**MANUALE TECNICO “myExam”**

# **Configurazione sistema**

Affinché il sistema possa essere utilizzato, è necessario impostare i parametri di configurazione

**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 (htdocs/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**  
all’interno della cartella setup, è possibile trovare il file database.sql. Attraverso la funzione di importazione del DBMS, occorre selezionare il file di cui sopra al fine di creare il database “esame” con la relativa struttura (tabelle).

**Inserimento titolo esame**

All’interno della cartella www/api/requires , è possibile trovare il file config.php. All’interno del file, è possibile trovare 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.

**Creazione Domande**

All’interno della cartella setup, è necessario porre il file questions.xml che conterrà l’elenco delle domande e delle risposte.

Il file dovrà avere la seguente struttura:  
<?xml version="1.0" encoding="UTF-8"?>

<questions>

<question>

<text>Testo Domanda 1</text>

<correctAnswer>Risposta 1</correctAnswer>

<answer>Risposta2</answer>

<answer>Risposta3</answer>

<answer>Risposta4</answer>

<answer>Risposta5</answer>

</question>

…  
 <question>

<text>Testo Domanda N</text>

<correctAnswer>Risposta 1</correctAnswer>

<answer>Risposta2</answer>

<answer>Risposta3</answer>

<answer>Risposta4</answer>

<answer>Risposta5</answer>

</question>

</questions>

1. **Inizializzazione sistema (con importazione delle domande)**

Recandosi all’indirizzo

<http://127.0.0.1:numeroporta/admin/install.php>

sarà possibile inizializzare il sistema.

L’installazione viene eseguita soltanto se il file questions.xml esiste e, nel caso in cui questa condizione è verificata, tutte le tabelle presenti nel db, vengono svuotate, vengono importate le domande e, il file questions.xml viene eliminato.

N.B. Nel momento in cui il file questions.xml non esiste, l’installazione non andrà a buon fine.

1. **Funzionamento del sistema**

Il sistema si basa su una logica basata su un frontend ed un backend.

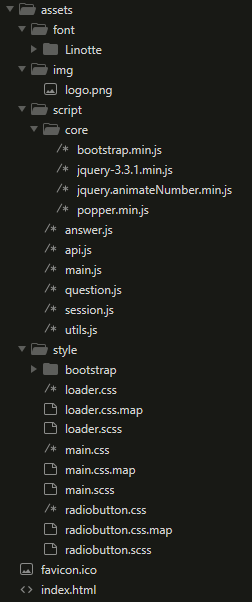
L’obiettivo del frontend è quello di fornire l’interfaccia all’utente mentre, il backend, ha l’obiettivo di fornire la logica di funzionamento.

L’interazione tra backend e frontend, avviane secondo uno schema comune alle api infatti, il client esegue le richieste al server non conoscendo la struttura delle varie componenti.

Per quanto riguarda l’implementazione del backend, esso fornisce i dati al frontend mediante delle API che hanno un comportamento da blackbox, infatti, queste, ricevute dei dati in ingresso, restituiscono dei risultati formattati in JSON.

1. **Frontend:**

La struttura del progetto per quanto concerne il frontend è la seguente:

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

All’interno troviamo diverse sottodirectory:

* **font:** contiene il font principale utilizzato per tutta la

web app, in questo caso **Linotte**

* **img:** contiene il logo del Politecnico di Bari
* **script:** contiene tutti i javascript che gestino la logica

della pagina

* **style:** contiene tutti i fogli di style css necessari alla

corretta visulizzazione della pagina web

E’ presente anche un file **favicon.ico** utile a mostrare l’icona nella finestra del browser.

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

Esaminaniamo nel dettaglio i singoli file.

**Index.html**



E’ presente la classica struttura di HTML5 per la creazione di pagine web.

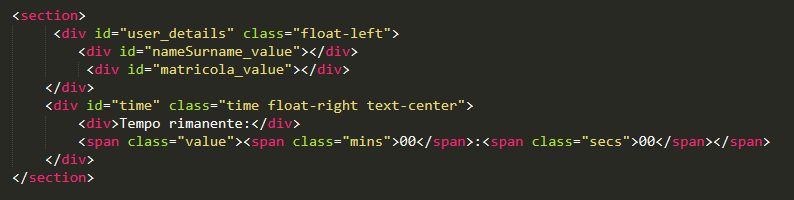
I due tag principale sono **head** e **body**.

