

ANALISI DEL PROGETTO

Struttura logica del database

Il nucleo centrale del servizio realizzato risiede all'interno del database. Esso infatti contiene tutti i dati relativi agli studenti, al questionario, e alle risposte date da ciascuno studente.

Il modello scelto per la realizzazione è di tipo relazionale. Si procede ad illustrare la sua struttura logica, attraverso l'utilizzo di un diagramma Entity-Relationship:

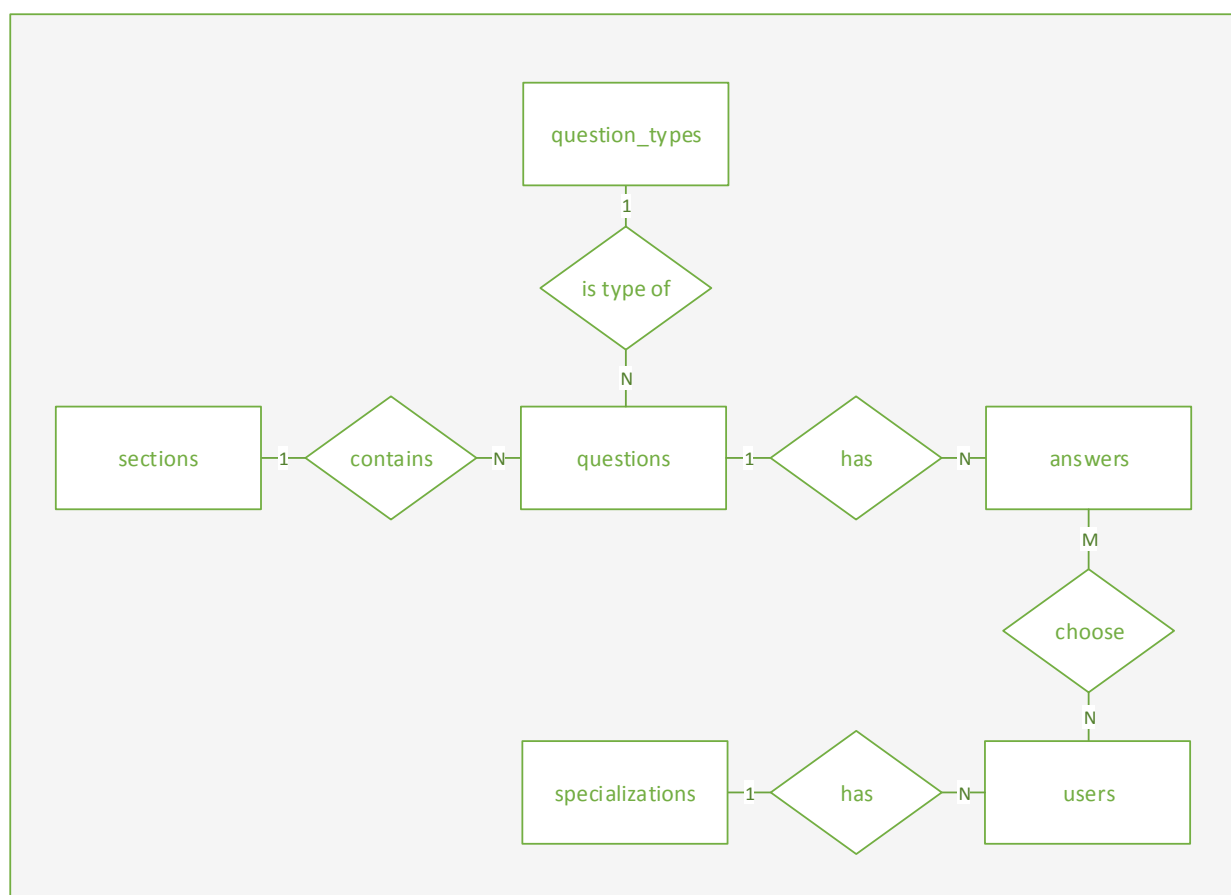


Figura 1 – Diagramma Entity-Relationship del modello scelto

ANALISI ENTITÀ

Le entità identificate nella modellazione del database sono le seguenti:

- **sections** – contiene un elenco delle sezioni (gruppi di domande relative ad un argomento comune)
- **questions** – contiene un elenco delle domande presenti all'interno del questionario
- **question_types** – contiene un elenco delle varie tipologie di domande presentate
- **answers** – contiene un elenco di tutte le possibili risposte per ciascuna domanda
- **users** – contiene un elenco degli studenti a cui verrà presentato il dizionario
- **specializations** – contiene un elenco delle singole specializzazioni di cui gli studenti hanno fatto parte

ANALISI RELAZIONI TRA LE ENTITÀ

Le relazioni presenti tra le entità identificate, sono le seguenti:

- sections - questions (1/N) – ciascuna sezione contiene una o più domande
- question_types - questions (1/N) – ciascuna domanda appartiene ad una delle tipologie stabilite
- questions - answers (1/N) – a ciascuna domanda sono associate un certo numero di risposte
- answers - users (M/N) – un singolo studente può scegliere più risposte all'interno del questionario, ma allo stesso modo una singola risposta può essere scelta da più studenti distinti
- specializations - users (1/N) – ciascuno studente è stato iscritto ad una determinata specializzazione durante il suo corso di studi

