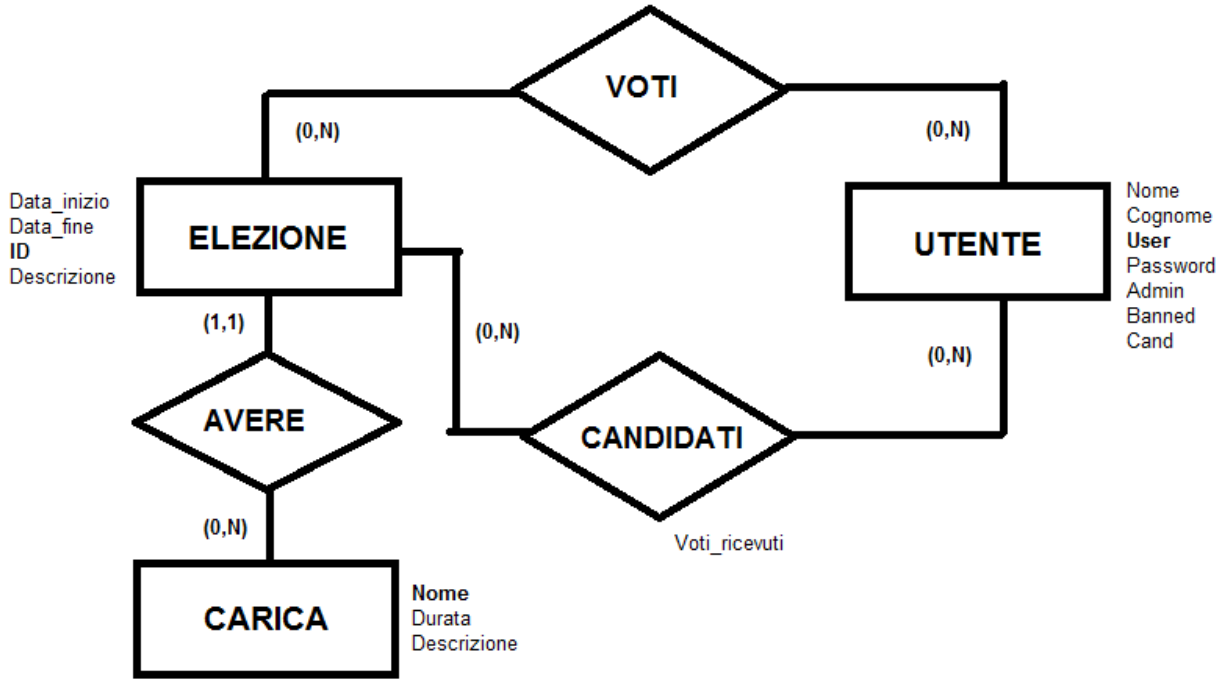
Carica: Per gestire le informazioni relative ad una carica elettorale.
Gli attributi di carica sono:
- Nome: di tipo stringa. Indica il nome della carica. È anche chiave primaria dell'entità Carica.
- Durata: di tipo intero. Indica la durata in anni in cui il candido eletto detiene la carica.
- Descrizione: di tipo testo per una descrizione delle funzioni che svolge la carica.

Elezione: Per gestire le informazioni relative alle elezioni.
Gli attributi di elezione sono:
- Data_inizio: di tipo data e indica l'inizio del periodo di votazione.
- Data_fine: di tipo data e indica la fine del periodo di votazione.
- ID: di tipo intero. È un codice univoco per identificare ciascuna elezione. È anche chiave primaria dell'entità
- Descrizione: di tipo testo. Contiene una descrizione dell'elezione.

Sono state individuate le seguenti associazioni:
Candidati: tra utente e elezione, di tipo N:M. Un utente può candidarsi a più elezioni e un'elezione è formata da più candidati. Voti_ricevuti è un attributo dell'associazione, di tipo intero, e indica il numero di voti ricevuti dal candidato in suddetta elezione.

