usauto.it

federico.rispo@studenti.unipd.it

Relazione di Progetto

Informazioni sul Documento

Referente	Federico Rispo
Autori	Federico Rispo 594374
	Federico Brian 1122698
	Georgiana Uglea 1121248
	Alessandro Canesso 1122701
Lista distribuzione	*Prof. Lamberto Ballan*
	Prof. Ombretta Gaggi
	Dott. Matteo Ciman

Scopo del Documento

Il presente documento contiene la relazione del progetto didattico del corso di Tecnologie Web svolto nell'A.A. 2018/2019

Indirizzo del sito web:

http://tecweb1819.studenti.math.unipd.it/frispo

Abstract

Il progetto, di nome “usauto” ha come obiettivo la creazione di un sito web dedicato alla rivendita di automobili usate tra privati. Lo scopo del sito è di permettere agli utenti di accordarsi per la vendita o l’acquisto di mezzi pubblicati su un mercato comune. I visitatori del sito (utenti non registrati) possono visualizzare tutti gli annunci pubblicati, cercare autovetture con qualità specifiche e contattare altri utenti per acquistare o saperne di più sui relativi annunci da loro messi sul mercato. Se un visitatore vuole pubblicare uno o più annunci riguardanti autovetture che ha intenzione di vendere è obbligato ad effettuare una registrazione, la quale permette di diventare utenti registrati. Una volta effettuato l’accesso, oltre alla creazione e pubblicazione di un annuncio, egli ha anche la possibilita’ di modificare e cancellare le proprie inserzioni già pubblicate.

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

1.1 Analisi d'utenza

Il sito è rivolto ad utenti in età adulta, intenzionati a vendere un'automobile di loro proprietà od in cerca di un mezzo. Il target è molto ampio in quanto si stima che, in media, in Italia un abitante su due possiede ed utilizza un'autovettura e quindi circa metà della popolazione italiana potrebbe essere interessata alla vendita oppure all'acquisto di un veicolo usato. Il pubblico è molto variegato e composto sia da utenti che da sempre hanno navigato sul web sia da persone poco familiari con la tecnologia.

1.1.1 Guidatori inesperti

I guidatori inesperti sono un tipo di utenza giovane, intesi come i neopatentati ed i guidatori che hanno ottenuto la patente di guida da meno di dieci anni. Abituati alla tecnologia, sono alla ricerca di un sito web moderno per poter acquistare la loro prima auto, eseguendo ricerche veloci preferibilemente dal loro dispositivo mobile.

1.1.2 Guidatori esperti

I Guidatori esperti sono un tipo di utenza tendenzialmente poco avvezza alle tecnologie, poiché si riferisce alla fascia d'età delle persone con età maggiore ai 28 anni. Essi comprendono lavoratori e/o pensionati in cerca di un'auto per sé e padri di famiglia in cerca, ad esempio, della prima auto da acquistare o regalare al proprio figlio neopatentato. Costoro effettueranno ricerche ad ampio raggio, poiché vorranno visionare tutte le auto appartenenti ad una determinata classe (per stile, cilindrata, ...) prima di decidere quale acquistare.

1.2 Considerazioni finali

Attraverso un'interfaccia intuitiva ed un linguaggio informale il sito cerca di essere fruibile alla quasi totalità delle persone interessate alla compravendita di automobili. Inoltre si cerca di non allontanare i meno esperti permettendo una ricerca modulare con molti campi opzionali. In questa maniera si spera di non intimidire gli utenti con poca dimestichezza per quanto riguarda le componenti specifiche delle automobili. Questo vale anche per la sezione annunci: è infatti possibile pubblicare un annuncio senza scendere nei dettagli riguardanti le qualità di un veicolo. Il sito dovrà essere responsive, in modo da potersi adattare a ogni tipo di dispositivo, e dovrà essere semplice e intuitivo per favorire gli utenti meno avvezzi alla tecnologia.

