

manuale tecnico “myexam”

Marco Rizzi (Web developer)

Potito Aghilar (Web developer)

Giuseppe Caporizzi (Project manager)

14 Maggio 2019

Versione 1.0

Indice

[1. Configurazione del sistema 2](#_Toc8908224)

[2. Inizializzazione sistema (con importazione delle domande) 3](#_Toc8908225)

[3. Funzionamento del sistema 3](#_Toc8908226)

[4. Pannello di amministrazione 4](#_Toc8908227)

[5. *Back end* 4](#_Toc8908228)

[*6.* *Front end* 7](#_Toc8908229)

[7. JavaScript e jQuery 12](#_Toc8908230)

**MANUALE TECNICO “MyExam”**

1. Configurazione del sistema

Affinché il sistema possa essere utilizzato, è necessario impostare i parametri di configurazione e la base di dati.

Connessione al database

Affinché “MyExam” possa funzionare è necessario che disponga di un database.

Per questo motivo, è stato definito il file **database.php** memorizzato in **/www/api/** che contiene i parametri di connessione.

I principali parametri sono:

* **host**: indica l’indirizzo ip del database (generalmente localhost)
* **port**: indica la porta del database (generalmente 3306)
* **username**: indica il nome utente dell’utente del database
* **password**: indica la password dell’utente del database
* **database**: indica il database da utilizzare



Creazione Database

Attraverso la funzione di importazione del DBMS, è necessario importare il file **database.sql** contenuto in **/setup/** che, consente di creare il database denominato **esame** con la relativa struttura tabellare.

Configurazione indirizzo IP amministratore

Affinché l’amministratore possa accedere al pannello di amministrazione, è necessario inserire all’interno del file **/www/admin/security.php** l’indirizzo ip pubblico dell’amministratore che, per motivi di sicurezza dovrà essere diverso da quello degli esaminandi.

Inserimento titolo esame

Nella cartella **/www/api/requires/**, è possibile trovare il file **config.php** in cui ci sono i seguenti parametri:

* **Questions**: permette di definire il numero delle domande;
* **TimeQuestion**: permette di definire il numero di secondi per ciascuna domanda;
* **MaxTime**: restituisce il tempo totale del quiz;
* **ExamTitle**: permette di definire il nome dell’esame.

Caricamento domande

Affinché sia possibile importare le domande, con le relative risposte, è necessario inserire nella cartella denominata **/setup/** il file **questions.xml**.

Il file dovrà avere la seguente struttura:

<?xml version="1.0" encoding="UTF-8"?>

<questions>

<question>

<text>Testo Domanda 1</text>

<answer correct=”1”>Risposta 1</answer>

<answer>Risposta 2</answer>

<answer>Risposta 3</answer>

<answer>Risposta 4</answer>

<answer>Risposta 5</answer>

</question>

….

….

<question>

<text>Testo Domanda N</text>

<answer>Risposta 1</answer>

<answer>Risposta 2</answer>

<answer>Risposta 3</answer>

<answer>Risposta 4</answer>

<answer correct=”1”>Risposta 5</answer>

</question>

</questions>

1. Inizializzazione del sistema con importazione del file recante le domande

Digitando il seguente indirizzo sarà possibile inizializzare il sistema:

<http://myexamplatform.altervista.org/admin/install.php>

L’inizializzazione sarà eseguita soltanto se all’interno della cartella **/setup/** sarà presente il file denominato **questions.xml**.

Al termine dell’inizializzazione, il file sarà eliminato.

Con l’esecuzionedella procedura di inizializzazione si cancellerà il contenuto di tutte le tabelle con relativa perdita dei dati.

Con l’eliminazione del file **questions.xml**, l’installer interpreterà il sistema come già inizializzato e dunque l’inizializzazione non andrà a buon fine.

1. Funzionamento del sistema

Il sistema è costituito da una logica basata su un *front end* ed un *back end*.

Il *front end* fornisce una interfaccia all’utente, il *back end* presiede alla logica di funzionamento.

