Care 2 Share - Dokumentation

Beschreibung der Funktionalitäten

Idee hinter Care2Share

Care2Share ermöglicht es den Nutzern Gegenstände / Räumlichkeiten und Dienstleistungen über eine zentrale Plattform zu leihen und zu verleihen.
Der Vorteil hierbei ist, dass Nutzer sowohl Geld sparen als auch die Umwelt schonen können, da für die Nutzung eines Gegenstandes dieser nichtmehr neu gekauft werden muss, sondern einfach von anderen Nutzern der Plattform bereitgestellt werden kann.
Außerdem können Privatpersonen oder gewerbliche Anbieter von Gegenständen die Plattform nutzen, um auf diesem Weg mögliche Mieter für die eingestellten Gegenstände zu finden.

Grundlegende Funktionen der Plattform

Neben den trivialen Funktionen wie der Registrierung / Anmeldung und Verwaltung der eigenen Nutzerdaten stehen folgende Funktionen zur Verfügung:

Für Standardnutzer

- Erstellung von Inseraten beliebiger Gegenstände
- Anfragen für vorhandene Inserate stellen /beantworten
- Anzeigen aller Anfragen
- Anfragenverwaltung für eigene Inserate
- Kontaktieren des Besitzers bei angenommener Anfrage
- Suchfunktion, die Ergebnisse in Echtzeit liefert
- Bestellhistorie über alle getätigen Geschäfte
- Reklamationsmanagement
- Nachrichten über Kontaktformular an Betreiber der Webseite schicken

Für Administratoren

- Nachrichtenverwaltung und Suche

- Verwaltung von Reklamationen - Annahme und Ablehnung
- Nutzerverwaltung - Datenbearbeitung
- Anzeige und Bearbeitung der Inserate aller Nutzer
- Anzeige und Bearbeitung aller Anfragen
- Anzeige der gesamten Bestellhistorie aller Nutzer

Wow-Faktoren der Anwendung

Revolutionäre Suchfunktion

Als ersten Wow-Faktor sehen wir die intuitiv zu bedienende Suchfunktion. Hierbei werden dem Nutzer Einträge in der Datenbank in Echtzeit als Vorschau dargestellt und stetig aktualisiert, falls der Nutzer Änderungen an den Eingabeparametern in der Suchleiste vornimmt. Als Eingabeparameter kann der Nutzer differenzieren nach :

- Dem Namen und den Tags des gewünschten Produkts/Dienstleistung
- Dem Preis mit Hilfe eines interaktiven Preisschiebers
- Einer der vom System vorgegebenen Produktkategorien

Durch einen einzigen Klick auf das "GO"-Feld wird der Nutzer direkt zur Anzeige der zur Suche passenden Artikel weitergeleitet, die dem Preis nach in absteigender Reihenfolge angezeigt werden. Jetzt hat der Nutzer die Gelegenheit seinen gewünschten Artikel aus der Liste gefundener Artikel auszuwählen. Durch einen weiteren Klick auf den gewünschten Artikel gelangt der Nutzer zur Anfrageseite und kann seine Anfrage an den Besitzer stellen.

Innovatives Reklamationsmanagement

Als zweiten Wow-Faktoren empfinden wir die Implementierung des Reklamationsmanagements. Beide Transaktionspartner eines abgeschlossenen Geschäfts (Hierzu zählen alle angenommenen Anfragen) können über die Bestellhistorie eine Liste aller abgewickelten Transaktionen anzeigen und mit einem Klick auf das jeweilige Geschäft einen Reklamationsfall eröffnen. Daraufhin muss lediglich ein Reklamationstyp ausgewählt werden und der Grund der Reklamation genauer spezifiziert werden. Nach der Eröffnung einer Reklamation erscheint diese beim Käufer/Verkäufer und dem Admin unter der Liste der offenen Reklamationen. Sobald der Admin die Reklamation bearbeitet hat ändert sich der Status der Reklamation in "Bestätigt" oder "Abgelehnt". Der Admin kann durch Einsicht der Kontaktdaten der Nutzer die betroffenen Parteien über das Ergebnis informieren und weitere Schritte einleiten.