2 Sviluppo

2.1 Progettazione

La tipologia del sito è stata decisa durante la fase preliminare: in seguito ad un *brainstorming* si sono delineate le idee del gruppo; idee che hanno permesso di elencare gli obiettivi da raggiungere durante lo sviluppo del progetto.
Successivamente, è stata organizzata una suddivisione generale delle responsabilità individuali.
Una volta fatti i primi passi si sono decisi il layout del sito, i pattern da seguire per lo sviluppo, le porzioni principali del sito e le tecnologie da utilizzare.
Infine, è stato progettato il database, il quale permette l'immagazzinamento e l'organizzazione delle informazioni sulle quali è fondato il sito web: **inserzioni**, **utenti**, **provincie**, **regioni**, **equipaggiamenti** montati dalle macchine, vari **optional** delle macchine, le **macchine** stesse, le **categorie** a cui appartengono le macchine e i **colori** delle macchine. Si è passati poi allo sviluppo del front-end e del back-end.
Dopo aver effettuato il merge del codice, il sito è stato collaudato per risolvere eventuali bug e verificare l'accessibilità e la compatibilità con dispositivi e browser differenti.

2.2 Design

Dato lo stato in continuo aggiornamento del sito, è stata posta particolare attenzione alla separazione tra contenuto statico e dinamico e alla separazione tra struttura, presentazione e comportamento.
Come standard si è deciso di usare **XHTML 1.0 STRICT**, il quale fornisce un ottimo supporto per strutturare il contenuto in maniera semplice ed efficace: ad esempio, grazie alla sua sintassi rigida, i validatori possono rilevare immediatamente la chiusura corretta dei tag. Le strutture delle pagine web sono state "templatizzate" in modo da incrementare la robustezza del sito web, sul quale sono applicati fogli di stile in CSS puri. I layout in CSS sono stati sviluppati con l'obiettivo di ottenere un design fluido e funzionale per la maggior parte dei browsers e dispositivi. Inoltre attraverso *media queries* si sono resi disponibili diversi layout per finestre browser di diverse risoluzioni, in fine Il supporto a mobile e tablet è stato fornito tramite appositi fogli di stile. Un altro aspetto preso in considerazione è stata l'accessibilità del sito per diverse categorie di utenti, questa è analizzata in dettaglio maggiore successivamente, in una sezione dedicata.

2.3 HTML e CSS

Riuscire a separare la struttura delle informazioni dallo stile è stato fondamentale per il tipo di sito preso in esame.
È stato utilizzato un solo file CSS, opportunamente diviso in categorie, per tutte le direttive di design esposte nelle varie pagine. Nello stesso file troviamo anche

le direttive specifiche per il sito in modalità mobile e tablet. Invece si è scelto di separare il css relativo alla stampa.
Per quanto riguarda le grandezze si è fatta molta attenzione ad esprimerle in unità relative (**rem**, **em**, %).

2.4 MySQL

Il database MySQL è stato utilizzato per memorizzare in modo permanente le informazioni inserite dagli utenti e per fornire procedure sql per facilitare le operazioni su di esse.
Le tabelle principali sono `User`, `Advertisement` e `Car`. La tabella `User` contiene, come il nome suggerisce, le informazioni di tutti gli utenti registrati. Le tabelle `Advertisement` e `Car` contengono tutte le informazioni degli annunci pubblicati. Abbiamo deciso di separare le informazioni in due tabelle per rendere la struttura più estendibile a future modifiche, infatti le due tabelle sono collegate da una chiave esterna `item = idCar` quindi se, ad esempio, in futuro si volessero gestire oggetti che per loro natura hanno caratteristiche diverse dalle autovetture (si pensi alle moto) basterà creare una nuova tabella e creare una nuova chiave esterna senza stravolgere la struttura corrente.
Inoltre abbiamo creato delle tabelle di utilità come `Region`, `Province`, `Category` e `Color` che contengono delle informazioni che vengono riferite molte volte dai record delle altre tabelle. Questo ci permette di salvare molto spazio nel database e renderlo scalabile.
Il database espone delle funzionalità tramite procedure per effettuare inserimenti, modifiche ed eliminazioni, preservando la coerenza dei dati. Questa scelta rende il backend completamente modulare ed estendibile perché il database è completamente slegato dal codice PHP, infatti il codice sql è memorizzato nel database e all'interno del codice PHP è presente solo la connessione ad esso e la chiamata alle procedure. Se in futuro si volesse cambiare tipologia di database questo richiederebbe il minimo sforzo.

