

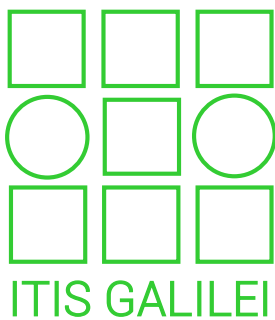


Progetto

Outlook

Tesina di Maturità

Daniele Del Giudice
5A INA A.S. 2012/2013



INDICE

Il progetto Outlook.....	1
Descrizione del progetto	1
Come si è svolta la realizzazione.....	2
Strumenti	2
Microsoft Access.....	2
Sublime Text	2
XAMPP	2
Git e GitHub.....	3
Convenzioni utilizzate	3
Utilizzo della lingua inglese.....	3
Creazione di una libreria comune	3
Analisi del progetto	4
Struttura logica del database.....	4
Analisi entità.....	4
Analisi relazioni tra le entità.....	5
Struttura fisica del database	6
La tabella given_answers.....	7
Flessibilità dell modello utilizzato	7
Problematiche delle dipendenze domanda – risposta.....	8
Normalizzazione.....	8
Tecnologie utilizzate	9
PHP, MySQL e MySQLi	9
HTML5 e CSS3.....	9
Javascript e jQuery	10
JSON	10
Considerazioni sulla sicurezza	10
Prepared Statements.....	10
Transazioni	11

Il sito web	12
Parti salienti di codice PHP	14
Meccanismo di login – Prepared statements	15
Invio del questionario – transazioni.....	16
Parti salienti di codice Javascript	17
Vista dinamica delle domande	18
Considerazioni finali	20
Sviluppi futuri	20
Appendice	22
Codice sorgente del progetto Outlook	22
Elenco abbreviazioni utilizzate	22
Elenco immagini.....	22
Elenco esempi di codice.....	23
Bibliografia	23
Sitografia.....	23

IL PROGETTO OUTLOOK

Il progetto Outlook consiste nella realizzazione di un sito web per la gestione di un questionario on-line, da proporre ai diplomati del nostro istituto degli ultimi dieci anni, per la costruzione di un quadro informativo circa i loro sbocchi occupazionali o di studio post-diploma. Il questionario mira tra l'altro ad indagare quanto il percorso formativo fruito durante il percorso scolastico nel nostro Istituto influisca sulla situazione successiva.

Descrizione del progetto

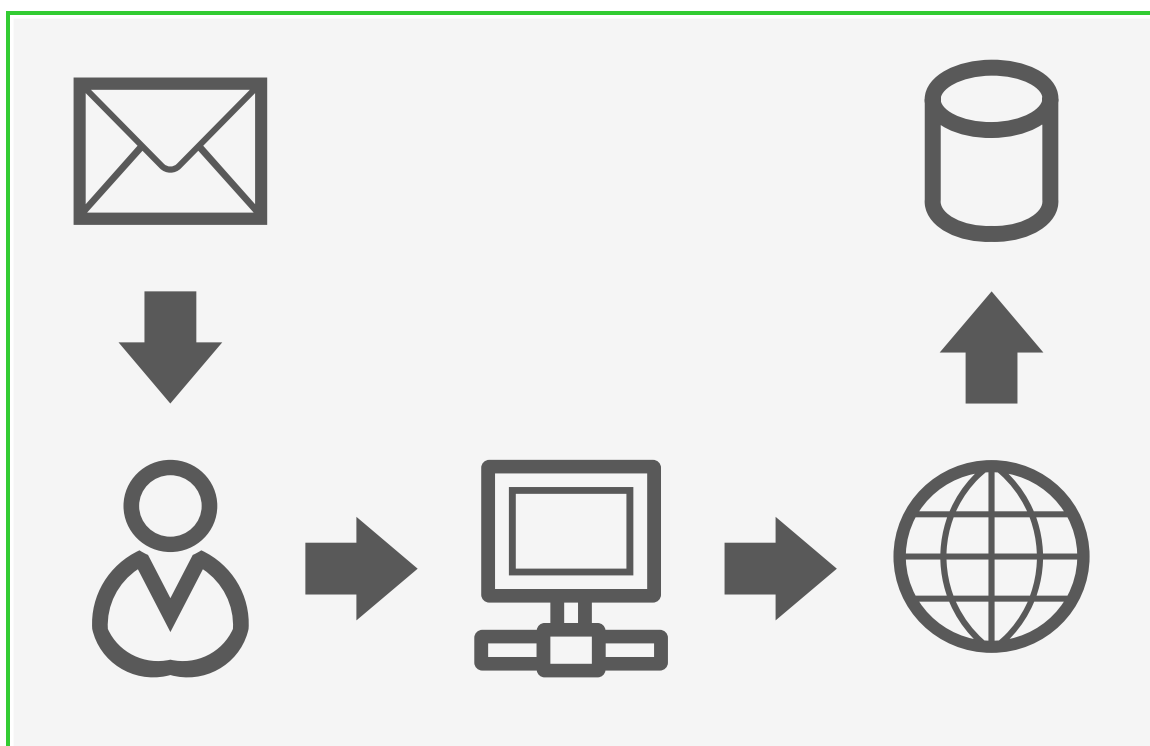


Figura 1 – Schema riassuntivo del funzionamento del progetto Outlook

La visione del progetto Outlook, da parte di ciascun diplomato, inizia con la ricezione di una lettera. All'interno sono presenti un codice identificativo (paragonabile ad una password) ed un invito a collegarsi al sito web della scuola per compilare il questionario. L'invio delle lettere è gestito attraverso il servizio di mail-merge offerto da Poste Italiane.