Feingranulares Anfragenmanagement

Anfragen werden in drei unterschiedlichen Kategorien "offen", "abgelehnt", "bestätigt" angezeigt. Nur wenn der Besitzer eines Inserats die Anfrage bestätigt, kann der Anfragende die Kontaktdaten des Inseratseinstellers einsehen. Durch einen Klick öffnet sich automatisch das Mailprogramm des Anfragenden.

Technische Implementierung der Funktionen

Nutzerverwaltung

Die Nutzertabelle wurde um folgende Spalten erweitert:

- telefonnr
- straße
- houseNo
- PLZ
- isAdmin

Dementsprechend wurde die Validierung für die Eingabedaten sowohl im Frontend (auth/register.blade), als auch im Backend (RegisterController.php -> protected function validator) angepasst.
Jeder Administrator kann über den Link `/profile` alle Nutzer anzeigen und zumindest die Daten bearbeiten. Jeder Nutzer hat selbstverständlich die Möglichkeit, jegliche seiner Daten zu verändern. Dies wird über die Methoden index(), edit() und update() im ProfileController gesteuert.

Inseratsverwaltung

Erstellen

Zur Erstellung eines Inserats dienen die 3 blades : create.blade.php, create_2.blade.php sowie create_3.blade.php. Die ersten beiden Views packen per Session die Werte : "title", "description", "price", "prodCategory" und "tags" als Werte des Arrays in die Variable "advert". Der dritte View zeigt dem Nutzer noch einmal alle eingegebenen Werte an und erlaubt eine Bearbeitung dieser Werte. Durch einen Klick auf den "Bestätigen und Hochladen" Button wird im AdvertController dieStore Methode aufgerufen. In der Store Methode werden alle vorher in der Session gespeicherten Variablen in ein Advert Eloquent-Model gespeichert und dieses per save Methode des Modells auf die Datenbank geschrieben.

Bearbeiten und löschen

Über den Link `/adverts` wird die advert/index.blade.php aufgerufen, welche alle Inserate, die der Nutzer eingestellt hat anzeigt. Ist der Benutzer als Admin eingeloggt, so zeigt die Applikation alle im System registrierten Inserate an. Dies wird mit Hilfe einer if-Abfrage nach dem "isAdmin"-Flag des Nutzers realisiert.

```
if (User::where('id', Auth::id())->pluck('isAdmin')->first())
```

Diese Abfrage des Adminstatus nutzen wir in unserer gesamten Applikation in den Controllern der einzelnen Funktionen.
Aus der index Ansicht der Inserate kann der Benutzer per Direktclick sein Angebot bearbeiten oder löschen.

Hierzu wird entweder die adverts/edit.blade.php über die Methode edit aufgerufen oder die destroy Methode im AdvertController. Bei beiden Methoden wird überprüft, ob der eingeloggte Nutzer auch der tatsächliche Besitzer des Inserats ist.Die Authentifizierungsmethode ist hier Beispielhaft in der destroy Methode im AdvertController abgebildet und wird so ebenfalls in diversen anderen Controllern verwendet:

```
$advertOwner = Advert::where('advertId', [$id])->get()->pluck('ownerId')->first();
        //Admin oder passender User
        if ($advertOwner == Auth::id() || User::where('id', Auth::id())->pluck('isAdmin')->first(
)) {
            DB::delete('delete from advert where advertId=?', [$id]);
            $request->session()->flash('alert-info', 'Artikel ' . $id . ' wurde gelöscht');
            return redirect()->action('AdvertController@index');
        } else {
            $request->session()->flash('alert-danger', 'Sie sind nicht Besitzer der Anzeige, Lösc
hen nicht möglich');
            return redirect()->action('AdvertController@index');
        }
```

In der adverts/edit.blade.php kann der Nutzer per Formular die bisherigen Werte für sein Inserat bearbeiten. Diese werden über den Controller über die edit Methode der View übergeben. Auch hier findet wieder eine Authentifkationsprüfung statt. Hat der Nutzer seine validen Änderungen wie gewünscht ins Formular geschrieben kann er über anklicken des "Update" Buttons seine Änderungen speichern. HIerbei wird die update Methode aufgerufen.