L’interazione tra *back end* e *front end* avviene secondo uno schema di funzionamento simile a quello delle API. In particolare, i dati sono restituiti attraverso un formato di tipo JSON (JavaScript Object Notation) il quale ha caratteristica fondamentale di risultare semplice da gestire sia da parte dell’utente che dell’elaboratore.

1. Pannello di amministrazione

Per implementare il pannello di amministrazione, si sono utilizzati i seguenti files:

* **index.php**: è l’interfaccia principale che permette all’amministratore di visualizzare l’elenco degli studenti, gestire i punteggi bonus/malus e visualizzare in dettaglio le prove di ogni singolo studente;
* **studente.php**: è l’interfaccia che permette la stampa della copia dell’esame dello studente selezionato dall’interfaccia principale e gestire i punti bonus/malus;
* **lista.php**: permette di stampare gli elaborati di tutti gli studenti;
* **updatePoints.php**: è un file di supporto che consente l’aggiornamento dei punteggi bonus/malus degli studenti;
* **requires.php**: è un file di supporto che permette l’importazione dei parametri di configurazione in ciascun script.php;
* **head.php**: è un file di supporto che consente l’importazione degli stili e delle librerie (file css e file js);
* **security.php**: è un file di supporto che permette di gestire gli accessi all’interfaccia dell’esaminatore;
* **install.php**: consente l’inizializzazione del sistema (con relativa importazione delle domande).

1. *Back end*

I files del *back end*, implementano la logica di funzionamento, sono localizzati all’interno della directory **/www/api/**, e sono:

* **examTitle.php**: permette di restituire al *client* il nome dell’esame;
* **login.php**: permette al *client* di eseguire l’autenticazione dell’utente;
* **publish.php**: permette di memorizzare le risposte date dagli utenti;
* **questions.php**: fornisce al *client* l’elenco delle domande;
* **submit.php**: permette al *client* di fornire l’interfaccia di chiusura dell’esame da parte dell’utente mediante la pressione del pulsante “termina esame”;
* **time.php**: fornisce al client il tempo residuo;
* **forcedSubmit.php**: permette la chiusura della sessione in modo forzato allo scadere del tempo.

file examTitle.php

Il file al suo interno contiene il seguente codice scritto in php:

L’istruzione **require ‘requires.php’;** permette di importare i parametri di configurazione del sistema.

L’istruzione **echo json\_encode([‘title’] => $examTitle,]);** permette di restituire al client il file JSON che restituisce al *front end* il nome dell’esame.

file login.php

Il file login.php, come detto precedentemente, permette all’utente di eseguire l’autenticazione.  
Prima di procedere con questa fase, lo script ha il compito di verificare se il client ha inviato dei campi validi e, per farlo, si utilizzano le seguenti istruzioni:

Validata la correttezza dei dati, viene eseguita una query che permette di verificare se l’utente si è già autenticato in precedenza.

Pertanto, possono verificarsi due scenari:

1. la query non restituisce alcun dato: il sistema memorizzerà i dati dell’utente nella tabella users;
2. la query restituisce un risultato: il sistema verifica lo stato dell’utente.

Nello scenario b. possono verificarsi i seguenti sotto scenari:

1. l’utente non ha terminato l’esame: il sistema restituirà l’elenco delle risposte date e il tempo residuo;
2. l’utente ha terminato l’esame: il sistema restituirà un messaggio di errore.

file forcedSubmit.php

Lo script ha il compito di terminare la sessione in modo forzato nel momento in cui il tempo scade.

La corretta esecuzione dello script php restituisce al client l’esito: studente idoneo o non idoneo.

file submit.php

Ha la stessa funzione dello script precedente terminando la sessione nel momento in cui l’utente chiede di chiuderla causando due possibili scenari:

1. lo studente non ha risposto a tutte le domande: lo script non permetterà la chiusura della sessione restituendo il numero delle risposte non date;
2. lo studente ha risposto a tutte le domande: la sessione sarà terminata restituendo al client l’esito “idoneo” o “non idoneo”.