In seguito, l'utente effettua l'autenticazione inserendo la password ricevuta, ed è così in grado di compilare il questionario. I dati relativi vengono infine memorizzati su un database interno alla scuola, dove potranno in seguito subire una fase di elaborazione e analisi.

COME SI È SVOLTA LA REALIZZAZIONE

Strumenti

Durante la realizzazione del progetto, sono stati utilizzati un insieme di strumenti che hanno permesso e/o agevolato il compimento dello stesso. Per completezza, si analizzano quelli ritenuti più importanti.

MICROSOFT ACCESS

Microsoft Access è un software per la gestione di basi di dati di tipo relazionale (RDBMS), sviluppato da Microsoft Corporation.

All'interno del progetto, Microsoft Access ha trovato applicazione durante la fase iniziale. È stato infatti utilizzato per effettuare una serie di operazioni sull'archivio di dati fornito dall'Amministrazione Scolastica (eliminazione di informazioni non rilevanti ai fini del progetto, o conversione in un formato standard di determinate informazioni).

SUBLIME TEXT

Sublime Text è un editor di testo per codice, markup e prosa.

Sublime Text è l'editor di testo scelto per la realizzazione del progetto. È stato scelto per la sua semplicità d'uso e per il grande numero di funzionalità che mette a disposizione.

XAMPP

XAMPP è una distribuzione Apache facile da installare contenente MySQL, PHP e Perl.

La scelta dell'utilizzo di XAMPP durante l'attuazione del progetto Outlook è stata motivata dalla mancanza di un server reale nella fase di sviluppo. Tale distribuzione di applicativi, installabile su qualsiasi computer, permette infatti la realizzazione di server web locale. Ciò ha permesso di semplificare in modo sensibile la creazione dell'applicativo.

GIT E GITHUB

Git è un sistema di controllo versione gratuito ed open source, progettato per gestire progetti di qualsiasi dimensione con velocità ed efficienza.

La decisione di utilizzare un sistema di controllo versione è stata motivata dai tangibili vantaggi che il suo utilizzo porta. Essi permettono infatti di effettuare liberamente modifiche al codice, senza preoccuparsi di “perdere” parte di esso o “rompere” qualche funzionalità. In particolare, la scelta di usare Git è stata motivata dalla facilità d’impiego e dalla presenza di un servizio web noto come GitHub, che semplifica il caricamento e la condivisione di codice controllato da Git.

Convenzioni utilizzate

Durante la realizzazione dell’applicativo, ci si è avvalsi inoltre di una serie di convenzioni, utilizzate per rendere più sistematica e pratica la fase di sviluppo. Come in precedenza si elencano quelle ritenute più significative.

UTILIZZO DELLA LINGUA INGLESE

In ogni parte del progetto Outlook si è scelto di utilizzare la lingua inglese. Ciò è stato motivato dall’ormai indiscussa affermazione di tale lingua come lo standard de-facto per lo sviluppo di software. Quasi la totalità dei linguaggi di programmazione, di documentazioni e di codice, infatti, effettuano questa scelta. L’utilizzo della lingua inglese permette inoltre di evitare determinate ambiguità che potrebbero verificarsi utilizzando la lingua italiana.

CREAZIONE DI UNA LIBRERIA COMUNE

Durante la fase di sviluppo si è scelto di operare creando una serie di funzioni comuni, utilizzate all’interno dell’intero applicativo. Ciò, seppur comporti inizialmente uno certo dispendio di tempo, permette di velocizzare notevolmente l’implementazione di nuove funzionalità e la manutenzione di quelle già presenti. In particolare sono stati realizzati funzioni responsabili dell’accesso e dell’interazione con la base di dati.

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 al questionario, agli ex-studenti e alle risposte date da ciascuno di essi.

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:

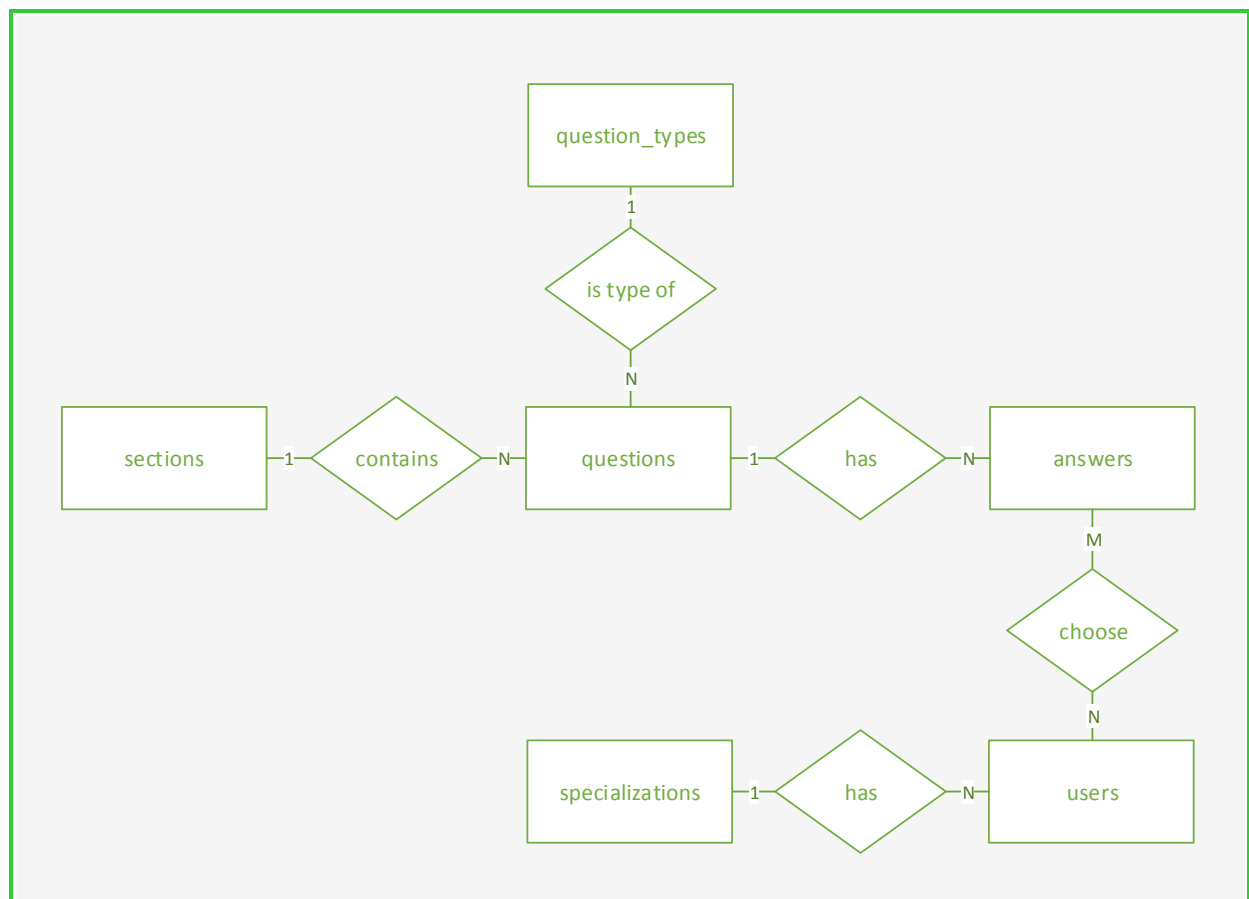


Figura 2 – Diagramma Entity-Relationship del modello scelto

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 ex-studenti a cui verrà presentato il questionario
- **specializations** – contiene un elenco delle specializzazioni a cui gli ex-studenti erano iscritti

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 ex-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)** – ciascun ex-studente è stato iscritto ad una determinata specializzazione durante il suo corso di studi