2.5 PHP

L'altra componente cruciale server side è PHP, che gestisce tutte le richieste ricevute dal server effettuando redirect alle componenti assegnate per soddisfare le richieste.
Abbiamo utilizzato una programmazione orientata agli oggetti adottando il pattern architetturale MVC per mantenere separata il più possibile la struttura statica delle pagine HTML dal comportamento dinamico.
Model: Fanno parte del model tutte le classi che si interfacciano con il database. Per una migliore estendibilità del codice vi è un'interfaccia `Repository` che espone dei metodi che vengono implementati nelle classi concrete che hanno il compito di effettuare le richieste al database. Ad esempio la classe `AdvRepository` implementa le funzioni `save()`, `update()`, `remove()`, `findOne()`, `findAll()` relative alle operazioni di inserimento, modifica, cancellazione e prelievo degli annunci. È presente la classe `MySqlDB` che, oltre a

gestire la connessione con il database, fornisce delle funzioni che generalizzano l'azione di invio di una richiesta query rendendo più leggibile il codice all'interno delle varie classi `*Repository.php`.
View: la classe `View` ha il compito di delineare la struttura generale di tutte le pagine HTML e di fornire delle funzioni per:

- Settare alcuni parametri come ad esempio il titolo della pagina o le variabili dinamiche da iniettare all'interno delle pagine.
- Creare dinamicamente e correttamente i campi di input e di selezione
- Comporre la struttura finale del documento HTML importando tutte i file html necessari nell'ordine corretto

Controller: abbiamo creato tre controller, uno per ogni macro categoria da gestire:

- `AdvController` che gestisce tutte le pagine relative agli annunci: inserimento, ricerca, dettaglio ecc.
- `MainController` che gestisce la pagina principale e tutte le pagine di supporto come ad esempio i *Crediti* e *Contatti*
- `UserController` che gestisce tutte le pagine relative all'utente: registrazione, accesso, profilo ecc.

Ogni controller contiene diverse funzioni e ognuna soddisfa uno dei seguenti compiti:

- Controllare i valori in input forniti dall'utente mediante una form
- Invocare il model relativo per inviare o ricevere dati dal database
- Creare l'oggetto View per generare la pagina HTML richiesta dall'utente contenente le informazioni richieste
- Reindirizzare la richiesta ad altre pagine

Routing: Ogni indirizzo URL è della forma `dominio.it/controller/function/parameter1/parameter2` e ogni richiesta HTTP viene gestita dal file `routing.php`. Quindi se, ad esempio, si volesse accedere al dettaglio di un'annuncio avente identificatore **27** basterà digitare il seguente url `dominio.it/adv/show/27` e la richiesta HTTP verra' gestita dal file di routing che si occuperà di

- Scomporre l'url in `adv`, `show` e 27
- Creare il controller `AdvController` dopo aver verificato la sua esistenza
- Invocare la funzione `show()` passandogli il valore 27 come parametro

Nel caso in cui l'indirizzo url fosse sbagliato l'utente viene indirizzato ad una pagina di errore 404.
Utility: Abbiamo implementato delle classi di utilità che vengono utilizzate in diverse situazioni:

- **Validate** Questa classe contiene una serie di funzioni statiche che permettono di validare ogni signola tipologia di input e di convertirlo nel formato più adeguato in base alle esigenze.
- **ImageResize** Questa classe permette di generare delle miniature (*thumbnail*) per ogni immagine caricata dagli utenti. Queste verranno poi utilizzate nei risultati della ricerca al posto delle immagini di dimensioni originali così da risparmiare bit durante la trasmissione della pagina HTML e facilitare la navigazione anche agli utenti con una connessione internet non performante.

2.6 JavaScript

È stato utilizzato il linguaggio JavaScript per rendere dinamico il comportamento *client-side* di alcune pagine di usauto. In particolare, l'utilizzo di JavaScript è circoscritto al controllo dell'input nei *form* di ricerca: questo è il frutto di una decisione presa all'unanimità in quanto non è possibile fare alcuna assunzione circa l'utilizzo di tale tecnologia, poiché può essere disabilitata o non presente. Proprio per questo motivo il controllo dell'input non è eseguito solamente da funzioni JavaScript: nell'eventualità che esso non sia presente, sarà compito del **PHP** fornire un controllo *server-side*, il quale fornirà una visualizzazione opportuna in caso di errore. Di seguito sono elencate le funzioni implementate, seguite da una breve descrizione.