Struttura fisica del database

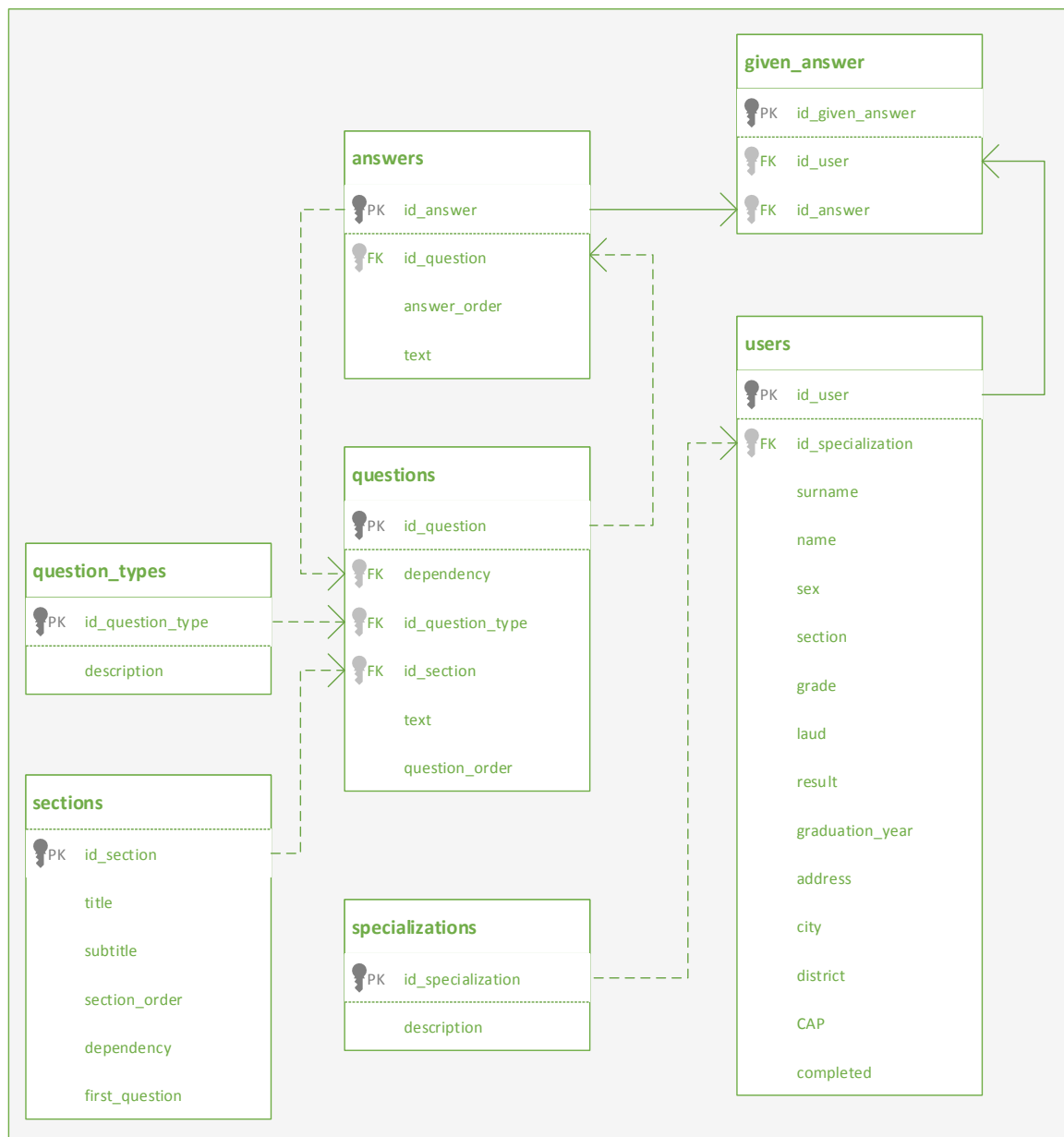


Figura 2 – Diagramma UML della struttura fisica del database

Si illustra in figura 2 la struttura fisica scelta per l'implementazione della base di dati, realizzata mediante l'RDBMS (Relational Database Management System) MySQL.