Struttura fisica del database

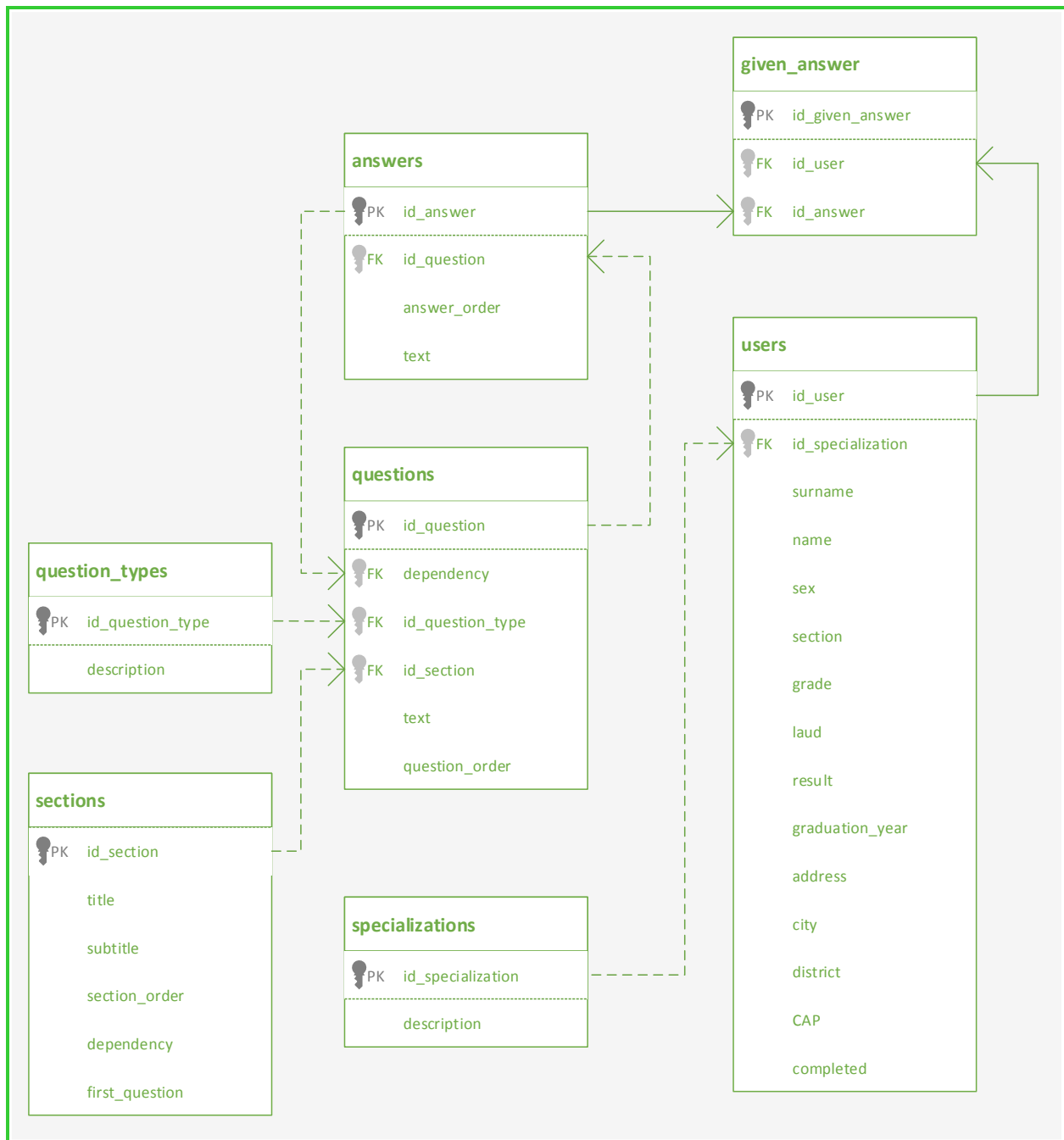


Figura 3 – Diagramma UML della struttura fisica del database

Si illustra in figura 3 la struttura fisica scelta per l'implementazione della base di dati, realizzata mediante l'RDBMS (Relational Database Management System) MySQL.

LA TABELLA GIVEN_ANSWERS

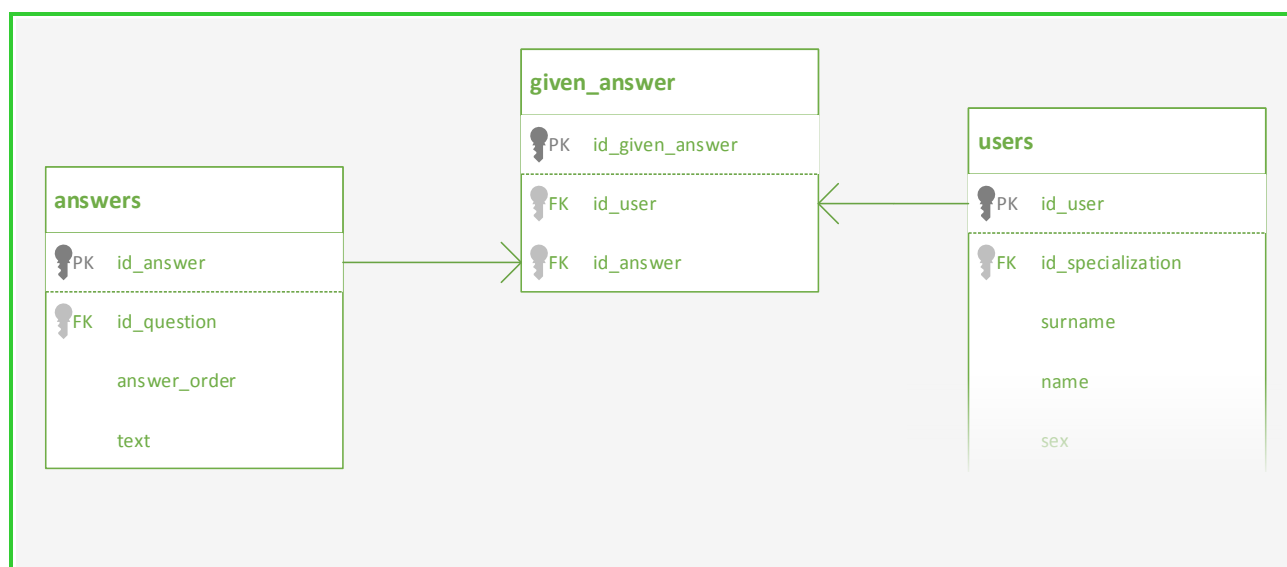


Figura 4 – Particolare della diagramma UML, tabella given_answer

Un limite dei database relazionali consiste nella loro impossibilità di rappresentare relazioni *multi-a-multi* (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* (note anche come *junction table*), e a due relazioni di tipo *uno-a-molti* (1:N), dalle due tabelle coinvolte a quest'ultima.

La tabella di giunzione realizzata assume il nome **given_answer**, e contiene un elenco delle risposte date da ciascun utente. E' opportuno notare come tale 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 Outlook 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