Elemento	Descrizione
`wrongInput`	Messaggio d'errore standard da visualizzare in caso di input sbagliato
`letters`	Espressione regolare che ammette solo lettere maiuscole, minuscole e lettere accentate
`numbers`	Esoressione regolare che ammette solo numeri, da zero a nove
`errors[]`	Array che contiene la form che ha generato l'errore con il messaggio d'errore da visualizzare
`displayError()`	Funzione che controlla l'array *errors* e decide se visualizzare il messaggio d'errore e disabilitare il pulsante di ricerca
`checkLetters()`	Funzione che controlla che l'input delle form sia corretto, ammettendo solo lettere
`checkNumbers()`	Funzione che controlla che l'input delle form sia corretto, ammettendo solo numeri

– *continua a pagina successiva*

– continuazione da pagina precedente	
`invalidInput`(element, paragraph, message)	Funzione invocata in presenza di un input errato in una form. Riceve come parametri attuali l'elemento (form) contenente l'input errato, l'id del paragrafo dove visualizzare il messaggio d'errore (tipicamente `inputErr`) ed il messaggio d'errore da visualizzare. Si occupa di modificare lo stile della form per darle una colorazione del bordo rossa e ad aggiungere l'errore all'array `errors`, se esso non lo contiene già. Viene infine invocata la funzione `displayErrors()`
`validInput`(val, element, paragraph)	Funzione invocata in presenza di un input corretto in una form. Riceve come parametri attuali il valore corretto da visualizzare (tipicamente il valore già contenuto dalla form), l'elemento interessato (form) contenente l'input corretto e l'id del paragrafo che contiene gli eventuali messaggi d'errore da visualizzare (tipicamente `inputErr`). Si occupa di modificare lo stile della form per darle una colorazione del bordo nera e a togliere l'errore all'array `errors`. Viene infine invocata la funzione `displayErrors()`

Tabella 1: Elementi implementati nel JavaScript

3 Accessibilità

3.1 Separazione tra contenuto, presentazione e struttura

Per migliorare la qualità dell'utilizzo del sito agli utenti con disabilità è stato posto l'accento sulla separazione tra presentazione, struttura e comportamento. La separazione è stata implementata tramite documenti XHTML strict 1.0, che richiamano fogli di stile CSS esterni.
Nonostante il controllo input e i messaggi di errore siano stati realizzati in JavaScript, esistono funzioni PHP per le stesse operazioni che permettono il corretto funzionamento del sito anche su dispositivi che non lo supportano. La totalità del codice è stata scritta seguendo le raccomandazioni del **W3C**[1], accertandosi che fossero rispettate tramite validazione.

3.2 Facilitazioni per la navigazione

L'interfaccia del sito è stata ideata per avere una navigazione comoda attraverso tab, non è stato quindi necessario aggiungere un attributo `tabindex` per dare un ordine prioritario alle tabulazioni. Tutte le immagini sono state marcate con apposite label e tag `alt`.
In modo da poter essere accessibile da dispositivi con schermi di dimensioni minori rispetto ai desktop, il sito web è stato costruito per avere un layout *responsive*. Sempre con questo scopo è stata posta particolare attenzione allo sviluppo orizzontale del sito, evitando in qualunque circostanza lo scroll orizzontale. Questo è risultato fondamentale per l'usabilità del sito su smartphone.
Con lo scopo di minimizzare il disorientamento e la confusione dei visitatori è stata creata una pagina di **errore 404**, contenente link utili a riprendere la navigazione nel sito e un logo personalizzato che, oltre ad aggiungere particolarità alla pagina, cerca di ridurre la frustrazione dell'utente sfruttando l'ironia.
Ogni link è stato reso distinguibile da ogni altro tramite apposite regole di stile, **hover** e **visited**: nel caso dell'hover la sottolineatura del link si ispessisce e scurisce, nel caso del visited il link cambia colore. All'apice di ogni pagina è presente una barra di navigazione composta da collegamenti rapidi alle funzionalità più importanti del sito. Ogni qualvolta l'utente si trovi in una pagina corrispondente ad una di queste funzionalità il relativo collegamento si evidenzia in modo aiutare l'utente nell'orientamento. Inoltre, avendo rispettato la regola dei 4 click, non abbiamo giudicato necessaria la creazione di una sitemap.
Un ulteriore aiuto al visitatore viene fornito dal breadcrumb. Il breadcrumb è una lista ordinata di collegamenti ipertestuali alle pagine che rappresentano il percorso dalla pagina iniziale alla pagina corrente. Ogni elemento della lista è separato da un front slash e l'ultimo elemento, che è la pagina corrente, non ha link. Esso consente di tornare con facilità alle pagine visitate in precedenza.

