Coupons Site

Corso di Tecnologie Web

Gruppo 17

Marzo 2023-Giugno 2023

Indice

1 Riferimenti

Coupons Site è stato sviluppato dal **Gruppo 17**, composto da:

Cognome	**Nome**	**Matricola**
Colabella	Davide	1098071
Fiorucci	Lorenzo	1099692
Giuliani	Matteo	1098285
Nollino	Ludovico	1093732

2 Contributo

Il contributo di ciascun membro, espresso in percentuale, è contenuto nella seguente tabella:

Cognome	**Nome**	**Contributo**
Colabella	Davide	35%
Fiorucci	Lorenzo	15%
Giuliani	Matteo	35%
Nollino	Ludovico	15%

2.1 Strumenti Utilizzati

Per la realizzazione di Coupon site sono stati utilizzati i seguenti strumenti:

- uizard.io: piattaforma online per la realizzazione dei mockup.
- draw.io: piattaforma online per la realizzazione dello schema dei link e la struttura del database.
- PhpStorm: IDE per la scrittura del codice.
- GitHub: piattaforma di condivisione del codice tra i membri del gruppo, usato anche come storico per tenere traccia delle modifiche apportate durante lo sviluppo.
- Bootsrap: framework utilizzato per la realizzazione di alcune componenti frontend.
- Laravel: framework utilizzato per la realizzazione del sito web.

2.2 Suddivisione del lavoro e andamento dello sviluppo

Per un' ottimizzazione delle tempistiche, il gruppo ha optato per la seguente suddivisione dei lavori:

Coppia	**Livelli**
Colabella Davide Fiorucci Lorenzo	0-1
Giuliani Matteo Nollino Ludovico	2-3

In seguito a questa suddivisione dei lavori sono state poi fissate delle scadenze in modo da rispettare i tempi di consegna fissati dal professore. Inoltre qualsiasi modifica apportata al progetto è sempre stata comunicata prima di essere caricata su github. L'andamento dello sviluppo è stato lineare: si è sempre rispettato ciò che è stato progettato (mockup, schema dei link, schema del DB) e, dalla creazione delle pagine HTML statiche si è subito passati alla loro interconnessione e infine a renderle interattive e dinamiche, utilizzando tutti gli strumenti spiegati nel corso di Tecnologie Web.

3 Descrizione del sito

Coupon site è un sito web dedicato alla pubblicizzazione di offerte promozionali offerte dalle aziende partner e destinate all'utente finale. Il sito web offre agli utenti la possibilità di scoprire e approfittare di sconti, promozioni e altre offerte vantaggiose. Di seguito sono elencate le principali sezioni del sito:

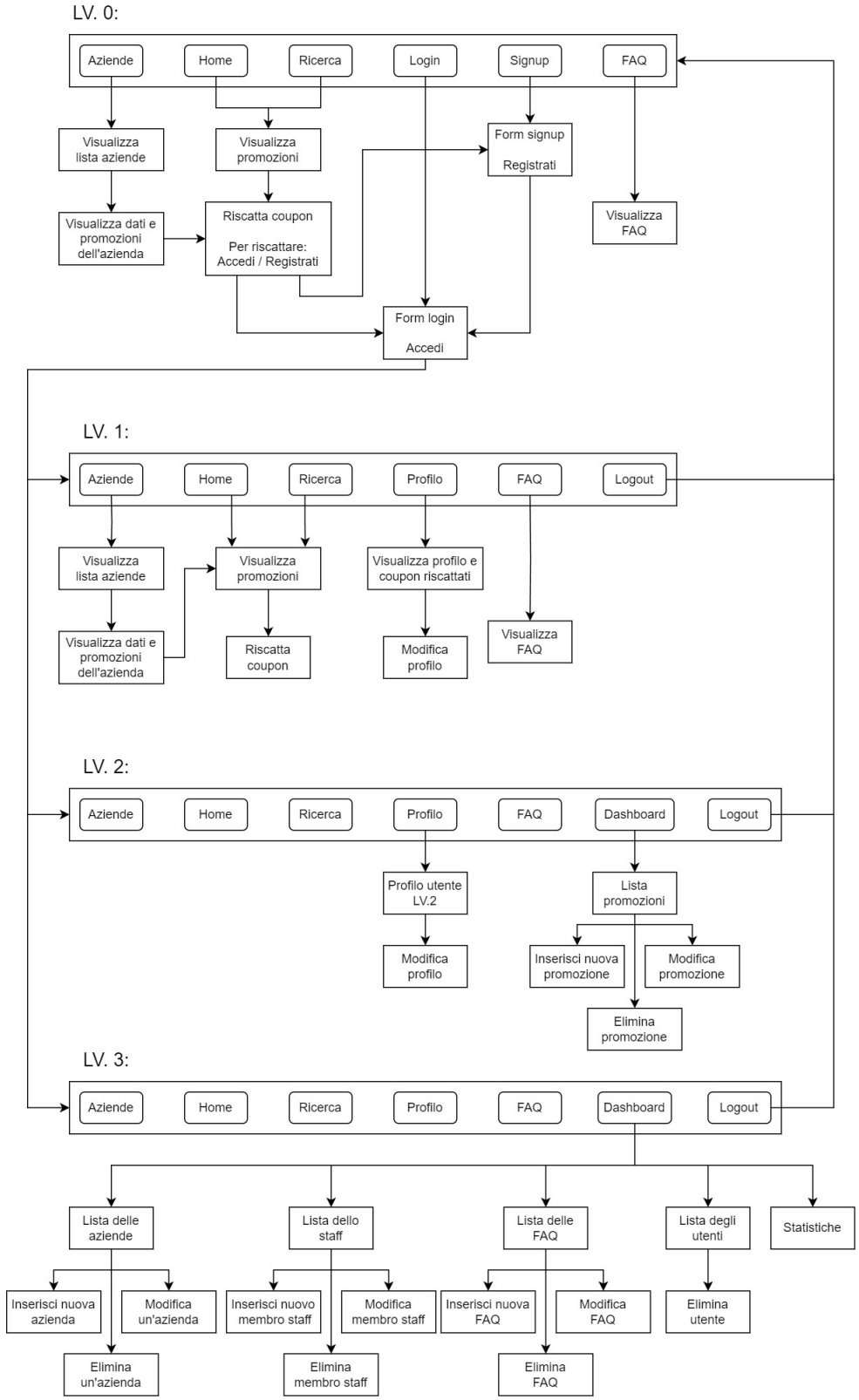
- **Login e Registrazione**:
  - Form che presenta diversi campi di input che permette all'utente di poter accedere al sito oppure di registrarsi. All'atto del login sono richiesti solo username e password mentre per registrarsi c'è bisogno di inserire informazioni anagrafiche (nome, cognome, età), username, email, telefono e password.
- **Home**:
  - Carosello delle nuove promozioni, che presenta le ultime offerte disponibili.
  - Elenco di offerte promozionali di cui l'utente, se iscritto, può riscattare il coupon.
- **Aziende**:
  - Lista delle aziende che offrono promozioni. Di ogni azienda è possibile visualizzare le informazioni e visualizzare le relative offerte disponibili.
- **FAQ**:
  - Sezione dedicata alle domande frequenti (FAQ) completa di risposte.
- **Dashboard**:
  - Sezione dedicata allo staff e agli amministratori del sito. Permette di gestire l'intero sito per mezzo di una semplice interfaccia.
  - Se l'utente che ha esegutio il login ha il ruolo di "staff" allora verrà reinderizzato verso una dashboard che permetterà di eseguire le operazioni di CRUD sulle promozioni, invece se ha il ruolo di "admin" allora la dashboard visualizzata sarà diversa permettendogli di eseguire le seguenti operazioni:
    - Operazioni di CRUD sulle aziende.
    - Operazioni di CRUD sullo staff.
    - Operazioni di CRUD sulle FAQ.
    - Operazione di delete per gli utenti.
    - Operazione di visualizzazione delle statistiche.
  - Le statistiche mostrano il numero totale di coupon emessi, il numero di coupon emessi per promozione e il numero di coupon emessi per singolo cliente.