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

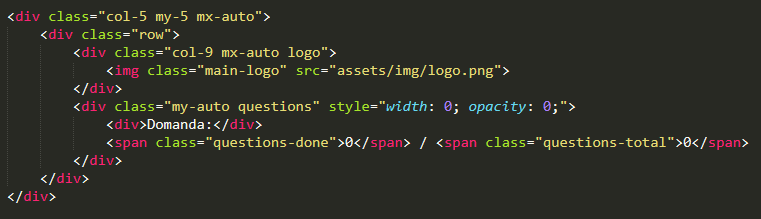
* **loader:**  ci permette di mostrare correttamente lo spinner di caricamento delle domande
* **radiobutton:** per il rendering degli input di tipo **radio** per la scelta delle risposte
* **main:** è il file di stile principale

Nel **body** è presente:

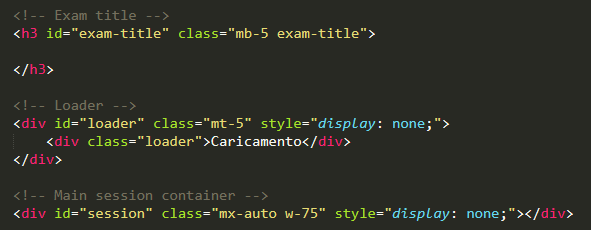
* Una **section** per mostrare nome, cognome, matricola e tempo rimanente durante la prova:



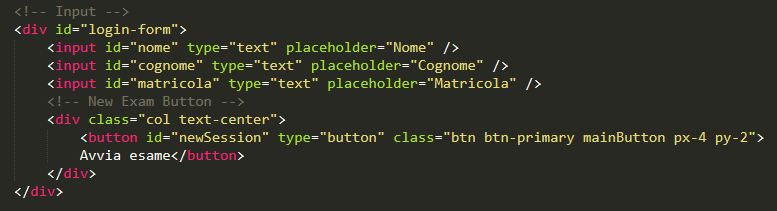
* Un tag **img** per mostrare il logo e dei tag **div** e **span** per la visualizzazione del numero della domanda attuale rispetto il numero totale di domande:

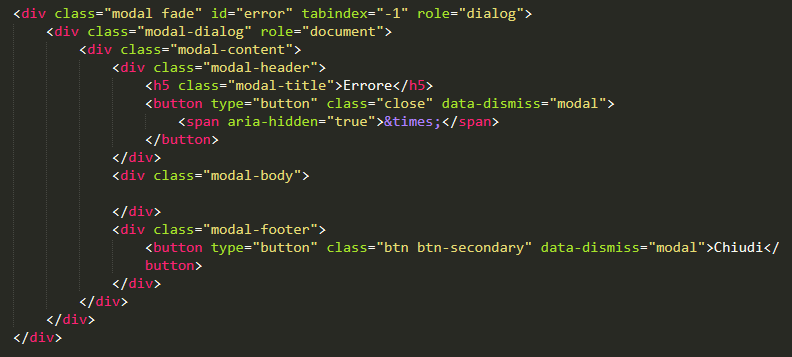
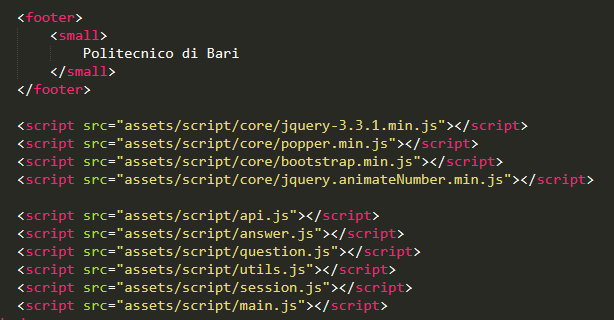


* Un tag **h3** per la visualizzazione del titolo esame caricato attraverso le API, il loader (ovvero lo spinner) e il container principale della sessione di esame:



* Un form di input per l’inserimento di nome, cognome e matricola e relativo pulsante per avviare l’esame:



* Un **div** con id **error** da mostrare in caso di errore di qualsiasi tipo richiamato dal backend:
* 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**):

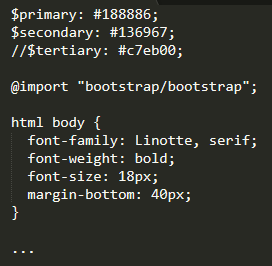
**SCSS e CSS (Cascading Style Sheets)**

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 fuzioni 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.