Voti: tra utente e elezione, di tipo N:M. Un utente può votare per più elezioni e una elezione ha più votanti. Si è deciso di non usare un attributo con la preferenza, in modo da garantire la segretezza del voto.
Avere: tra carica e elezione, di tipo N:1, in quanto una carica può avere più elezioni, mentre un'elezione è per una e una sola carica.

Ne deriva, quindi, il seguente **schema concettuale**:



SCHEMA RELAZIONALE

Dal precedente schema logico, si può determinare il seguente schema relazione. Le chiavi primarie sono sottolineate, mentre le chiavi esterne sono in grassetto.

Utente(User, Nome, Cognome, Password, Admin, Banned, Cand)
Elezione(ID, Data_inizio, Data_fine, Descrizione, **Carica_Nome**)
Carica(Nome, Durata, Descrizione)
Candidati(**User, ID_Elezione**, Voti_ricevuti)
Voti(**User, ID_Elezione**)

PRODOTTI SOFTWARE E LINGUAGGI USATI

Per gestire la base di dati è stato usato il DBMS MySQL Workbench 5.2.47,
mentre per quanto riguarda l'applicazione web è stato usato un server Apache, mentre le pagine web sono state scritte con il linguaggio PHP.

Le tabelle sono state scritte con il linguaggio DDL SQL

```
CREATE TABLE `utente` (
  `Nome` varchar(45) DEFAULT NULL,
  `Cognome` varchar(45) DEFAULT NULL,
  `User` varchar(45) NOT NULL,
  `Password` varchar(45) DEFAULT NULL,
  `Admin` tinyint(1) DEFAULT '0',
  `Banned` tinyint(1) DEFAULT '0',
  `Cand` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`User`),
  UNIQUE KEY `User_UNIQUE` (`User`)
)

CREATE TABLE `elezione` (
  `Data_inizio` date DEFAULT NULL,
  `Data_fine` date DEFAULT NULL,
  `Descrizione` text,
  `Carica_Nome` varchar(45) NOT NULL,
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`ID`),
  KEY `fk_Elezione_Carica` (`Carica_Nome`),
  CONSTRAINT `fk_Elezione_Carica` FOREIGN KEY (`Carica_Nome`) REFERENCES `carica`
(`Nome`)
)

CREATE TABLE `carica` (
  `Nome` varchar(45) NOT NULL,
  `Durata` int(10) unsigned NOT NULL,
  `Descrizione` text,
  PRIMARY KEY (`Nome`)
)

CREATE TABLE `candidati` (
  `User` varchar(45) NOT NULL,
  `ID_Elezione` int(11) NOT NULL,
  `Voti_ricevuti` int(11) DEFAULT '0',
  PRIMARY KEY (`ID_Elezione`,`User`),
  KEY `User` (`User`),
  CONSTRAINT `candidati_ibfk_1` FOREIGN KEY (`User`) REFERENCES `utente` (`User`),
  CONSTRAINT `candidati_ibfk_2` FOREIGN KEY (`ID_Elezione`) REFERENCES `elezione`
(`ID`)
)

CREATE TABLE `voti` (
  `User` varchar(45) NOT NULL,
  `ID_Elezione` int(11) NOT NULL,
  PRIMARY KEY (`User`,`ID_Elezione`),
  KEY `ID_Elezione` (`ID_Elezione`),
  CONSTRAINT `voti_ibfk_1` FOREIGN KEY (`User`) REFERENCES `utente` (`User`),
  CONSTRAINT `voti_ibfk_2` FOREIGN KEY (`ID_Elezione`) REFERENCES `elezione` (`ID`)
)
```

Per facilitare le ricerche è stata creata una VIEW con tutte le cariche detenute.

```
CREATE VIEW `cariche_detenute` AS
   select
      `elezione`.`ID` AS `ID`,
      `elezione`.`Data_inizio` AS `Data_inizio`,
      `elezione`.`Data_fine` AS `Data_fine`,
      `candidati`.`Voti_ricevuti` AS `Voti_ricevuti`,
      `utente`.`User` AS `User`,
      `utente`.`Nome` AS `Nome`,
      `utente`.`Cognome` AS `Cognome`,
      `elezione`.`Carica_Nome` AS `Carica_Nome`,
      max(`candidati`.`Voti_ricevuti`) AS `Tot_voti`
   from
      (((`carica`
      join `elezione`)
      join `candidati`)
      join `utente`)
   where
      ((`carica`.`Nome` = `elezione`.`Carica_Nome`)
         and (`elezione`.`ID` = `candidati`.`ID_Elezione`)
         and (`candidati`.`User` = `utente`.`User`)
         and (`elezione`.`Data_fine` < curdate()))
   group by `elezione`.`ID`
```