[1] https://www.w3.org/

L'accessibilità del sito è stata migliorata ulteriormente dalla presenza di un tasto ancora che permette di tornare all'apice della pagina.
Tutte parole di lingua diversa da quella italiana sono state racchiuse all'interno di tag con l'attributo lang, per permettere una corretta lettura da parte degli screen reader.

3.3 Schema colori

Per quello che concerne l'accessibilità delle categorie di utenti con difficoltà visive si è deciso di usare unicamente colori "**web safe**", mantenendo sempre un contrasto elevato tra lo sfondo e le scritte, evitando così di renderle meno leggibili; non è presente inoltre alcun contenuto pericoloso per utenti a rischio epilessia, o contenuti che possano indurre l'utente in stato confusionale. Per quanto riguarda le immagini non è stato possibile fare molto perché vengono caricate da utenti, è possibile però ingrandirle cliccandoci sopra.
(Inserire screenshot del sito versione desktop anche con modalità per disabili)

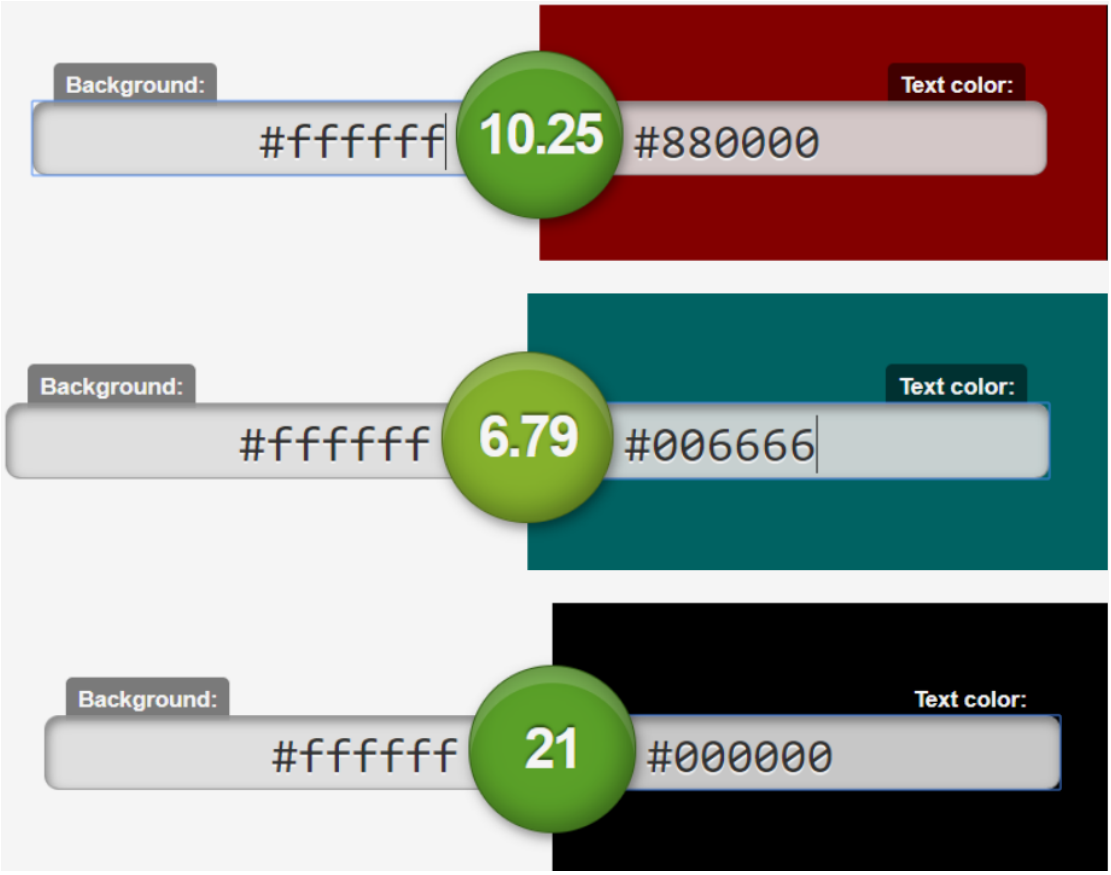

Figura 1: Schema dei colori

3.4 Accessibilità mobile

L'accessibilità per mobile è stata sviluppata con CSS puro, questo ha garantito la compatibilità con la maggior parte dei dispositivi mobile.
(Inserire screenshot del sito versione mobile)

3.5 CSS di stampa

Viene incluso un layout di stampa per ogni pagina: le informazioni principali di una pagina vengono estrapolate e organizzate in un formato adeguato. Sono stati tolti tutti gli elementi visivi non strettamente necessari al contenuto, quindi tutte le immagini di background o di presentazione. Abbiamo rimosso anche il menu, la barra di ricerca, il footer e tutti i form di ricerca e di invio di un messaggio perché non ne abbiamo ritenuta fondamentale la visualizzazione su una pagina stampata. Tuttavia abbiamo deciso di mantere il form per la creazione e modifica di un annuncio perché, essendoci molti campi da compilare per le caratteristiche di un'autovettura, all'utente può tornare utile avere traccia di quali informazioni sono richieste per completare un'annuncio.