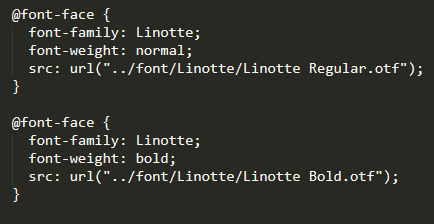
Inoltre è stato utilizzato il framework per frontend **Bootstrap** per la definizione di spazi quali righe e colonne per una migliore impostazione della pagina.

Il framework è presente nella directory sotto il nome di style/bootstrap ed è importato nel file **main.scss**: unico file fondamentale per la modifica stilistica dei vari elementi nella pagina.

**N.B.:** gli altri file presenti nella directory **style** sono codici riutilizzati e compilati, quindi 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 gestile la logica dell’applicazione è stato utilizzato codice JavaScript insieme all’uso di un framework per frontend **jQuery**, il quale permette di gestire situazioni più o meno complesse con poche righe di codice grazie alle sue API.

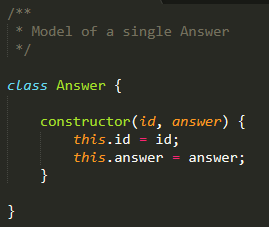
All’interno della directory **core** troviamo:

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

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

* **answer.js:** contiene la classe che definisce la risposta da dare ad ogni domanda
* **api.js:** contiene l’interfaccia che collega il frontend con il backend
* **main.js:** è il codice di controllo principale (l’entry point della webapp)
* **question.js:** contiene la classe che definisce ogni domanda dell’esame
* **session.js:** è la classe che contiene la logica per gestire l’intera sessione d’esame
* **utils.js:** contiene dei metodi ausiliari richiamabili globalmente nell’applicazione

Procedendo per ordine logico analizziamo i singoli file:

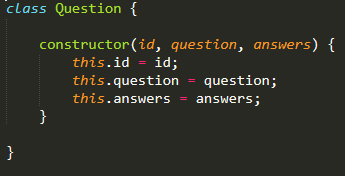
**answer.js**

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

I campi che contiene sono:

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

**question.js**

Contiene la classe che definisce il modello di ogni domanda che arriva dal backend.

I campi che contiene sono:

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

**utils.js**

Definisce tutti metodi definiti *“utili”* all’interno dell’intera applicazione, 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 mettere in pausa l’esecuzione per un tempo definito

**api.js**

E’ la classe contenente l’interfaccia di comunicazione tra backend e frontend.

All’inizio viene definito un baseEndpoint per indirizzare le chiamate HTTP al giusto URL della REST API.

Vengono definiti gli endpoint 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 (autoesplicativi) definiti all’interno della classe sono:

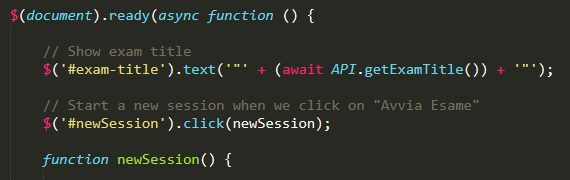
* **login (nome, cognome, matricola)**
* **getExamTitle ()**
* **getTime (matricola)**
* **getQuestions ()**
* **submit (matricola)**
* **submitForced (matricola)**
* **postAnswer (matricola, question\_id, answer\_id)**

Cio che questi metodi fanno internamente è mandare 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.

**main.js**

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

Non appena il documento è pronto, ovvero è stato totalmente caricato nel browser, viene utilizzata l’API per mostrare il titolo esame sullo schermo e subito dopo viene collegata un metodo alla pressione del pulsante **Avvia Esame**.



Alla presione del tasto per avviare l’esame vengono eseguiti dei controlli preventivi lato client per la verifica di inserimento di informazioni corrette.

Passate le quali 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.



**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 della prova.

**showResults (apiResult)**

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

**Backend:**

Le componenti del backend sono:

**examTitle.php**: permette di restituire al client il nome dell’esame

**login.php:** permette al client di eseguire l’autenticazione e, in determiante

circostanze, di recuperare la sessione d’esame

**publish.php:** permette di memorizzare le risposte degli utenti all’interno del db

**questions.php:** permette di fornire al client l’elenco delle domande

**submit.php:** permette al client di terminare la sessione a seguito della pressione

da parte dell’utente del pulsante “termina esame”

**time.php:** permette di restituire al client il tempo residuo

**forcedSubmit.php:** permette di terminare la sessione in modo forzato allo scadere del

tempo