Nel momento in cui la chiusura della sessione viene accettata, il server memorizza il *timestamp* impedendo il recupero della sessione.

file publish.php

Lo script php ha il compito di memorizzare le risposte degli utenti.

In ingresso richiede: la matricola dello studente, l’id della domanda e l’id della risposta data.

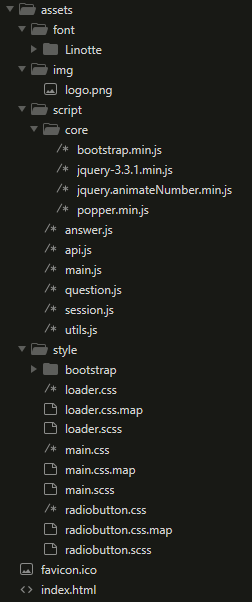
file time.php

Lo script php ha il compito di restituire al client il tempo residuo.

1. Spiegazione del Front end

Struttura del progetto

La struttura del progetto per quanto concerne il *front end* è la seguente:



La directory **assets** contiene tutte le risorse necessarie al corretto funzionamento della parte grafica.

All’interno troviamo diverse sottodirectory che contengono rispettivamente:

* **font:** il font principale utilizzato per tutta la web app, in questo

caso **Linotte**

* **img:** il logo del Politecnico di Bari
* **script:** tutti i javascript che gestino la logica della pagina
* **style:** tutti i fogli di style css necessari alla corretta

visualizzazione della pagina web.

E’ presente anche un file **favicon.ico** che mostra l’icona nel tab del browser.

L’ultimo file, ovvero **index.html**, contiene la struttura principale della web app, che verrà modificata dinamicamente di volta in volta durante l’esecuzione del codice JavaScript.

Spiegazione pagina principale Index.html

In questo file si nota la classica struttura di HTML5 che consente la creazione di pagine web.

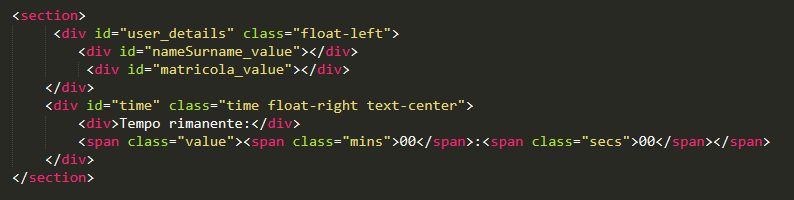
I due tag principali sono **head** e **body**.

In **head**, attraverso i tag **link**,vengono importati i fogli di stile quali:

* **Loader,** per lo spinner usato nelle fasi di caricamento;
* **Radiobutton,** per il rendering degli input di tipo **radio** per la scelta delle risposte;
* **Main,** per lo stile principale.

Nel **body** è presente:

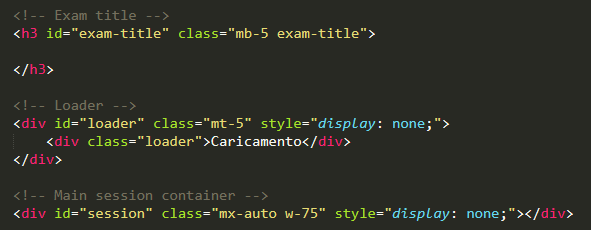
* Una **section** contenente il nome, il cognome, la matricola e il tempo residuo durante la prova;



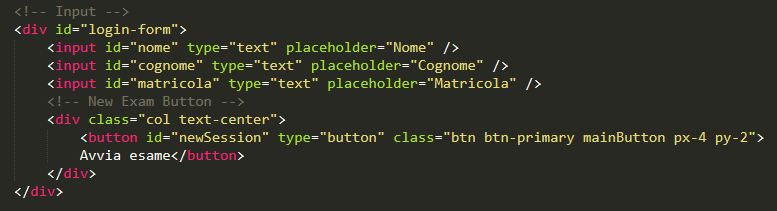
* + - Un tag **img** per mostrare il logo, e dei tag **div** e **span** per la visualizzazione del numero della domanda corrente rispetto al numero complessivo di domande;