LA TABELLA GIVEN_ANSWERS

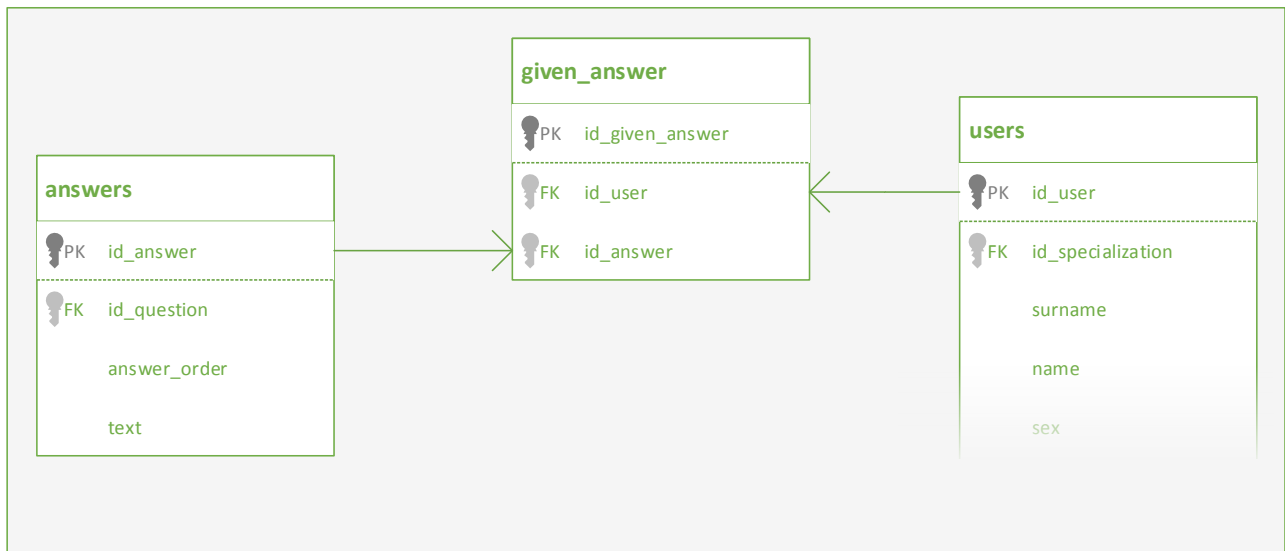


Figura 3 – Particolare del diagramma UML della struttura del database, che mostra la tabella given_answer

Un limite dei database relazionali consiste nella loro impossibilità di realizzare relazioni many-to-many (M:N), come quella presente tra le entità **users** e **answers**. Tale limite è però facilmente superabile mediante l'introduzione di una tabella di giunzione (junction table), e a due relazioni di tipo one-to-many (1:N) dalle due tabelle coinvolte a quest'ultima.

La junction table realizzata assume il nome **given_answer**, e contiene un elenco delle risposte date da ciascun utente. E' opportuno notare come tecnica permetta la semplice risoluzione di problemi altrimenti ostici, come la scelta di risposte multiple per una singola domanda (è sufficiente inserire più entry, una relativa ad ogni risposta).

FLESSIBILITÀ DELL MODELLO UTILIZZATO

Una delle maggiori sfide presenti all'interno del progetto è la realizzazione di una base di dati flessibile, capace di adattarsi in modo rapido e semplice ad eventuali cambiamenti. La struttura proposta segue tale principio, in quanto ogni singola informazione relativa al questionario è compresa all'interno del database.

Aggiungere una nuova domanda, cambiarne la tipologia o l'ordine, correggere il testo di una risposta: sono tutte operazioni effettuabili mediante semplici modifiche all'interno del database. Non è richiesta alcun tipo di modifica al codice dell'applicazione.

Questo permette di ridurre gran parte dei costi di manutenzione, e limita l'eventualità di compromettere il sistema nel caso siano necessarie modifiche.

PROBLEMATICHE DELLE DIPENDENZE DOMANDA – RISPOSTA

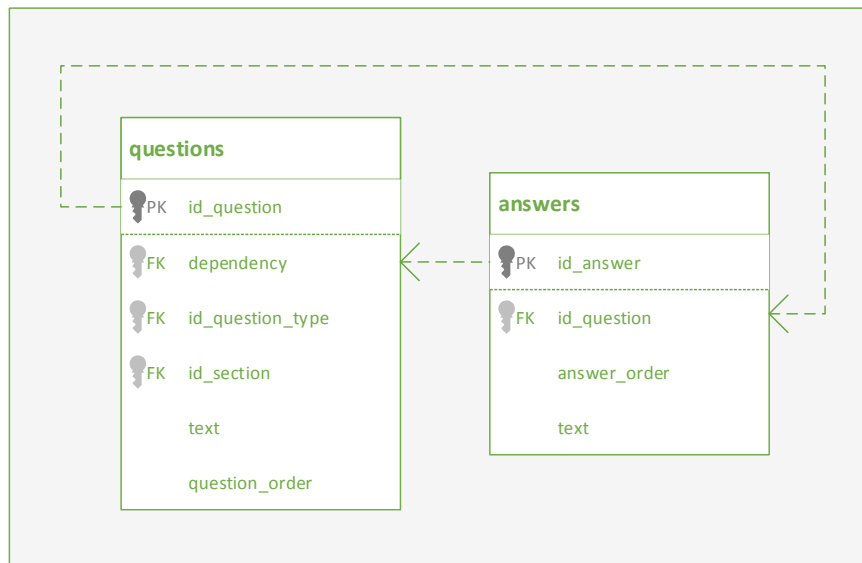


Figura 4 – Particolare del diagramma UML della struttura del database, che mostra le tabelle **questions** e **answers**

Nell'ottica di una completa separazione tra dati e applicativo, risulta necessario introdurre un meccanismo per modellare le dipendenze presenti tra determinate domande ed altre risposte. Si consideri, per esempio, come ad uno studente che ha dichiarato di non aver mai frequentato l'università, non sia necessario effettuare domande relative al corso di laurea scelto.