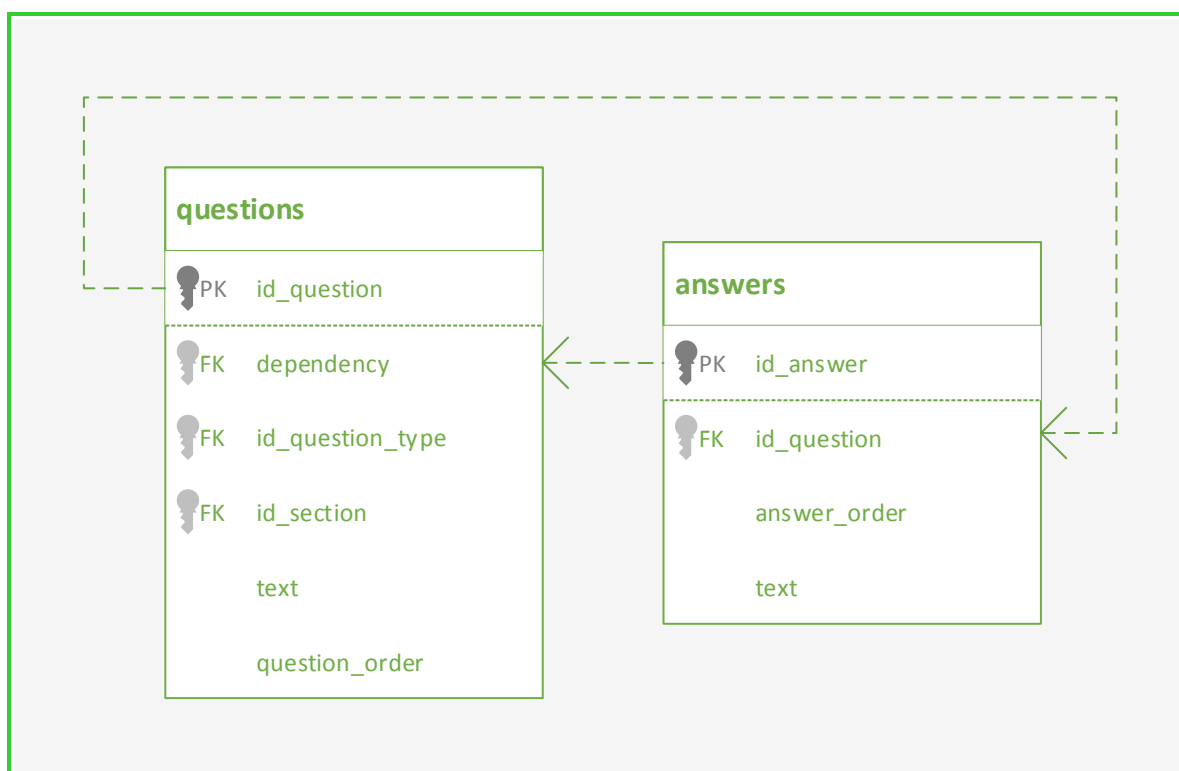


Figura 5 – Particolare della diagramma UML, tabelle questions e answers

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 di 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 statement* (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 utilizzato per la realizzazione di pagine web.

CSS3 (*Cascading Style Sheets*) è definisce 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. Tutti i moderni browser web, infatti, dispongono di un interprete Javascript.

jQuery è una veloce e 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 date.

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 il formato **JSON** 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, 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.

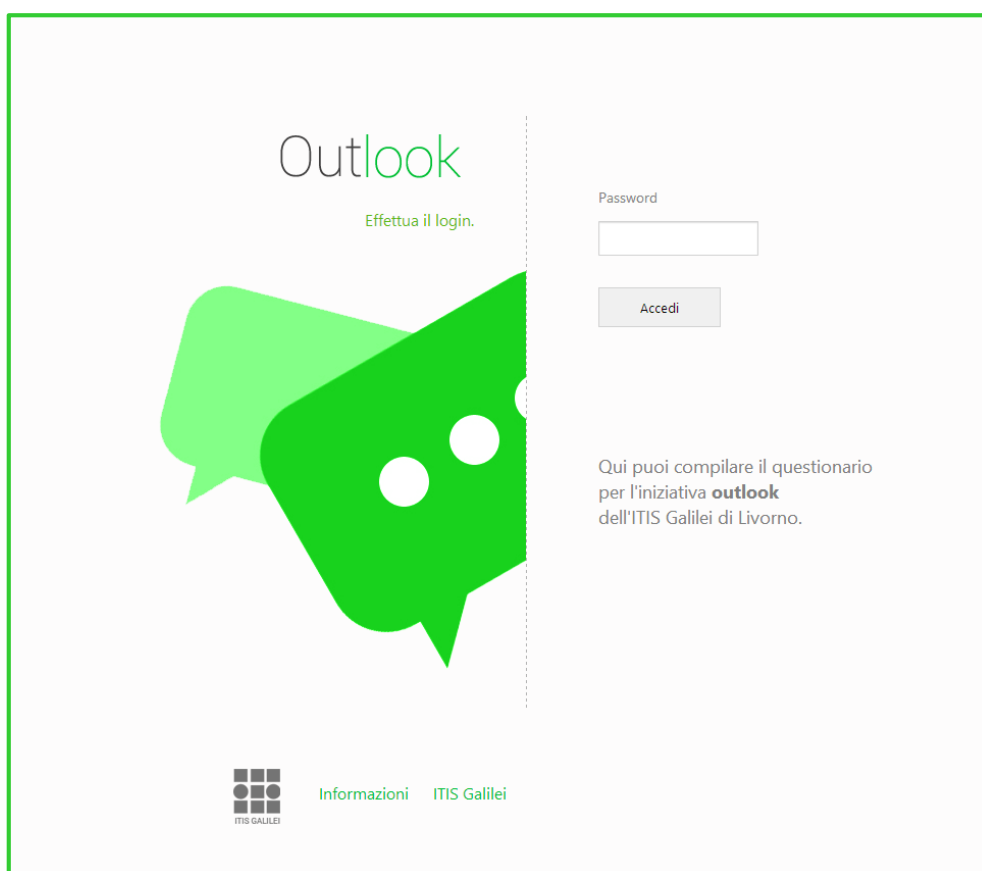


Figura 6 – Pagina di autenticazione

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.

Outlook questionario

Invia questionario

Ci siamo! Compila il questionario Outlook, e clicca su "Invia questionario" quando hai finito.

Questionario progetto Outlook
iniziativa dell'ITIS Galilei di Livorno per la raccolta di informazioni sui diplomati della scuola.

MATTIA CHIOCCHI
anno diploma: 2011
Informatica Abacus
voto maturità: 75

Situazione attuale

☐ Occupato
☐ Disoccupato
☐ Studente
☐ Studente lavoratore

SEZIONE RISERVATA A CHI È ISCRITTO ALL'UNIVERSITÀ

Frequenti o hai frequentato l'università?

Figura 7 – Pagina del 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?".

SEZIONE RISERVATA A CHI È ISCRITTO ALL'UNIVERSITÀ

Frequenti o hai frequentato l'università?