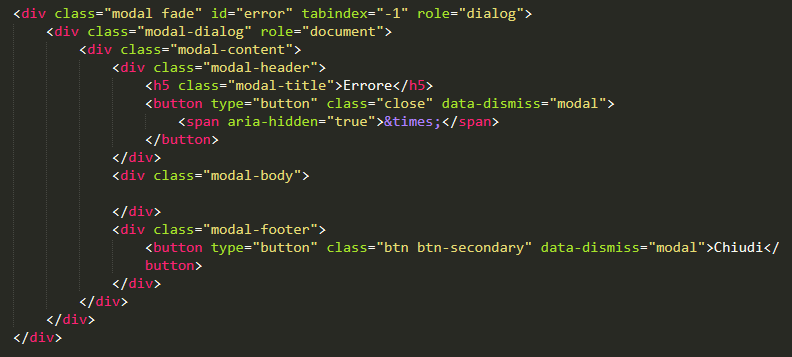
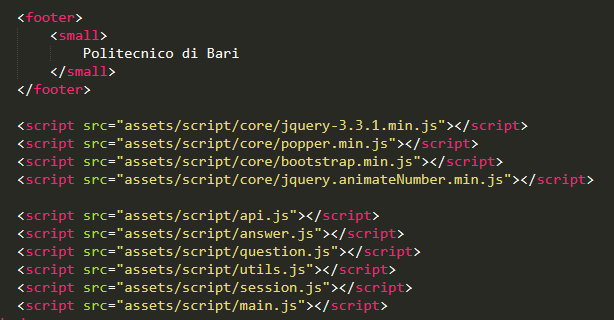


* + - Un tag **h3** per la visualizzazione del titolo dell’esame in corso, il loader (ovvero lo spinner) e il container principale della sessione di esame;



* + - Un form di input per l’inserimento del nome, del cognome e della matricola con relativo bottone per avviare l’esame;



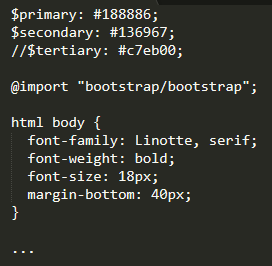
* + - Un **div** con id **error** che mostra l’eventuale messaggio di errore restituito dal back end;
    - In chiusura è presente un tag **footer** ed i relativi script da caricare solo al termine del documento (motivo per cui sono collocati in basso al documento, prima della chiusura del tag **body**).

Spiegazione SCSS e CSS (Cascading Style Sheets)

Per il progetto sono state utilizzare diverse soluzioni per uno styling efficace ed efficiente. E’ stato preferibile utilizzare file SCSS per tutti i vantaggi che essi offrono rispetto al semplice CSS (dichiarazione di variabili globali, chiamate a funzioni quali **mix** per il mescolamento dei colori ecc..), poi compilati attraverso un apposito compilatore in file CSS interpretabili dal browser. Altri file CSS, invece, sono stati importati dall’esterno favorendo il riuso del codice.

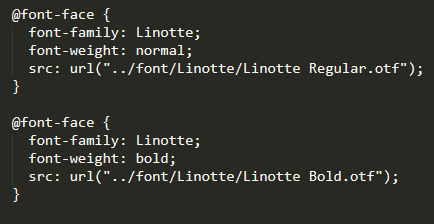
Inoltre, è stato utilizzato il framework per front end **Bootstrap** per la definizione di spazi quali righe e colonne per una migliore impostazione della pagina.

Il framework è presente nella directory sotto il nome di **/www/style/bootstrap/** ed è stato incorporato nel file **main.scss**: unico file fondamentale per la modifica stilistica dei vari elementi nella pagina. Gli altri file presenti nella directory **style** sono codici riutilizzati e compilati. L’unico file necessario per la modifica degli stili della webapp è main.scss. In **main.scss** troviamo la dichiarazione dei colori primario e secondario dell’intera webapp, l’importazione di bootstrap è la definizione degli stili custom definiti da noi per ogni elemento della pagina.