(Inserire screenshot)

4 Installazione

4.1 Requisiti

Per un corretto funzionamento del sito i requisiti minimi sono i seguenti:

- PHP versione 7.0 o migliore;
- Un server HTTP - Apache consigliato;
- Un motore database compatibile con MySql.

4.2 Configurazione e dump

Abbiamo creato un file `sql/init.sql` che contiene la struttura delle tabelle, il popolamento e le procedure sql. Prima di utilizzare il sito è necessario importare all'interno del database questo file mediante il comando `source path/sql/init.sql`.
Il file `utility/config.php` contiene tutte le variabili d'ambiente che devono essere configurate in modo tale che l'accesso al database e il caricamento dinamico delle pagine funzioni correttamente. In particolare si deve inizializzare l'url del sito web e le credenziali per accedere al database.
Per un funzionamento corretto sono necessarie alcune configurazioni:

- estensione "mysqli" di PHP attiva necessaria per effettuare le richieste query al database;
- estensione "gd" di PHP attiva necessaria per creare le anteprime delle immagini caricate dagli utenti;
- permessi di upload concessi a PHP.

5 Organizzazione interna

5.1 Suddivisione interna dei compiti

- **Referente:** Federico Rispo;
- **HTML e CSS:** Georgiana Uglea, Federico Rispo;
- **MySQL:** Federico Rispo, Alessando Canesso;
- **PHP:** Federico Rispo;
- **JavaScript:** Federico Brian;
- **Grafiche:** Georgiana Uglea;
- **Contenuti:** Alessandro Canesso, Georgiana Uglea;
- **Accessibilità:** Federico Rispo;
- **Relazione:** Alessandro Canesso, Federico Brian;
- **Testing:** Federico Brian.

6 Tecnologie utilizzate

6.1 Elenco strumenti

- **Versionamento:** Git, GitLab;
- **IDE:** Vim, Geany, WebStorm, PhpStorm;
- **Stesura relazione:** Microsoft Word, TeXMaker (per LaTeX);
- **Grafica:** Photoshop, Adobe Illustrator, GIMP;
- **DBMS:** MariaDB;
- **Test di contrasto dei colori:** https://contrast-ratio.com/
- **Verifica indice di leggibilità:** https://farfalla-project.org/readability_static/
- **Verifica accessibilità:** https://achecker.ca/checker/index.php