Ds Group, Via Pompeo Mariani, 4, 20128, Milano MI
Data: 8 Marzo 2018

TEAM:
Lo Preiato Vittorio, Arrom Sami

Analisi e Metodo di Sviluppo dell'applicazione Web "JsonUser"

Analisi della richiesta:
È stato richiesto in sede di lavoro, di sviluppare un'applicazione web che potesse richiedere l'utilizzo delle tecnologie apprese durante il corso in merito allo sviluppo web. Tale pagina deve svolgere la funzione di richiamo dei dati da un server, e di renderli visibili all'utente, aggiungendo alcune funzioni come la possibilità di vedere nel dettaglio i dati o di modificarli a seconda delle proprie esigenze.

Progettazione della richiesta:

-HOME PAGE (index.html):
in questa sezione verranno raccolte le informazioni relative ad ogni user registrato nel db "https://jsonplaceholder.typicode.com/users", ogni utente sarà di seguito inserito in una tabella con l'aggiunta delle opzioni di modifica e visualizzazione dei dettagli per ogni utente. Sarà inoltre possibile aggiungere i dati di un nuovo utente che verrà aggiunto di conseguenza alla tabella degli utenti.

-DETAILS PAGE (details.html):
in questa sezione verranno elencati in dettaglio i dati dell'utente selezionato dalla tabella e registrato sul database che non sono visibili nella tabella. Sarà possibile accedere alla sezione dei dettagli selezionando la voce "details" presente su ogni riga della tabella per ogni utente presente. Nel database.

-CREATE/MODIFY/DELETE PAGE (form.html):
questa sezione include una pagina multi-funzione generata dal codice JavaScript all'interno della quale è presente una form che cambierà funzione in base alle necessità, dov'è possibile inserire, modificare o cancellare i dati di ciascun utente in base alle proprie esigenze modificando di conseguenza i dati di ciascun utente a proprio piacimento.

Tecnologie utilizzate:

-Javascript:

l'idea è quella di rendere dinamiche anche le righe della tabella degli utenti in base alla quantità di utenti salvati nel db, verranno quindi create n righe per n user presenti nel server.

-Ajax:
viene utilizzato per le richieste dei serivizi.

-JQuery :
utilizzato per il binding dei dati.

-HTML:

la struttura dell'applicazione è stata creata rispettando i tag semantici del linguaggio HTML5.

CSS:

Implementazione della libreria Bootstrap (versione 4.0.0-alpha.6) per la stilizzazione del

DOM.

Layout:

Il layout dell'applicazione è stato sviluppato con le regole semantiche di css3 flex.

ALTRO:

Utilizzo di git come sistema di versioning, packet manager npm, gulp, jshint per controllo qualità del codice javasciript, utilizzo del preprocessore css Sass.

Tutti le informazioni necessarie alla manipolazione delle informazioni fra le varie pagine (id, selettore modalità creazione/modifica/delete) verranno salvate nel localStorage.

Sviluppo dell'applicazione:

l'applicazione è stata pensata per la visualizzazione di tre sezioni.
L'Index è la prima e si occupa della visualizzazione parziale di ciascun utente inserito in una tabella. Questa tabella non è stata sviluppata con i tag html, bensì viene creata da una pagina JavaScript, questa pagina avvia la funzione "loadTable" a cui verrà passato come parametro l'oggetto ricevuto dal servizio rest. La successiva funzione, "clickBtnTable" è una funzione di servizio che al click del bottone esegue la funzione dedicata allo scopo (dettaglio visto più avanti). "$ ('.loader').fadeOut()" si occupa di nascondere il contenitore loader attivo all'esecuzione della pagina web. Infine solo a questa sezione è stato aggiunto un footer che presenta due link, uno di contatti ed uno di informazioni, al click di questi, si aprono due finestre modali che contengono rispettivamente un form da compilare per contattare, ed uno di informazioni.

La seconda pagina è quella dei dettagli, a pagina caricata, esegue una chiamata Get al servizio rest passando l'id salvato nella LocalStorage (in base al suo valore, il servizio restituisce l'oggetto con l'user dal medesimo id), verrà successivamente eseguita la funzione loadDetails dove provvederà ad effettuare il binding dei dati popolando la pagina in modo tale da poter visualizzare anche gli altri dati dell'utente selezionato che non erano visibili nella tabella dell'index.

La terza ed ultima pagina invece, è una form, in cui si possono effettuare diverse operazioni. A pagina caricata, legge dal localStorage in quale stato si trova la variabile "selector" (possibili valori: create, modify, remove), in base a ciò, esegue la funzione corrispondente ("loadCreate()" o "loadModifyRemove()") che si occuperà di adattare la pagina in base alla funzione richiesta. "backButton()" è una funzione di ascolto sul bottone "indietro" presente nella pagina, se premuto, reindirizza alla pagina index.js. In parole povere, in base alle nostre esigenze, la pagina si adatterà allo scopo per cui è stata richiamata, nel caso un utente dovesse modificare alcuni dati di un determinato user, può selezionare il pulsante di modifica ed essere indirizzato al form, che si adatterà alla richiesta e svolgerà soltanto il compito di modificare i dati dell'user selezionato.

Ogni pagina è stata stilizzata con le regole flex del css3 in modo da rendere responsive ogni pagina affinchè possano essere perfettamente adattate su ogni dispositivo. Inoltre la libreria bootstrap si è rivelata fondamentale per la realizzazione dell'applicazione, con essa sono stati aggiunti vari contenuti come i button predefiniti, una stripped table in cui venivano inseriti gli utenti, una classe jumbotron per gli header delle varie pagine, ed il footer inserito nell'index con due modali di contatti e di informazioni. La stilizzazione inoltre è stata realizzata con l'ausilio dell'estensione sass i quali file sono tutti inseriti in una apposita cartella che contiene un index.scss .

GULP TASK

Sono presenti 2 task:

sass:watch avviato, il task si mette in attesa di compilazione fin quando non viene modificato un file sassi, in quel caso avvia la compilazione del file css aggiornato alle ultime modifiche attuate.

Build task la cui funzione è quella di creare una cartella dist con tutti i file necessari alla distribuzione finale. Si occupa di unire tutti i file Javascript, minificare il nuovo file creato (build.js) e spostarlo nella cartella dist, uguale per il css, dopo averlo minificato viene spostato nella medesima cartella. I file html subiranno una sostituzione delle patch nei link del css e del javascript, inoltre vengono eliminati i collegamenti a tutti gli altri file .jv. .

Repository git progetto: https://github.com/falconeta/testDsGroupTeam.git