La soluzione scelta consiste nell'introduzione di un campo opzionale **dependency** (dipendenza) all'interno della tabella **questions**. Se la risposta identificata da tale campo è stata selezionata in precedenza la relativa domanda viene mostrata. Se tale risposta non è stata selezionata, la domanda invece non viene mostrata.

NORMALIZZAZIONE

Nell'ottica dell'eliminazione di ridondanze informative, e conseguentemente del rischio introdurre uno stato di incoerenza all'interno del database, sono stati seguiti una serie di accorgimenti volti alla riduzione dello stesso alla terza forma normale.

La normalizzazione consiste di fatto nella riduzione di tabelle che presentano campi interdipendenti in tabelle più piccole. Nel caso sia necessario ottenere la tabella originaria per effettuare un'interrogazione è possibile utilizzare un'operazione di join.

Tecnologie utilizzate

Si effettua adesso una rassegna delle tecnologie impiegate. In una successiva sezione si avrà modo di analizzare nello specifico l'utilizzo di ciascuna tecnologia all'interno del progetto Outlook.

PHP, MYSQL E MYSQLI

Una tecnologia fondamentale ai fini della realizzazione dell'applicativo lato server è PHP.

PHP, acronimo di "PHP: Hypertext Preprocessor", è un linguaggio di programmazione interpretato nato per la programmazione web. Al giorno d'oggi è largamente utilizzato per sviluppare applicazioni *lato server*.

All'interno dell'applicativo PHP (nello specifico la versione 5.2.17) viene utilizzato per la creazione di pagine dinamiche, come quella di presentazione del questionario, e per la realizzazione dei meccanismi di login e di invio delle risposte.

L'interfaccia di collegamento tra l'interprete PHP e l'RDBMS MySQL è fornita dalla libreria MySQLi (MySQL Improved). Tale libreria consiste essenzialmente in un miglioramento della precedente libreria MySQL: permette un approccio di tipo orientato agli oggetti (mantenendo intatta la possibilità di sfruttare un approccio procedurale), introduce l'utilizzo dei prepared statements (fondamentali per eliminare il rischio di attacchi di tipo SQL Injection) e permette la realizzazione di tecniche quali le transazioni.

HTML5 E CSS3

La realizzazione del sito web collegato al servizio Outlook, è stata realizzata mediante l'utilizzo dei linguaggi di markup HTML5 e CSS3.

HTML5 (Hyper Text Transfer Protocol) è comunemente utilizzato per la realizzazione di pagine web.

CSS3 (Cascading Style Sheets) è usato per definire la formattazione e lo stile di documenti HTML.

L'utilizzo di tali tecnologie permette una completa separazione tra il significato semantico del contenuto e l'aspetto grafico, relativo al modo in cui i contenuti sono presentati. I vantaggi sono una maggiore chiarezza e la possibilità di riutilizzare parti di codice.

JAVASCRIPT E JQUERY

La natura dinamica del sito web realizzato, è possibile grazie all'utilizzo di tecnologie quali il Javascript, e di una sua potente libreria, jQuery.

JavaScript è un linguaggio di scripting orientato agli oggetti comunemente usato nella creazione di siti web, lato client-side. Tutti i moderni browser web, infatti, dispongono di un interprete Javascript.

jQuery è una potente libreria Javascript che si propone lo scopo di semplificare la programmazione lato client delle pagine HTML. Per farlo, fornisce una serie di strumenti che permettono di effettuare svariate operazioni con poche e semplici linee di codice.

L'utilizzo di queste tecnologie ha trovato uso, specificatamente, nella realizzazione del meccanismo di visualizzazione di domande dipendenti da determinate risposte.

JSON

JSON, acronimo di JavaScript Object Notation, è un formato adatto per lo scambio dei dati in applicazioni client-server.

All'interno del progetto Outlook tale formato trova uso nello scambio di informazioni relative alle domande da visualizzare tra l'interprete Javascript eseguito sul browser dell'utente ed il server PHP che ospita l'applicazione.

La scelta di tale formato è stata effettuata in seguito a considerazioni relative alla facilità d'uso ed il supporto nativo nei linguaggi Javascript e PHP.

Considerazioni sulla sicurezza

Ogni applicativo, prima di essere rilasciato, deve necessariamente affrontare una fase di revisione e di considerazioni su quanto riguarda la sicurezza e l'affidabilità dello stesso. Si esamineranno adesso alcuni aspetti fondamentali; in una sezione successiva si vedrà in che modo è stata eseguita l'implementazione vera e propria.

PREPARED STATEMENTS

Applicazioni web, che devono necessariamente interfacciarsi con un grande numero di persone, sono esposti ad una serie di attacchi che minano l'integrità dei dati memorizzati o dei servizi stessi. Una tecnica di attacco tra le più semplici da realizzare, ma non per questo meno pericolosa, è quella delle SQL Injection.

È possibile applicare tale tecnica in tutti i casi in cui dati inseriti dall'utente vengono utilizzati per costruire dinamicamente una query SQL. Dei dati opportunamente costruiti, infatti, forniscono al malintenzionato la capacità di leggere e modificare i dati presenti nel database.