Al termine del file è presente anche l’importazione del font **Linotte**.

In questo caso sono stati importati sia la variante **Bold** che **Regular.**



JavaScript e jQuery

Per la realizzazione del codice che gestisce la logica dell’applicazione è stato utilizzato codice JavaScript con la libreria l’uso per front end **jQuery**, la quale consente di gestire situazioni più o meno complesse con poche righe di codice grazie alle sue API.

All’interno della directory **core** troviamo:

* jQuery versione 3.3.1
* Bootstrap
* Popper (un plugin necessario a mostrare correttamente determinati componenti della User Interface)
* AnimateNumber (un plugin di jQuery utile ad animare in maniera incrementale dei valori numerici sulla User Interface).

Elenco dei file presenti nella directory **script** contenenti:

* **answer.js:**  la classe che definisce la risposta da dare ad ogni

domanda;

* **api.js:** l’interfaccia che collega il front end con il back end;
* **main.js:** è il codice di controllo principale (l’entry point della webapp);
* **question.js:** la classe che definisce ogni domanda dell’esame;
* **session.js:** la classe che definisce la logica relativa alla gestione dell’esame;
* **utils.js:** i metodi ausiliari richiamabili globalmente nell’applicazione.

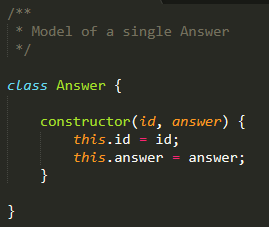
Procedendo per ordine logico analizziamo i singoli file:

* + - 1. answer.js

Il file contiene la classe che definisce il modello di ogni risposta che lo studente può dare ad ogni domanda.

I campi sono:

* **id** (intero)
* **risposta** (stringa)

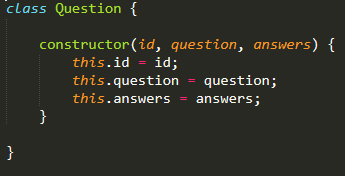


* + - 1. question.js

Contiene la classe che definisce il modello di ogni domanda restituita dal back end.

I campi sono:

* **id** (intero)
* **domanda** (stringa)
* **risposte** (array di oggetti *risposta*)



* + - 1. utils.js

Definisce tutti metodi di supporto (helpers) richiamabili globalmente.

Questi sono:

* **swap,** inverte l’ordine di due elementi;
* **shuffle,** randomizza l’ordine degli elementi;
* **pad,** aggiunge degli zeri per formare una stringa di lunghezza fissa;
* **sleep,** per il blocco dell’esecuzione per un tempo definito.
  + - 1. api.js

E’ la classe contenente l’interfaccia di comunicazione tra back end e front end. All’inizio viene definito un **baseEndpoint** per indirizzare le chiamate HTTP al corretto URL della REST API.

Vengono definiti gli end point che sono:

* **LOGIN,** per eseguire il login dello studente;
* **TITLE,** per ottenere il nome dell’esame in corso;
* **TIME,** per ottenere il tempo residuo utile al

completamento dell’esame;

* **QUESTIONS,** per ottenere la lista delle domande;
* **SUBMIT,** per notificare il server della chiusura dell’esame;
* **FORCEDSUBMIT,** per forzare la chiusura dell’esame in corso;
* **ANSWER,** per inviare una risposta data da uno studente al

Server.