Anfragenverwaltung

Erstellen/Löschen

Die Erstellung von Anfragen setzt das Vorhandensein eines Inserats voraus. Über `adverts/{id}/show` werden dem Nutzer Informationen über die Anzeige mit der über die URL übergebene ID angezeigt. Nur wenn der Nutzer nicht Besitzer des o.g. Inserats ist, wird ihm der Button “Anfrage schicken” angezeigt. Ansonsten kann er die edit Funktion über den Button “Bearbeiten” aufrufen. Durch klicken auf den “Anfrage schicken” Button gelangt der Nutzer auf die inquiries/create.blade.php. Dort werden ihm erneut Details zum Produkt angezeigt und er kann eine Anfragebotschaft formulieren. Durch das klicken der “Anfrage senden” wird im InquiryController die store Methode aufgerufen, die eine neue Anfrage mit dem Statusflag 0 (ausstehend) erstellt.
Gelöscht wird ein Inserat über die URL `/inquiries/{id}/destroy` (welche erreichbar ist über den Button “Anfrage löschen” in der Übersicht aller Anfragen in der inquiries/index.blade.php) . Hier wird die destroy Methode im InquiryController aufgerufen.

Annehmen/Ablehnen

Hat ein Nutzer für ein Inserat eine Anfrage gestellt, wird diese dem Besitzer über die inquiries/mine.blade.php angezeigt. Dieser hat nun die Möglichkeit den Anfragetext zu lesen und durch die buttons “Ablehnen” und

“Annehmen” die reject bzw. confirm Methode im InquiryController aufzurufen. Diese funktionieren prinzipiell gleich, mit dem Unterschied, dass ein unterschiedliches Statusflag gesetzt wird und beim confirm ein neuer History-Eintrag erstellt wird.
Auszug der confirm-Methode:

```
        $inquiry=Inquiry::where('inquiryId',$id)->get()->first();
        $advert=Advert::where('advertId',$inquiry->advertId)->get()->first();
        $advertOwner=Inquiry::where('ownerId',Auth::id())->where('inquiryId',$id)->get()->pluck('
ownerId')->first();
        if($advertOwner==Auth::id()||User::where('id',Auth::id())->pluck('isAdmin')->first()){
            DB::update('UPDATE `inquiry` SET `statusflag` = \'1\' WHERE `inquiry`.`inquiryId` = ?'
,[$id]);
            $request->session()->flash('alert-success','Anfrage angenommen');
            //nach Bestätigung einer Anfrage wird Eintrag in Historie angelegt

            $history=new History();
            $history->title=$advert->title;
            $history->description=$advert->description;
            $history->sellerId=$inquiry->ownerId;
            $history->buyerId=$inquiry->buyerId;
            $created_at=now();
            $updated_at=now();
            $history->save();
            return redirect()->action('InquiryController@mine');
        }else{
            $request->session()->flash('alert-danger','Annahme war nicht möglich, sie sind nicht
Besitzer dieser Anfrage');
            return redirect()->action('InquiryController@mine');
        }
```

Anzeigen

Alle Anzeigen können über die entsprechenden Links `inquiries/denied` und `inquiries/requested`, sowie `inquiries/confirmed` angezeigt werden. Dabei werden für jede blade die Anfragen mit dem entsprechenden Statusflag vom InquiryController mitgegeben.Der Admin hat die Möglichkeit über `/inquiries/all` alle Anfragen einzusehen.

Besitzer kontaktieren

Wurde eine Anfrage angenommen, kann der Anfragende dies in der inquiries/confirmed blade einsehen. Dort kann er über klicken des “Besitzer kontakieren” Buttons die Email, sowie die Telefonnummer des Besitzers einsehen. Wenn er die Email-Adresse direkt anklickt, wird automatisch das Mail-Programm des Nutzers angezeigt.