I prepared statement forniscono una protezione verso questo tipo di attacchi. Le fasi del loro utilizzo si possono riassumere nei seguenti passi:

1. Creazione del prepared statement: ogni statement coincide con una query, priva dei valori forniti dall'utente. L'RDBMS inoltre lo ottimizza in modo da rendere efficienti più esecuzioni consecutive.
2. Binding dei parametri del prepared statement: i valori forniti dall'utente vengono collegati al prepared statement. In questa fase subiscono controlli per eliminare il rischio di SQL Injection.
3. Esecuzione del prepared statement e fetch dei risultati.

TRANSAZIONI

In un database accessibile contemporaneamente da più utenti, come quello utilizzato da questo applicativo, assume grande importanza l'utilizzo delle transazioni per il mantenimento dell'integrità dei dati memorizzati. Una transazione consiste in una sequenza di operazioni eseguite in modo atomico, che comporta il passaggio del database da uno stato iniziale consistente S_i ad uno stato consistente S_n .

Le transazioni hanno due principali funzioni:

1. Isolamento dei processi di accesso e di modifica eseguiti da più utenti in modo concorrente: se non si garantisce l'esecuzione di ciascuna transazione in modo atomico, i risultati ottenuti possono essere inesatti.
2. Mantenimento del database in uno stato di coerenza: se anche una sola delle operazioni contenute in una transazione fallisce, il database viene riportato allo stato iniziale S_i .

È possibile riassumere le quattro proprietà fondamentali che devono soddisfare i DBMS che implementano le transazioni, perché queste operino in modo corretto sui dati, mediante l'acronimo ACID (Atomicity, Consistency, Isolation, Durability).

IL SITO WEB

Il sito web rappresenta il vero e proprio centro dell'applicazione. Attraverso di esso, infatti, gli ex-studenti sono in grado di compilare il questionario.

Collegandosi al sito del servizio Outlook viene presentata una pagina d'introduzione. In seguito si accede alla pagina di login. Come indicato in precedenza a ciascun ex-studente è stato fornito una password che lo identifica, e permette l'accesso al servizio.

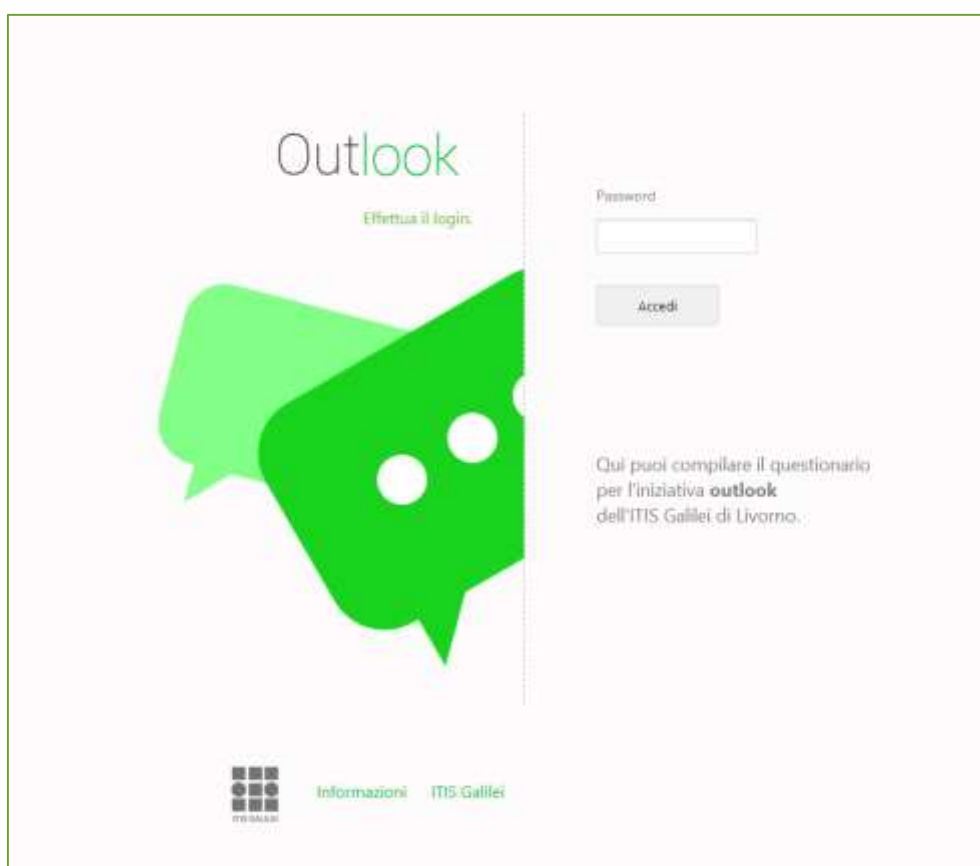


Figura 5 – Pagina di login del sito web

Inserendo una password valida viene presentata la schermata contenente le domande del questionario. Nella parte superiore sono mostrati alcuni dati relativi all'ex-studente che sta compilando il questionario, mentre nella inferiore è presente il questionario vero e proprio. In fondo alla pagina sono presenti i crediti, dei collegamenti al sito della scuola ed un link che può essere utilizzato per effettuare il logout.