☐ Sì
☐ No

SEZIONE RISERVATA A CHI LAVORA O HA LAVORATO

Lavori o hai lavorato?

☐ Sì
☐ No

SEZIONE RISERVATA A CHI È IN CERCA DI LAVORO

Sei in cerca di lavoro?

SEZIONE RISERVATA A CHI È ISCRITTO ALL'UNIVERSITÀ

Frequenti o hai frequentato l'università?

☒ Sì
☐ No

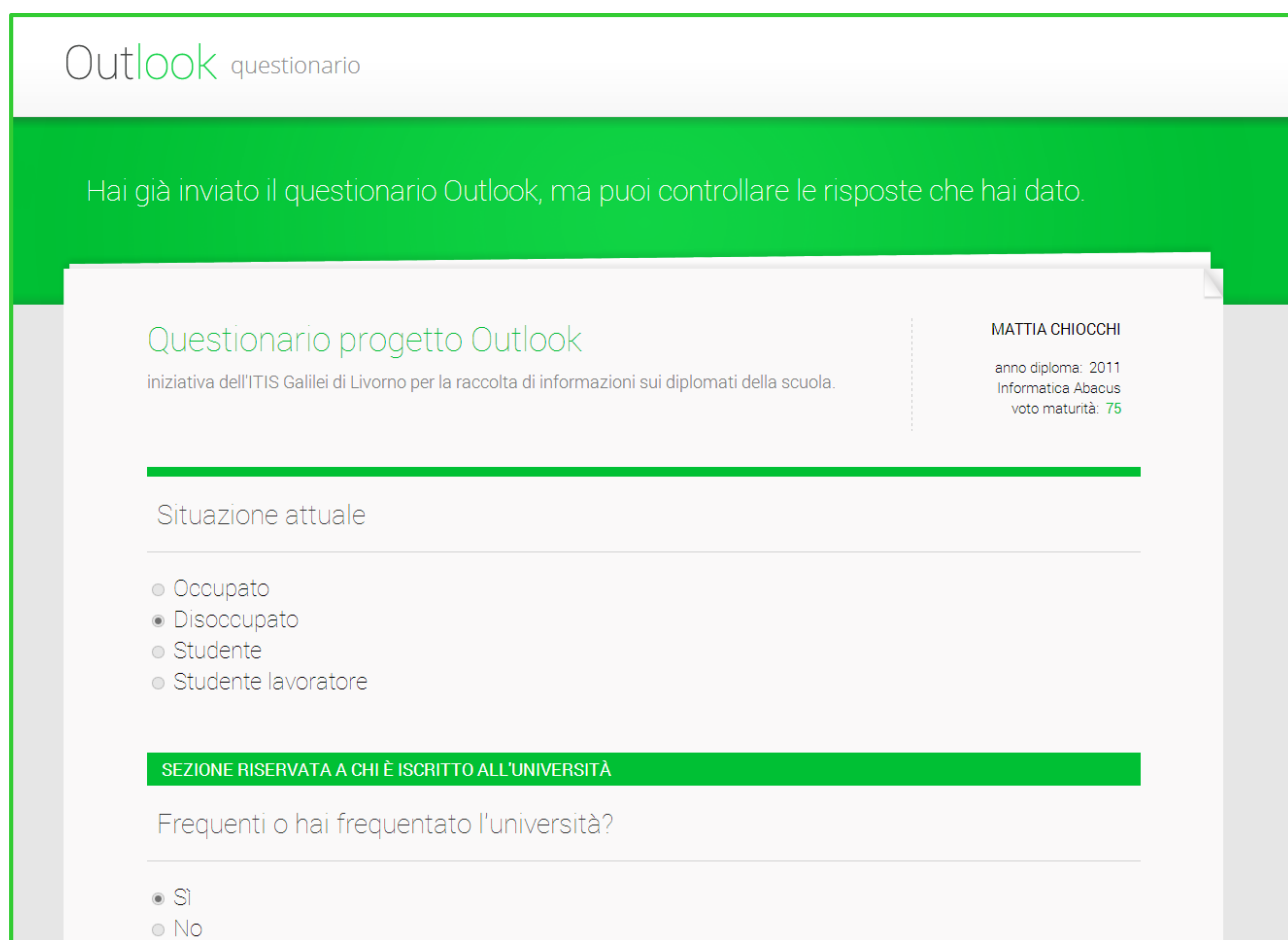
Sede universitaria

Settore facoltà

Figura 8 – Visualizzazione dinamica delle domande

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 interface for the 'Outlook' questionnaire. At the top, a green banner contains the text 'Hai già inviato il questionario Outlook, ma puoi controllare le risposte che hai dato.' Below this, a white card displays the title 'Questionario progetto Outlook' and its purpose: 'iniziativa dell'ITIS Galilei di Livorno per la raccolta di informazioni sui diplomati della scuola.' The user's name, 'MATTIA CHIOCCHI', is shown in the top right, along with their diploma year (2011), subject (Informatica Abacus), and score (75). The main section, 'Situazione attuale', features a list of radio buttons for 'Occupato', 'Disoccupato' (selected), 'Studente', and 'Studente lavoratore'. A green bar indicates a section reserved for university graduates. Below this, the question 'Frequenti o hai frequentato l'università?' is followed by 'Si' (selected) and 'No' radio buttons.