Per facilitare alcune operazioni sono state definite le seguenti funzioni e procedure:

anno_minimo(): restituisce l'anno minimo presente nella tabella elezione

```
CREATE DEFINER=`root`@`localhost` FUNCTION `anno_minimo`() RETURNS int(11)
BEGIN
RETURN (SELECT MIN(YEAR(Data_inizio))
FROM elezione);
END ;;
```

candidatura_utente: candida un utente ad una elezione. Se la candidatura va a buon fine restituisce 0, altrimenti se l'utente ha già detenuto 2 volte la carica relativa all'elezione, restituisce -1

```
CREATE DEFINER=`root`@`localhost` FUNCTION `candidatura_utente`(usr VARCHAR(45),
id_elez INT(11)) RETURNS int(11)
BEGIN
        DECLARE volte INTEGER;
        SET volte = volte_carica_detenuta(usr, id_elez);
        IF volte > 1 THEN
                RETURN -1;
        ELSE
                INSERT INTO `democrazia`.`candidati`
(`User`,
`ID_Elezione`,
`Voti_ricevuti`)
```

```
VALUES
(
usr,
id_elez,
0
);
                RETURN 0;
        END IF;
        RETURN 0;
END ;;
```

nuova_carica: crea una nuova carica e restituisce 0, se le date sono sbagliate restituisce -1

```
CREATE DEFINER=`root`@`localhost` FUNCTION `nuova_carica`(nom VARCHAR(45), dur
INT(10),
 descr_c TEXT, iniz DATE, fine DATE, descr_e TEXT) RETURNS int(11)
BEGIN
        IF (iniz>fine OR CURDATE() >= iniz) THEN
                RETURN -1;
        END IF;
        INSERT INTO democrazia.carica
        (Nome, Durata, Descrizione)
        VALUES
        (nom, dur, descr_c);
        INSERT INTO democrazia.elezione
        (Data_inizio, Data_fine, Descrizione, Carica_Nome)
        VALUES
        (iniz, fine, descr_e, nom);

RETURN 0;
END ;;
```

nuovo_utente: crea un nuovo utente

```
CREATE DEFINER=`root`@`localhost` FUNCTION `nuovo_utente`(nom VARCHAR(45),
cognom VARCHAR(45),
 usr VARCHAR(45), pass VARCHAR(45), can TINYINT(1)) RETURNS int(11)
BEGIN

        INSERT INTO democrazia.utente
        (Nome, Cognome, User, Password, Banned, Cand)
        VALUES
        (nom, cognom, usr, pass, 1, can);

RETURN 0;
END ;;
```

rinnovo_automatico: per ogni carica scaduta, crea una nuova elezione per quella carica secondo dei parametri predefiniti (le elezioni iniziano un mese dopo la data attuale e finiscono due mesi dopo). La funzione restituisce il numero di nuove elezioni indette.

```
CREATE DEFINER=`root`@`localhost` FUNCTION `rinnovo_automatico`() RETURNS int(11)
BEGIN
	DECLARE times INT DEFAULT 0;
	DECLARE car VARCHAR(45);
	DECLARE done INT DEFAULT 0;
	DECLARE iniz DATE;
	DECLARE fin DATE;

	DECLARE cur CURSOR FOR (SELECT tab2.Carica_Nome
			FROM (SELECT Carica_Nome,MAX(Data_fine) AS datafine
			FROM elezione INNER JOIN carica
			ON elezione.Carica_Nome = carica.Nome
			GROUP BY Carica_Nome, Durata
			HAVING DATE_ADD(datafine, INTERVAL Durata YEAR) < CURDATE())
AS tab2);
	DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

	SET iniz = DATE_ADD(CURDATE(), INTERVAL 1 MONTH);
	SET fin  = DATE_ADD(CURDATE(), INTERVAL 2 MONTH);

	OPEN cur;
	read_loop: LOOP
	FETCH cur INTO car;
	IF done = 1 THEN
		LEAVE read_loop;
	END IF;
	SET times = times + 1;
	/* insert */
	INSERT INTO democrazia.elezione
	(Data_inizio, Data_fine, Carica_Nome)
	VALUES
	(iniz, fin, car);

	END LOOP;
	CLOSE cur;
	RETURN times;

END ;;
```