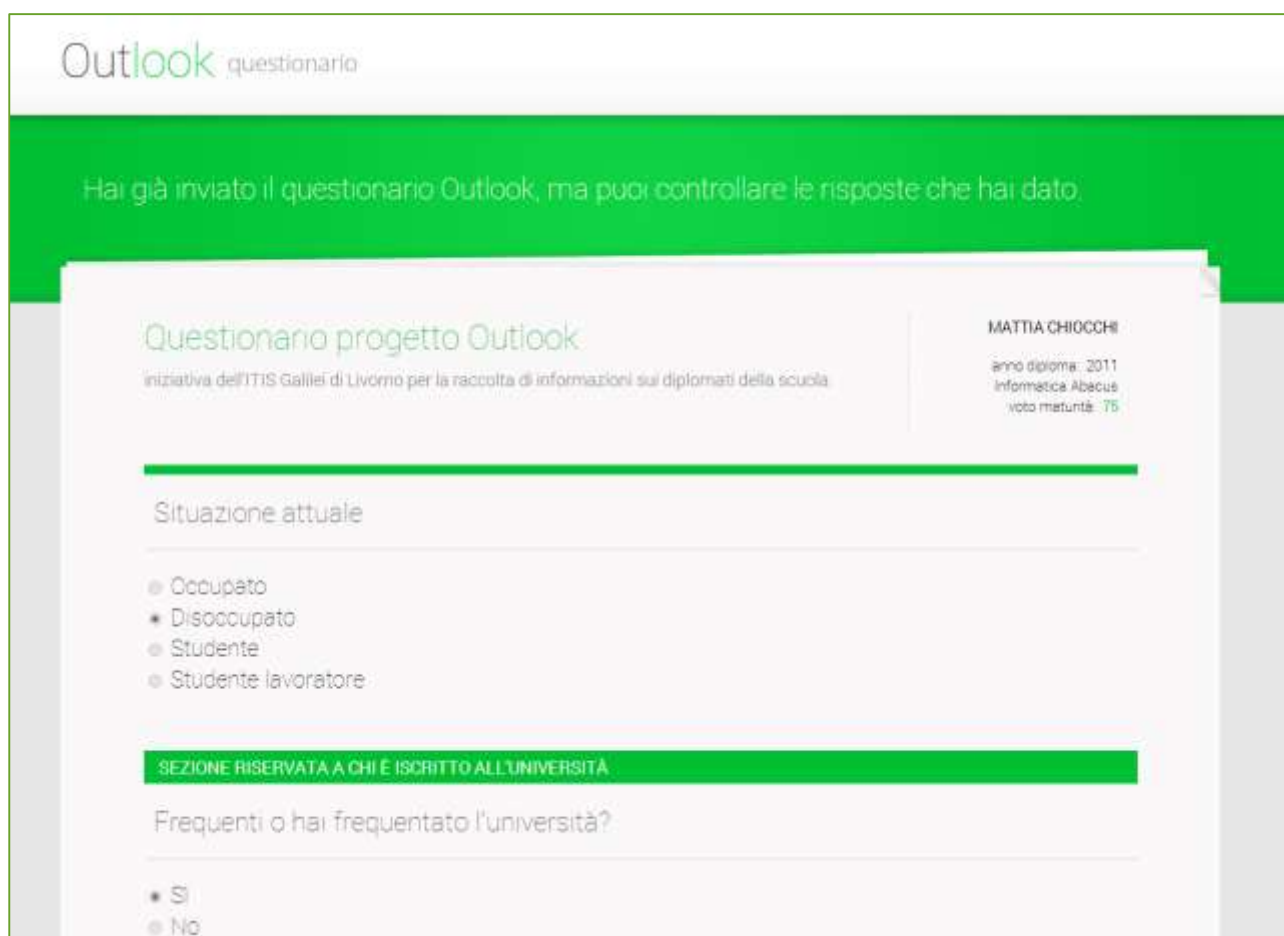
Figura 6 – Pagina del sito web contenente il questionario

Il questionario è composto da varie sezioni, e parte delle domande vengono mostrate (o nascoste) in base alle risposte date alle domande precedenti. Si procede a titolo illustrativo a mostrare cosa accade quando si risponde positivamente alla domanda "Frequenti o hai frequentato l'università?".

Figura 7 – Particolare di domanda condizionata da una risposta

Terminata la compilazione del questionario è possibile effettuare l'invio delle risposte date. In seguito alla pressione del pulsante relativo (posizionato nella parte alta della pagina) viene effettuato un controllo lato-client per verificare che ogni domanda abbia effettivamente ottenuto una risposta. In seguito le risposte vengono inviate al server, che provvede a memorizzarle.

Se tutto è andato a buon fine viene presentata una pagina di avvenuto completamento, tramite la quale è possibile raggiungere il sito della scuola o tornare al questionario. Se l'utente ha già completato l'inserimento, infatti, vengono mostrate le risposte inserite in precedenza, e non risulta possibile inviare nuovamente il questionario.