Gli utenti del sito devono registrarsi per poter riscattare il codice del coupon associato ad un'offerta.

- **Livello 0 - Area Pubblica**:
  - L'area pubblica del sito web viene mostrata interamente all'utente non registrato.
  - Si possono effettuare operazioni di ricerca di un'offerta sia per il nome dell'azienda che ha pubblicato l'offerta e/o per la descrizione dell'offerta.
  - Chiunque ha la possibilità di potersi registare al sito fornendo i propri dati anagrafici e altre informazioni, quali e-mail e numero di telefono.
  - Non ha accesso alle funzionalità degli altri livelli.
- **Livello 1 - Utenti Registrati**:
  - Consente agli utenti di modificare i propri dati personali, escluso l'username.
  - Gli utenti possono riscattare il codice del coupon per le promozioni disponibili e hanno la possibilità di stampare la pagina del coupon in modo da poterlo esibire nel negozio o sito web indicato.
  - Consente di visualizzare lo storico dei coupon riscattati, siano questi ultimi validi o scaduti.
  - Non ha accesso alle funzionalità dei livelli 2 e 3.
- **Livello 2 - Membri dello Staff**:
  - Ogni membro dello staff ha accesso alla sezione pubblica.
  - Consente di modificare il proprio profilo (nome e cognome) e la password di accesso.
  - Consente la gestione CRUD delle promozioni.
  - Non ha accesso alle funzionalità di livello 3.
  - Non ha la possibilità di riscattare coupon.
- **Livello 3 - Amministratore**:
  - Consente l'accesso alla sezione pubblica.
  - Consente la gestione CRUD delle aziende che offrono le promozioni.
  - Consente l'eliminazione degli utenti di livello 1.
  - Consente la gestione CRUD dei membri dello staff.
  - Consente la gestione CRUD delle FAQ del sito.
  - Consente la visualizzazione delle statistiche sull'attività del sito, come il numero totale di coupon emessi, il numero di coupon emessi per promozione selezionata e il numero di coupon emessi a nome di un cliente specifico.
  - Non ha accesso al CRUD delle promozioni.
  - Non ha la possibilità di riscattare coupon.

In sintesi, il sito fornisce un'esperienza diversificata agli utenti a seconda del livello di accesso, consentendo loro di esplorare promozioni, acquisire coupon, gestire promozioni e fornire funzionalità amministrative quali la gestione delle aziende, la gestione degli utenti e la gestione dei membri dello staff. Le statistiche permettono di monitorare l'attività del sito.

4 Organization chart



5 Soluzioni adottate

Di seguito sono elencate alcune delle funzioni più interessanti adottate nello svolgimento del progetto:

5.1 Form validation

Di seguito viene mostrata l'implementazione del codice per la validazione dei campi della form per inserire una nuova promozione.

```
function validateForm(actionUrl, formId){
    //intercetta la perdita di focus dell'input
    $(":input").on('blur',function (event) {
        var formElementId = $(this).attr('id');
        doElemValidation(formElementId, actionUrl, formId);
    });
    $('input[type="date"], input[type="number"]').on('change',
       function() {
        var formElementId = $(this).attr('id');
        doElemValidation(formElementId, actionUrl, formId);
    });
    $("#" + formId).on('submit', function (event) {
        event.preventDefault();
        doFormValidation(actionUrl, formId);
    });
}
```

Questa funzione si mette in ascolto dei cambiamenti dei vari input e dell'invio della form stessa:

- Il metodo onBlur() esegue un'azione quando un qualsiasi campo di input perde il focus.
- Il metodo onChange() viene utilizzato per gli input di tipo *date* e di tipo *number* per risolvere problemi grafici causati dall'utilizzo di onBlur().
- Il metodo onSubmit() intercetta il click del pulsante per l'invio dei dati.

```
function doElemValidation(id, actionUrl, formId) {

    var formElems;

    //prendo il token CSRF dal form e lo aggiungo alla formdata
    function addFormToken() {
        var tokenVal = $("#" + formId + " input[name=_token]").val();
        formElems.append('_token', tokenVal);
    }

    function sendAjaxReq() {
        $.ajax({
            type: 'POST',
            url: actionUrl,
            data: formElems,
            dataType: "json",
            error: function (data) {
```

```
18                  if (data.status === 422) {
19                      var errMsgs = JSON.parse(data.responseText);
20                      console.log(errMsgs.errors[id]);
22                      if (errMsgs.errors[id] === undefined) {
23                          $("#" + id).removeClass('is-invalid');
24                      } else {
25                          $("#" + id).addClass('is-invalid');
26                          $("#" + id).parent().find('.errors').html(' '
                                );
27                          $("#" + id).parent().find('label').after(
                                getErrorHtml(errMsgs.errors[id]));
28                      }
29                  }
30              },
31              contentType: false,
32              processData: false
33          });
34      }
36      var elem = $("#" + id);
37      if (elem.attr('type') === 'file') {
38          // elemento di input type=file valorizzato
39          if (elem.val() !== '') {
40              inputVal = elem.get(0).files[0];
41          } else {
42              inputVal = new File([""], "");
43          }
44      } else {
45          // elemento di input type != file
46          inputVal = elem.val();
47      }
48      formElems = new FormData();
49      formElems.append(id, inputVal);
50      addFormToken();
51      sendAjaxReq();
53  }
```

In questo caso stiamo mostrando l'implementazione di doElemValidation() ossia il metodo che viene richiamato da onChange() e onBlur(). Questa funzione definisce un nuovo oggetto FormData() al quale vengono aggiunti, tramite il metodo append(), l'id e il valore dell'elemento di input sul quale si è verificato l'evento. Infine preleviamo il valore del token CSRF e definiamo la richiesta AJAX da eseguire affinchè mostri gli errori a livello grafico nel caso in cui ce ne fossero.

```
function getErrorHtml(elemErrors) {
    if ((typeof (elemErrors) === 'undefined') || (elemErrors.length < 1))
        return;

    var out = '<div class="invalid-feedback ms-2 errors">';

    if (elemErrors.length === 1) {
        // Se c'è solo un errore, aggiungi semplicemente il messaggio di errore a "out"
        out += elemErrors[0];
    } else if (elemErrors.length > 1) {
        // Se c'è più di un errore, crea una lista non ordinata di valori con gli errori
        out += '<ul>';
        for (var i = 0; i < elemErrors.length; i++) {
            out += '<li>' + elemErrors[i] + '</li>';
        }
        out += '</ul>';
    }

    out += '</div>'
    return out;
}
```

Questo metodo si occupa di generare l'elemento grafico da mostrare in caso di errore. Si tratta di una div che stampa a schermo l'errore o gli errori nel caso fossero più di uno. Le informazioni vengono visualizzate diversamente in base al numero di errori.

5.2 Homepage e ricerca delle promozioni

```
    public function index(Request $request)
    {
        $searchCompany = $request->input('company');
        $searchDescription = $request->input('description');
        $promozioniCarosello = null;

        if ($searchCompany || $searchDescription) {
            $query = Promozione::query();
            $query->where('visibile', true);

            if ($searchCompany) {
                $query->whereHas('azienda', function ($query) use ($searchCompany) {
                    $query->where('nome', 'LIKE', "%{$searchCompany}%");
                });
            }

            if ($searchDescription) {
                $query->where('fine', '>', Carbon::now())
                    ->where('inizio', '<=', Carbon::now())
                    ->where('descrizione', 'LIKE', "%{$searchDescription}%");
            }
```

```
23              $promozioniPaginated = $query->paginate(12);
24          } else {
25              $promozioniCarosello = Promozione::where('inizio', '<=',
                    Carbon::now())
26                  ->where('fine', '>', Carbon::now())
27                  ->where('visibile', true)
28                  ->get();

30              $promozioniPaginated = Promozione::where('fine', '>',
                    Carbon::now())
31                  ->where('inizio', '<=', Carbon::now())
32                  ->where('visibile', true)
33                  ->paginate(12);
34          }

36          return view('homepage', compact('promozioniCarosello', '
                promozioniPaginated'));
37      }
```