attiva_disattiva_utente: dato un utente, se esso è attivo, sarà disattivato, altrimenti verrà attivato.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `attiva_disattiva_utente`(usr
VARCHAR(45))
BEGIN
	UPDATE `democrazia`.`utente`
   SET Banned = 1 - Banned
   WHERE User = usr;
END ;;
```

vota: salva nel database il voto di un utente ad una data elezione. Se il voto non era nullo, il totale dei voti ricevuti dal candidato che si è votato sarà incrementato di 1.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `vota`(votante VARCHAR(45), elez
INT(11), candidato VARCHAR(45))
BEGIN
	INSERT INTO `democrazia`.`voti`
(`User`,
`ID_Elezione`)
VALUES
(
votante,
elez
);
IF candidato <> '' THEN
	UPDATE `democrazia`.`candidati`
  SET Voti_ricevuti = Voti_ricevuti + 1
  WHERE ID_Elezione = elez
  AND User = candidato;
END IF;

END ;;
```

STRUTTURA SITO PHP

index.php è la home page del sito.
header.php è l'intestazione del sito. Contiene il menu verticale da cui si possono accedere a tutte le altre pagine (Home, Elezioni in corso, Storico, Ricerca, Candidature)
footer.php è il footer del sito
login2.php contiene la form del login che comparirà a destra nel sito. Se l'utente è loggato, comparirà una form di logout.
admin_menu.php Se è loggato un utente amministratore, oltre alla form di login, apparirà anche un menu con le operazioni riservate agli amministratori (Attiva/Disattiva Utenti, Crea nuova carica, Rinnovo cariche)
start_session.php crea una nuova sessione o riprende la sessione in corso.
connessione.php si apre una connessione al database tramite le istruzioni PHP $link = mysql_connect('localhost', 'root', 'database'); e $db_selected = mysql_select_db('democrazia', $link);
controllo.php controlla che i dati provenienti dalla form di login siano corretti. Si effettua la seguente query $query="SELECT * FROM democrazia.utente where User='".$username."' and Password='".$password."'"; se l'utente è presente e attivo nel db, allora si aggiornano le seguenti variabili di sessione:
$_SESSION["log"]=1; per indicare che c'è un utente loggato
$_SESSION["usr"]=$username; per indicare l'username dell'utente
$_SESSION["admin"]=$riga["Admin"]; per indicare se l'utente è amministratore
$_SESSION["cand"]=$riga["Cand"]; per indicare se l'utente può candidarsi
controllo_sessione.php controlla che l'utente sia loggato
controllo_sessione_admin.php controlla che l'utente loggato sia un amministratore
forbidden.php nel caso i precedenti controlli falliscano, si viene rimandati in questa pagina che contiene un messaggio d'errore di accesso negato.
registrazione.php pagina con form per registrare un nuovo utente. I dati verranno poi mandati alla

