Diario di lavoro

|  |  |
| --- | --- |
| Luogo | Scuola d’arti mestieri Trevano |
| Data | 20.01.2021 |

|  |
| --- |
| Lavori svolti |
| Oggi ho iniziato l’implementazione del progetto.  Ho creato la struttura del progetto utilizzando il template MVC. MVC (Model-view-controller) è un pattern che mi permette di separare meglio le funzioni dell’applicativo dalle varie pagine web. La struttura è la seguente:  Immagine che contiene tavolo  Descrizione generata automaticamente  Nella cartella controllers ci saranno tutti i controller che permettono di fare le azioni sulle varie pagine web.  Nella cartella models ci saranno tutti i model che permettono di manipolare i dati (connessione al database, query, visualizza file, …)  Nella cartella sources ci saranno tutte le librerie che utilizzo nel progetto e per finire nella cartella views ci saranno tutte le views che corrispondono alle pagine web dell’applicativo.  Il file si configurazione del progetto/sito web (config.php) si trova nella cartella config. In questo file ho configurato l’URL del sito.    Nella variabile actual\_link memorizzo il protocollo e il nome del host. In effetti con il metodo isset vado a controllare se sto utilizzando il protocollo https o http. La variabile globale $\_SERVER contiene tutte le informazioni del server. Nella variabile documentRoot vado a memorizzare la cartella di root, ad esempio gestione\_infromazioni\_sede. Con il metodo str\_replace vado a togliere tutti i backsalsh inseriti nell’URL per sbaglio dall’utente. Per finire nella variabile final costruisco l’URL completo.  Per implementare il CSS, cioè lo stile delle varie pagine, ho scaricato la libreria bootstrap. Bootstrap è una raccolta di strumenti per la creazione di siti e applicazioni web. Quest’ultima contiene modelli di progettazione basati su HTML e CSS. Contiene anche degli script JavaScript che mi permette di creare varie animazioni. Ho scaricato la vesrione 5.0.0-beta1.  Ho creato una cartella \_templates nella cartella views. Al suo interno c’è il file header.php e footer.php. L’header.php corrisponde alla pagine di intestazione di tutto l’applicativo web. Nel seguente file devo includere tutte le librerie come ad esempio la libreria di bootstrap.  Immagine che contiene testo  Descrizione generata automaticamente  Ho anche incluso lo script bootsrap.min.js che mi permette di caricare i vari JavaScript come ad esempio, l’animazione del menu hamburger (modalità telefono).  Oltre alle libreria ho anche creato la navbar che corripsonde alla barra di navigazione del sito web.  Immagine che contiene testo  Descrizione generata automaticamente  La classe navbar-expand-sm permette di rendere visibile l’icona del menu haburger. Quando un utente diminuisce la dimensione dello schermo, i link vengono collassati e appare al loro posto un icona a forma di hamburger. Se l’utente clicca questa icona, la navbar si espande verso il basso mostrando i vari link.  Navbar (versione desktop)    Navabar (versione mobile)      Il file footer.php corrisponde al pie di pagina delle varie pagine web dell’applicativo. In effetti ho creato il seguente footer.    Immagine che contiene testo  Descrizione generata automaticamente  La classe container permette di creare un contenitore suddiviso in griglia. La griglia mi permette di gestire il responsive dei vari elementi presenti nel contenitore. La classe row mi permette di creare una nuova riga e la classe col mi permette di creare una nuova colonna.  Ho creato anche la views home nella quale ho iniziato ad implementare il form che mi permette di inserire un nuova informazione da rendere visibile.  Immagine che contiene testo  Descrizione generata automaticamente  La classe form-group permette di raggruppare diversi input nello stesso contenitore. La classe valid-feedback permette di rendere visibile un messaggio se l’utente scrive qualcosa di corretto nel campo input, invece la classe invalid-feedback è quando un utente inserisce un valore non valido. In effetti la classe needs-validation permette di controllare che un input sia corretto o sbagliato.  Quando un utente no scrive niente appare il seguente messaggio e l’ombreggiatura dell’input diventa rosso. Se invece è corretto l’ombreggiatura diventa verde.      Per realizzare questa animazione ho dovuto implementare il seguente script.  Immagine che contiene testo  Descrizione generata automaticamente  Il metodo addEventListener imposta uan funzione che verrà chiamata ogni volta che l’evento specificato (caricamento del form) viene consegnato all’elemento di destinazione. Il metodo Array.prototype.filter permette di eseguire il loop sul form impedendo la visualizzazione dei messaggi di feedback. Il metodo checkValidity permette di controllare che un elemento ha dei vincoli e se li soddisfa, cioè se l’utente ha inserito qualcosa nell’input ho se non ha inserito niente. La proprietà classList di sola lettura restituisce una raccolta DOM dinamica delle classi dell'elemento. |

|  |
| --- |
| Problemi riscontrati e soluzioni adottate |
| Oggi non ho riscotrato dei problemi. |

|  |
| --- |
| Punto della situazione rispetto alla pianificazione |
| Rispetto alla pianificazione sono ad un buon punto. |

|  |
| --- |
| Programma di massima per la prossima giornata di lavoro |
| Per la prossima giornata di lavoro devo terminare il form che mi permette di inserire le informazioni da rendere visibili. In seguito devo creare un dropzone che mi permetta di caricare i filmati/presentazioni e per finire devo creare il sistema calendario. |