Mentre i metodi statici (auto esplicativi) definiti all’interno della classe sono:

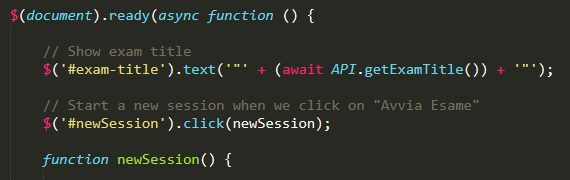
* **login (nome, cognome, matricola)**
* **getExamTitle ()**
* **getTime (matricola)**
* **getQuestions ()**
* **submit (matricola)**
* **submitForced (matricola)**
* **postAnswer (matricola, question\_id, answer\_id)**

Ciò che questi metodi fanno internamente è inviare una richiesta di tipo **GET** o **POST**, a seconda della situazione, al web server che elaborerà una risposta e che verrà restituita in formato **JSON** (in generale)all’oggetto chiamante.

* + - 1. main.js

E’ il file principale che inizializza l’intera applicazione.

Non appena il documento è pronto, ovvero è stato totalmente caricato nel browser, viene utilizzata l’API per mostrare a video il titolo dell’esame per poi assegnare al bottone il metodo necessario all’avvio della sessione dell’esame.



Alla pressione del bottone “Avvia Esame” sono eseguiti dei controlli preventivi lato client per la verifica di inserimento di informazioni corrette.

In caso di successo si viene mostrato lo spinner per il caricamento delle domande, vengono mostrati anche i dati utenti relativi al suo profilo (nome, cognome e matricola), utili alla sua identificazione in fase di esame.

Subito dopo viene instanziata una nuova sessione di esame con le informazioni fornite e viene richiamato il medoto per eseguire il login.

* + - 1. session.js

Gestisce tutta la sessione d’esame.

Vengono definite delle costanti per poter facimente navigare tra le varie domande.

Il costruttore, richiamato al momento della creazione dell’oggetto relativo, inizzializza tutti parametri necessari al funzionamento della sessione.

I metodi presenti nella classe sono:

* **setTimer ()**
* **login ()**
* **fetchQuestions ()**
* **createQuestion (questionData)**
* **initSession ()**
* **setTimerTick ()**
* **changeQuestion (nextOrPrevious)**
* **closeSession ()**
* **forceCloseSession ()**
* **updateTimerUI ()**
* **saveAnswer (answerId)**
* **draw ()**
* **showQuestionsCounter ()**
* **showResults (apiResult)**

**setTimer ()**

Inizzializza il tempo residuo con quello ottenuto dal server master.

**login ()**

Esegue il login attraverso l’API e se la risposta è positiva vengono caricate le domande, altrimenti viene mostrato un messaggio di errore

**fetchQuestions ()**

Vengono caricate le domande attravero l’API e per ogni domanda restituita viene chiamato il metodo **createQuestion**.

Inoltre viene caricato il totale delle domande e avviata la sessione.

**createQuestion (questionData)**

Per la domanda che viene creata vengono istanziate tante risposte quante sono quelle provenienti dall’API, mescolate e inserite all’interno dell’oggetto domanda.

**initSession ()**

Viene aggiornato e mostrato il tempo residua sulla UI.

Viene nascosto lo spinner.

Viene mostrato il contatore di domande affianco al logo.

Viene mostrata la prima domanda.

**setTimerTick ()**

Decrementa il tempo residuo per completare la prova e viene chiusa non appena il tempo risulta scaduto.

**changeQuestion (nextOrPrevious)**

Cambia la domanda in precedente o successiva e richiama il metodo **draw ()**.

**closeSession ()**

Chiude la sessione dell’esame in corso dopo aver controllato da backend che tutte le risposte siano state date.

**forceCloseSession ()**

Mostra il risultato finale dopo aver forzato la chiusura dell’esame.

**updateTimerUI ()**

Aggiorna minuti e secondi residui sull’UI per completare la prova.

**saveAnswer (answerId)**

Invia, attraverso l’API, la risposta data dallo studente per una relativa domanda.

**draw ()**

Genera e rimpiazza il codice HTML sulla pagina per mostrare la domanda precedente o successiva e le relative risposte con i pulsanti **Precedente**, **Successiva**. Viene mostrato il pulsante per chiudere l’esame al termine.

**showResults (apiResult)**

Viene mostrata la schermata nella quale è presente l’esito dell’esame.