pagina **registrazione2.php** che comunicherà se la creazione di un nuovo utente è andata a buon fine. L'utente appena creato per poter accedere ai servizi dovrà attendere l'attivazione da parte dell'amministratore.
crea_carica.php pagina che si può accedere solo dal menu amministratore, contiene una form per l'inserimento di una nuova carica e la relativa prima elezione. I dati verranno poi inviati alla pagina **crea_carica2.php** che ne comunicherà i risultati.
disattiva_utente.php pagina che si può accedere solo dal menu amministratore, contiene l'elenco di tutti gli utenti. L'amministratore può scegliere quali utenti attivare o disattivare. La chiamata alla procedura attiva_disattiva_utente è effettuata nel file **disattiva_utente2.php**.
rinnovo_automatico.php in questa pagina apparirà il numero di nuove elezioni indette quando l'amministratore ha attivato la funzione per il rinnovo automatico delle cariche.
elezioni.php Cliccando dal menu su 'Elezioni in corso', si accede a questa pagina che visualizza l'elenco delle elezioni in corso, tramite la query

```
SELECT elezione.*, COUNT(User) AS Tot_voti
FROM democrazia.elezione LEFT JOIN democrazia.voti
ON elezione.ID  = voti.ID_Elezione
WHERE Data_inizio < CURDATE() AND Data_fine > CURDATE()
GROUP BY ID;
```

Se l'utente è loggato potrà scegliere quale elezione volare, il link lo porterà alla pagina **vota.php** e nel caso l'utente non abbia già votato per questa elezione, comparirà l'elenco dei candidati che si possono votare e l'opzione scheda bianca. Espressa una preferenza, il voto sarà memorizzato tramite **vota2.php**.
storico.php pagina accessibile dal menu cliccando su 'Storico'. Viene visualizzato lo storico delle elezioni, tramite la query:

```
SELECT elezione.*, COUNT(User) AS Tot_voti
FROM democrazia.elezione LEFT JOIN democrazia.voti
ON elezione.ID  = voti.ID_Elezione
WHERE Data_fine < CURDATE()
GROUP BY ID
ORDER BY Data_fine DESC;
```

È possibile anche vedere la classifica di un'elezione cliccando sull'apposito link. Il link porterà alla pagina **classifica.php** che visualizzerà la classi+ dei candidati ordinata per voti ricevuti.
ricerca.php altra pagina accessibile dal menu. Contiene tutte le form necessarie per effettuare le ricerche richieste.
ricerca_candidato.php dato un candidato, si visualizzano tutte le elezioni a cui ha partecipato e i risultati ottenuti. Viene usata la seguente query:

```
$query="SELECT * FROM democrazia.candidati INNER JOIN democrazia.elezione
ON candidati.ID_Elezione = elezione.ID
WHERE candidati.User = \"".$cand."";
```

ricerca_carica.php data una carica, si visualizzano tutti gli utenti che l'hanno detenuta.

```
$query="SELECT * FROM cariche_detenute
WHERE Carica_Nome = \"".$carica."";
```

ricerca_periodo.php data un periodo, si visualizzano tutte le elezioni svoltesi in quel periodo.

```
$query="SELECT elezione.*, COUNT(User) AS Tot_voti
FROM democrazia.elezione LEFT JOIN democrazia.voti
ON elezione.ID  = voti.ID_Elezione
WHERE Data_fine < CURDATE()
AND Data_inizio BETWEEN ".$inizio." AND ".$fine."
OR Data_fine BETWEEN ".$inizio." AND ".$fine."
GROUP BY ID
```

```
   ORDER BY Data_fine DESC;";
```
candidatura.php mostra tutte le elezioni ancora non incominciate a cui ci si può candidare. Se un utente loggato è candidabile e soddisfa i primi due vincoli, allora potrà scegliere il link di quale elezione candidarsi. Il primo vincolo è verificare che l'utente non abbia una candidatura attiva

```
$query="SELECT *
 FROM candidati INNER JOIN elezione
 ON candidati.ID_Elezione = elezione.ID
 WHERE Data_fine > CURDATE()
 AND User = '".$_SESSION["usr"]."';";
```

Il secondo vincolo è verificare se l'utente non detiene già una carica

```
$query="SELECT *
FROM candidati, elezione, carica
WHERE candidati.ID_Elezione = elezione.ID
AND carica.Nome = elezione.Carica_Nome
AND DATE_ADD(Data_fine, INTERVAL Durata YEAR)  > CURDATE()
AND User = '".$_SESSION["usr"]."';";
```

Soddisfatti questi vincoli e scelta una elezione a cui candidarsi, il link rimanderà alla pagina **candidatura2.php** che controllerà anche il terzo vincolo e dirà se la candidatura è andata a buon fine.