Figura 9 – Come appare la pagina del questionario dopo averlo compilato

Parti salienti di codice PHP

Si procede all'illustrazione di 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 Outlook 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 dell'applicativo 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 comprendente 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.         {
7.             return this.value? this.value : null;
8.         }).get().join();
9.
10.    // esecuzione get
11.    $.get(
12.        'fetch_questions_list.php',
13.
14.        // seleziono parametri inviati con la richiesta get
15.        {q: s},
16.
17.        // funzione di callback
18.        function(data)
19.        {
20.            // parse JSON dell'elenco delle domande da mostrare
21.            var questions_to_show = JSON.parse(data);
22.            questions_to_show = $.map(questions_to_show,
23.                function(id) { return id.toString() });
24.
25.            // per ciascuna domanda
26.            $('[.question]').each(function()
27.            {
28.                // controllo se va mostrata e se è già visibile
29.                var to_show = $.inArray($(this).attr('data-question_id'),
30.                    questions_to_show) >= 0;
31.                var visible = $(this).is(':visible');
32.

```

```

33.         if(to_show && (!visible))
34.         {
35.             // se la domanda va mostrata ed è nascosta la mostro e abilito
36.             $(this).slideDown('slow');
37.             $(this).find('input:not([data-disabled=true]),'+
38.                 'select:not([data-disabled=true])').removeAttr('disabled');
39.         }
40.         else if((!to_show) && visible)
41.         {
42.             // se la domanda non va mostrata ed è visibile la nascondo e disabilito
43.             $(this).slideUp('slow');
44.             $(this).find('input,select').attr('disabled', 'disabled');
45.         }
46.     });
47.
48.     // caricamento delle domande da mostrare terminato
49.     // reimposto il cursore allo stato di default
50.     $('body').css('cursor', 'auto');
51. }
52. );
53.
54. // imposto il cursore come "in progress"
55. $('body').css('cursor', 'progress');
56. }

```

Esempio di codice 3 – Aggiornamento dinamico delle domande da visualizzare

CONSIDERAZIONI FINALI

L'esperienza di realizzazione del progetto Outlook ha avuto una particolare valenza formativa in quanto ha simulato la "commissione" di un determinato prodotto da parte di un cliente. Durante la sua creazione sono state necessarie conoscenze ad ampio spettro, spaziando dalla programmazione lato server a quella lato client.

Durante la realizzazione, infatti, sono stati affrontati i seguenti punti:

1. Definire e gestire la struttura del database di supporto all'applicazione
2. Acquisire e normalizzare alcuni insiemi di dati da inserire preventivamente nel database (dati degli ex-studenti a cui viene rivolto il questionario, struttura del questionario stesso, ecc.)
3. Curare la grafica dell'interfaccia web ed i meccanismi di sicurezza per l'acquisizione dei dati del sondaggio e privacy degli utenti esterni che li forniscono
4. Allestire il server destinato alla gestione del servizio
5. Definire l'iter relativo all'invito degli utenti esterni per la compilazione del questionario
6. Definire le metodologie per la gestione dei dati rilevati, in funzione di una successiva analisi

Un aspetto molto importante è stato inoltre la possibilità di affrontare implicazioni di tipo tecnico ed organizzative collegate alla realizzazione di un applicativo di questa tipologia.

Sviluppi futuri