Questa funzione restituisce alla vista le infomazioni necessarie per poter gestire al meglio la visualizzazione degli elementi grafici nella pagina. Nello specifico controlliamo se almeno uno dei due campi per la ricerca contiene del testo, in caso di risultato positivo eseguiamo delle query al database con il testo fornito in input dall'utente, altrimenti mostriamo la homepage nel suo classico layout con un carosello contenente tutte le promozioni attive inserite nell'ultima settimana e una grid view paginata di tutte le promozioni attive.

5.3 Gestione delle rotte inserite manualmente

Per gestire l'eventuale inserimento manuale delle rotte da parte dell'utente per accedere a delle promozioni non attive abbiamo implementato una verifica nella seguente funzione che si occupa di restituire all'utente la vista con la promozione corrispondente:

```
1       public function promozione($id)
2       {
3           $promozione = Promozione::findOrFail($id);

5           if (Carbon::now()->isAfter($promozione->fine)) {
6               return view('expired_promozione');
7           } else if (Carbon::now()->isBefore($promozione->inizio)) {
8               return view('upcoming_promozione');
9           }

11          return view('promozione', compact('promozione'));
12      }
```

Nel caso in cui la promozione non sia attiva, perché non ancora iniziata o già scaduta, l'utente verrà reindirizzato verso una pagina che lo informa del perchè la promozione cercata non sia disponibile.

5.4 Gestione dell'input di tipo file

Normalmente con Laravel possiamo utilizzare il metodo old() per recuperare un valore di input precedentemente memorizzato nella sessione, questo non vale però per i campi di tipo file, ragion per cui abbiamo trovato una soluzione a questo problema inserendo nel codice queste due funzionalità:

```
1     if ($request->hasFile('immagine')) {
2             $file = $request->file('immagine');
3             $extension = $file->getClientOriginalExtension();
4             $fileName = Str::random(10) . '.' . $extension;
5             $file->move(public_path('images/promozioni'), $fileName);
6         } else {
7             $fileName = 'promozione.png';
8         }
```

Questa logica viene implementata all'atto della creazione di un nuovo oggetto sul database. Controlliamo se il campo di tipo file contiene il file di tipo m MIME tramite la funzione hasFile() (mostrando un messaggio d'errore se il file caricato non è di tipo MIME), se non viene caricato alcun file allora impostiamo un'immagine predefinita altrimenti salviamo all'interno del path public/images/{destinazione} il file caricato con un nome univoco.

```
1     if ($request->hasFile('immagine')) {
2             $file = $request->file('immagine');
3             $extension = $file->getClientOriginalExtension();
4             $fileName = Str::random(10) . '.' . $extension;
5             $file->move(public_path('images/promozioni'), $fileName);
6             if ($promo->immagine   file_exists(public_path('images/
                  promozioni/' . $promo->immagine))) {
7                 unlink(public_path('images/promozioni/' . $promo->
                      immagine));
8             }
9         } else {
10            $fileName = $promo->immagine;
11        }
```

Questa logica, invece, viene implementata all'atto della modifica di un oggetto sul database, in particolare controlliamo se il campo di tipo file ha un media caricato. Se l'esito è negativo allora carichiamo il file immagine che era stato inserito all'atto della creazione, altrimenti prendiamo l'oggetto file, lo salviamo nel path public/images/{destinazione} ed eliminiamo la vecchia immagine salvata all'interno del path in modo da evitare un sovraffollamento di immagini.