The screenshot shows a web page titled "Outlook questionario". A green banner at the top states: "Hai già inviato il questionario Outlook, ma puoi controllare le risposte che hai dato." Below this, the main content area is titled "Questionario progetto Outlook" and includes the subtitle "iniziativa dell'ITIS Galilei di Livorno per la raccolta di informazioni sui diplomati della scuola". On the right, user information is displayed: "MATTIA CHIOCCHI", "anno diploma: 2011", "informatica Abacus", and "voto maturità: 75". The main section is titled "Situazione attuale" and contains a list of radio buttons: "Occupato", "Disoccupato" (which is selected), "Studente", and "Studente lavoratore". Below this is a green bar with the text "SEZIONE RISERVATA A CHI È ISCRITTO ALL'UNIVERSITÀ". Underneath, the question "Frequenti o hai frequentato l'università?" is followed by "Si" (selected) and "No" radio buttons.

Figura 8 – Pagina del questionario dopo averlo inviato

Parti salienti di codice PHP

Si procede ad illustrare alcune parti di codice lato-server relative all'applicazione realizzata, focalizzandosi sugli aspetti ritenuti più interessanti.

MECCANISMO DI LOGIN – PREPARED STATEMENTS

In relazione a quanto accennato in precedenza, durante la realizzazione del progetto è stata realizzata una piccola libreria contenente una serie di funzioni atte all'accesso in modo semplice e standardizzato alla base di dati.

In particolare è stata realizzata una funzione `exec_query_many_results` che permette l'utilizzo di prepared statement in modo semplice. Tale funzione deve accettare quindi un numero arbitrario di parametri, e allo stesso modo restituire un numero arbitrario di valori. Per ottenere ciò la funzione contiene numerose istruzioni non rilevanti ai fini dell'analisi dell'implementazione di prepared statement.

Si procede quindi ad illustrare un codice esemplificativo di utilizzo degli stessi per effettuare il meccanismo di autenticazione, ricordando che tale codice non è presente all'interno del progetto realizzato. Per informazioni su come ottenere il codice sorgente del progetto si rimanda all'appendice.

```
1. // creazione connessione
2. $db = new mysqli($dbLocation, $dbUser, $dbPassword, $dbName);
3.
4. // controllo connessione
5. if (mysqli_connect_errno()) {
6.     printf("Connect failed: %s\n", mysqli_connect_error());
7.     exit();
8. }
9.
10. // creazione prepared statement
11. if ($stmt = $db->prepare("SELECT id_user FROM users WHERE id_user = ?")) {
12.
13.     // binding dei parametri
14.     // $password contiene la password fornita dall'utente
15.     $stmt->bind_param("s", $password);
16.
17.     // esecuzione query
18.     $stmt->execute();
19.
20.     // bind dei risultati
21.     $stmt->bind_result($result);
22.
23.     // fetch dei valori
24.     $stmt->fetch();
25.
26.     // close statement
27.     $stmt->close();
28. }
29.
30. // chiusura connessione
31. $db->close();
32.
33. if($result)
34. {
35.     // login effettuato con successo
36. }
37. else
38. {
39.     // login non valido
40. }
```

Esempio di codice 1 – Utilizzo di prepared statement per l'autenticazione

INVIO DEL QUESTIONARIO – TRANSAZIONI

Per garantire l'integrità e la coerenza della base di dati in seguito alla fase di inserimento di nuovi record relativi ad un questionario, è stato sfruttato il meccanismo delle transazioni. Tale tecnica permette di eseguire le operazioni di inserimento (ne è richiesta una per ciascuna risposta data dall'utente) in modo atomico. Nel caso si verifichi un errore tutte le operazioni di inserimento associate all'utente, effettuate in precedenza, vengono annullate.

La libreria MySQLi permette l'implementazione di tale meccanismo mediante l'utilizzo di due semplici funzioni:

1. `$db->commit()`, che esegue un'operazione di consolidamento del nuovo stato della base di dati dopo le modifiche effettuate.
2. `$db->rollback()`, che annulla le modifiche effettuate al database e lo ripristina al suo stato iniziale, precedente all'inizio della transazione.

Si ricorda inoltre che per effettuare manualmente un controllo sulle transazioni è necessario chiamare la funzione `$db->autocommit(FALSE)` per disattivare il commit automatico delle operazioni.