L'elaborazione dei dati rilevati, fase conclusiva del progetto che avverrà in seguito al termine della rilevazione, fornirà informazioni che consentiranno di effettuare interessanti considerazioni relative all'offerta formativa del nostro istituto. In previsione di tale fase è stata realizzata una pagina che, grazie alle funzionalità offerte dalle API **Google Chart**, permette di effettuare in modo immediato alcune considerazioni iniziali circa i dati raccolti.

Le API Google Chart permettono agli sviluppatori di applicazioni web la creazione di grafici, e il loro inserimento all'interno di pagine web.

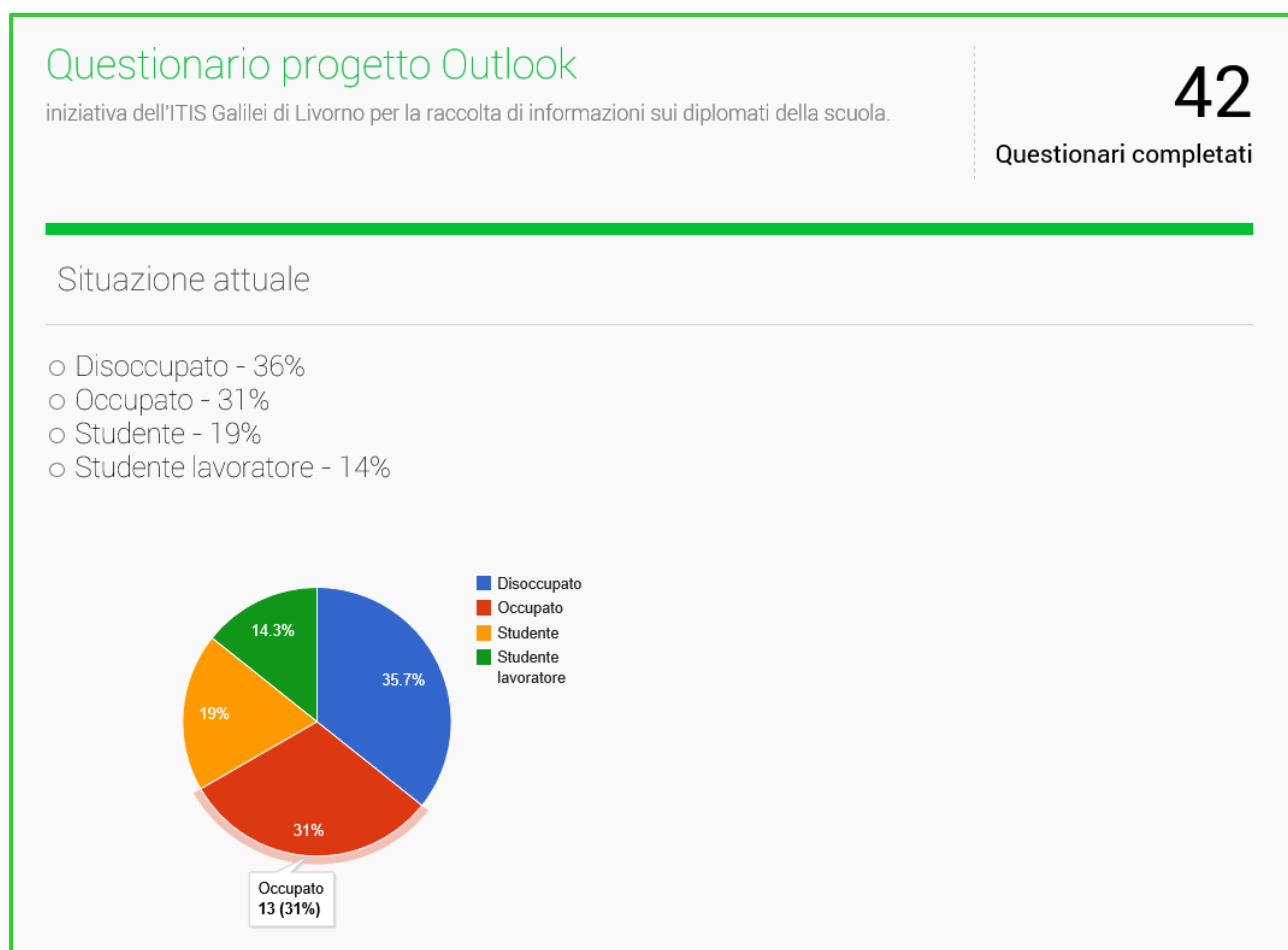


Figura 10 – Possibile pagina di presentazione dei risultati

La struttura della pagina presentata è simile a quella utilizzata per la visualizzazione del questionario. Ciò permette di riutilizzare in modo efficace i meccanismi presenti in essa, nell'ottica di una più accurata selezione dei risultati presentati.

Si precisa inoltre che, per ovvi motivi, i dati visualizzati in figura sono derivanti da inserimenti arbitrari effettuati ai fini di dimostrare il funzionamento della pagina stessa.

APPENDICE

Codice sorgente del progetto Outlook

L'intero progetto Outlook è rilasciato sotto MIT License, è possibile scaricare il codice sorgente:

- Tramite Git: <https://github.com/D4n13le/outlook.git>
- Tramite link diretto: <https://github.com/D4n13le/outlook/archive/master.zip>

Elenco abbreviazioni utilizzate

ACID – Atomicity, Consistency, Isolation, e Durability

API – Application Programming Interface

CSS – Cascading Style Sheets

DBMS – Database Management System

HTML – HyperText Markup Language

JSON – JavaScript Object Notation

MySQLi – MySQL Improved

PHP – PHP: Hypertext Preprocessor

RDBMS – Relational Database Management System

XAMPP – Cross-platform Apache MySQL PHP and Perl

Elenco immagini

Figura 1 – Schema riassuntivo del funzionamento del progetto Outlook	1
Figura 2 – Diagramma Entity-Relationship del modello scelto	4
Figura 3 – Diagramma UML della struttura fisica del database	6
Figura 4 – Particolare della diagramma UML, tabella given_answer	7
Figura 5 – Particolare della diagramma UML, tabelle questions e answers	8
Figura 6 – Pagina di autenticazione	12
Figura 7 – Pagina del questionario	13
Figura 8 – Visualizzazione dinamica delle domande	13
Figura 9 – Come appare la pagina del questionario dopo averlo compilato	14
Figura 10 – Possibile pagina di presentazione dei risultati	21

Elenco esempi di codice

Esempio di codice 1 – Utilizzo di prepared statement per l'autenticazione	15
Esempio di codice 2 – Codice sorgente della pagina di invio delle risposte	17
Esempio di codice 3 – Aggiornamento dinamico delle domande da visualizzare	19

Bibliografia

- I. Bigatti, Picca, Noel Castro – Sistemi di gestione di basi di dati e SQL – Apogeo – 2007
- II. Formichi, Meini – Corso di Informatica Volume 3 – Zanichelli, 2012
- III. Formichi, Meini – Il Linguaggio PHP – Zanichelli – 2012
- IV. Gigliotti – HTML 4.0.1 – Apogeo – 2004
- V. Venuti – JavaScript dalle basi ad AJAX – Edizioni Fag – 2008

Sitografia

- I. Apache Friends - XAMPP – <http://www.apachefriends.org/it/xampp.html> – 06/2013
- II. Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification – <http://www.w3.org/TR/CSS21/> – 06/2013
- III. Git – <http://git-scm.com/> – 06/2013
- IV. HTML5, a vocabulary and associated APIs for HTML and XHTML – <http://www.w3.org/TR/html5/> – 06/2013
- V. jQuery – <http://jquery.com/> – 06/2013
- VI. JSON – <http://www.json.org/> – 06/2013
- VII. Software e applicazioni di database - Access - Office.com - <http://office.microsoft.com/it-it/access/> - 06/2013
- VIII. Sublime Text: The text editor you'll fall in love with – <http://www.sublimetext.com/> – 06/2013