Il funzionamento di tale tecnica è osservabile all'interno del codice responsabile del salvataggio delle risposte date dall'utente. Il codice dimostra inoltre l'utilizzo di alcune funzioni presenti all'interno della libreria realizzata.

```
1. require_once('lib/common.php');
2.
3. if(user_is_not_logged_in()) // utente non autenticato
4.     header('location:login.php') || die();
5.
6. if(user_has_completed_the_survey()) // questionario già inviato
7.     header('location:questions.php') || die();
8.
9. if(!isset($_POST)) // nessuna risposta fornita
10.     header('location:questions.php');
11.
12. $id_user = get_user_id(); //ottengo codice utente
13.
14. // ...
15. // omessa costruzione $answers_list, contenente le risposte date
16.
17. // filtraggio risposte invalide
18. $n = count($answers_list);
19.
20. $question_marks_string = build_question_marks_string($n);
21.
22. $query = "SELECT answers.id_answer
23.           FROM answers, questions
24.           WHERE answers.id_question = questions.id_question
25.                 AND answers.id_answer IN ({ $question_marks_string })
26.                 AND ( questions.dependency IS NULL
27.                       OR questions.dependency IN ({ $question_marks_string }) )";
28. $types = str_repeat('i', $n * 2);
29. $args = array_merge(array($query, $types), $answers_list, $answers_list);
30. $result = call_user_func_array('exec_query_many_results', $args);
31.
32. // inizio inserimento
33. disable_autocommit();
34.
```

```

35. $success = True;
36. foreach($result as $row)
37. {
38.     $answer = $row->id_answer;
39.
40.     $query = 'INSERT INTO given_answers
41.             (id_given_answer, id_user, id_answer)
42.             VALUES
43.             (DEFAULT, ?, ?)';
44.     $result = exec_query($query, 'ii', $id_user, $answer);
45.
46.     if($result === FALSE)
47.         $success = FALSE;
48. }
49.
50. // salvataggio completamento questionario
51. $query = 'UPDATE users
52.         SET completed=1
53.         WHERE id_user=?';
54. $result = exec_query($query, 'i', $id_user);
55.
56. if($result === FALSE)
57.     $success = FALSE;
58.
59. $newlocation = "";
60. if($success)
61. {
62.     // nessun errore avvenuto, effettuo il commit
63.     commit();
64.     $newlocation = 'completed.php';
65. }
66. else
67. {
68.     // si è verificato un errore, effettuo il rollback
69.     rollback();
70.     $newlocation = 'questions.php';
71. }
72.
73.
74. enable_autocommit(); // riabilito l'autocommit
75.
76. header("Location:$newlocation") || die(); // effettuo redirect

```

Esempio di codice 2 – Codice sorgente della pagina di invio delle risposte

Parti salienti di codice Javascript

Il funzionamento del progetto Outlook dipende, però, anche da una serie di istruzioni lato-client. La presenza di tale codice permette di ridurre il carico del server, affidando parte delle all'interprete Javascript presente nel browser dell'utente.

Il codice necessario per realizzare quanto voluto risulta tuttavia contenuto. Questo è possibile grazie all'utilizzo della popolare libreria jQuery.

VISTA DINAMICA DELLE DOMANDE

All'interno del questionario sono presenti alcune domande, che dovrebbero essere visualizzate solo nel caso in cui determinate risposte siano state date. Si ricorda, a titolo esemplificativo, il caso in cui ad un utente che non ha frequentato l'università non debba essere chiesta la facoltà frequentata.

La soluzione adottata consiste nel creare in un primo momento (tramite uno script PHP) una pagina contenente tutte le domande presenti nel questionario. A questo punto entra in funzione uno script Javascript, che, dopo aver inviato al server un elenco delle risposte date, ottiene un elenco contenente gli identificativi delle domande da mostrare. Infine viene eseguito un ciclo sugli elementi relativi ad ogni domanda che viene mostrata, o nascosta, in base a quanto ricevuto dal server.

Questa operazione viene eseguita in seguito al cambiamento di stato di una domanda con “effetti collaterali” (le cui risposte possono influenzare la visibilità di altre domande). Il passaggio delle risposte attualmente selezionate, dal client al server, viene effettuato mediante richiesta GET alla pagina `fetch_questions_list.php`. Tale pagina ritorna l'elenco degli identificativi delle domande da visualizzare, in codifica JSON.

Si illustra il codice relativo.

```
1. function fetch_questions()
2. {
3.     // costruzione elenco risposte attualmente selezionate
4.     var s = $('[data-side-effects=true]:checked,'
5.         + '[data-side-effects=true] option:selected').map(function() {
6.         return this.value? this.value : null;
7.     }).get().join();
8.
9.     // esecuzione get
10.    $.get(
11.        'fetch_questions_list.php',
12.        {q: s},
13.        // funzione di callback
14.        function(data) {
15.            // parse JSON dell'elenco delle domande da mostrare
16.            var questions_to_show = JSON.parse(data);
17.            questions_to_show = $.map(questions_to_show,
18.                function(id) { return id.toString() });
19.            $('.question').each(function() {
20.                var to_show = $.inArray($(this).attr('data-question_id'),
21.                    questions_to_show) >= 0;
22.                var visible = $(this).is(':visible');
23.
24.                if(to_show && (!visible))
25.                {
26.                    // se la domanda va mostrata ed è nascosta la mostro e abilito
27.                    $(this).slideDown('slow');
28.                    $(this).find('input:not([data-disabled=true]),'+
29.                        'select:not([data-disabled=true])').removeAttr('disabled');
30.                }
31.                else if(!to_show && visible)
32.                {
33.                    // se la domanda non va mostrata ed è visibile la nascondo e disabilito
34.                    $(this).slideUp('slow');
35.                    $(this).find('input,select').attr('disabled', 'disabled');
36.                }
37.            });
38.
39.            $('body').css('cursor', 'auto');
40.        }
41.    );
42.
43.    $('body').css('cursor', 'progress');
44. }
```