Artikel suchen

Dies ist das Herzstück der Applikation. Auf der Homepage kann der Benutzer ein Eingabeformular ausfüllen. Die drei Eingabeparameter sind ein Suchbegriff, eine Produktkategorie und ein maximaler Preis, den der Nutzer bereit ist, für sein gesuchten Gegenstand zu bezahlen.
Für jegliche Änderungen an den Eingabeparametern sind Eventfunktionen in AJAX definiert, die automatisch Suchergebnisse, die auf die derzeitige Auswahl passen, anzeigen.
Die Methoden funktionieren alle sehr ähnlich. Der einzige Unterschied ist die ID und der Eventname.
Beispielhaft ist die Anpassung bei Änderung des Suchbegriffs.

```
    $('#search').on('keyup',function(){
    $value=$('#search').val();
    $price=$('#price').val();
    $category=$('#category').val();
    $.ajax({
        type : 'get',
        url : '/search',
        data:{'search':$value,'price':$price,'category':$category}, //dem AJAX-Call als Parameter
 übergeben werden
        success:function(data){
            $('tbody').html(data);
        }
    });
});
```

Der AJAX-Call ruft im SearchController die search-Methode auf. Es werden die Artikel, die zu den Suchparametern des AJAX-Calls passen, aufgerufen und als HTML-Code zurückgegeben.

```
public function search(Request $request)
    {
        if ($request->ajax()) {
            $output = "";
            $categoryNumber=ProductCategory::where('description',$request->category)->pluck('prod
uctCategory')->first();
            $search=$request->search;
            $adverts = DB::table('advert')
                ->where(function ($query) use ($search){
                    $query->orWhere('title','like','%'.$search.'%')
                        ->orWhere('description','like','%'.$search.'%')
                        ->orWhere('tags','like','%'.$search.'%');

                })
                ->where('prodCategory','=',$categoryNumber)
                ->where('price','<=',$request->price)
                ->orderBy('price')
                ->get();
            if ($adverts) {
                foreach ($adverts as $key => $advert) {
                    $output .= '<tr>' .
                        '<td>' . $advert->title . '</td>' .
```

```
                    '<td>' . $advert->description . '</td>' .
                    '<td>' . $advert->price . '</td>' .
                    '</tr>';
            }
            return Response($output);       //html Response
        }
    }
}
```

Reklamationsverwaltung

Alle abgeschlossenen Transaktionen (Anfrage wurde angenommen) werden in einer separaten Tabelle history in der Datanbank gespeichert. So ist gewährleistet, dass auch nach dem Löschen einer Anzeige immernoch eine Reklamation erstellt werden kann. Über einen Link in der history/index.blade.php kann der Reklamationsprozess eingeleitet werden. Über ein Formular kann man nun Kategorie und genaue Beschreibung des Fehlers abschicken. All dies wird über den ReclamationController koordiniert.
Um die Reklamationen anzuzeigen werden die views reclamations/index.blade.php und reclamaations/againstMe.blade.php verwendet. Hier werden zum einen die Reklamationsfälle als Käufer und als Verkäufer abgebildet.
Der Admin kann sich alle Reklamationsfälle über die reclamations/index.blade.php anzeigen lassen und durch klicken der entsprechenden Icons das Statusflag der jeweiligen Reklamation mit Hilfe des Aufrufs der Methoden confirm und reject verändern.

Kontaktaufnahme/Nachrichtenverwaltung

Über das Kontaktformular, das unter `/kontakt` ausfüllbar ist, kann jeder beliebige Nutzer (auch nicht registrierte Nutzer) Kontakt zu den Betreibern der Homepage bzw. Administratoren aufnehmen. Dies wird durch den Aufruf der store()-Methode erreicht.

Als Admin kann man sich die Nachrichten über `/messages` anzeigen lassen und in Echtzeit werden die angezeigten Ergebnisse je nach Suchparameter aktualisiert. In der messages/show.blade.php läuft ein javascript, das sobald ein Input in das Formularfeld eingegeben wird, einen AJAX-Call ausführt, der die search()-Methode im MessageController ausführt. Diese transformiert die Eloquent results der DB-Abfrage in ein JSON-Objekt und gibt dieses zurück. Danach wird die Tabelle mit Ergebnissen dynamisch durch Zugriff auf die JSON-Variablen erzeugt.

JS-Code:

```
    <script>
        var original = $('#dynamic-row').html();
        $('body').on('keyup', '#search-message', function () {
            var searchQuest = $(this).val();
            if(searchQuest!='') {

                $.ajax({
                    method: 'POST',
                    url: '{{route("searchmessage")}}',
```

```
                    dataType: 'json',
                    data: {
                        '_token': '{{ csrf_token() }}',
                        searchQuest: searchQuest,
                    },
                    success: function (res) {
                        var tableRow = '';
                        console.log(res);

                        $('#dynamic-row').html('');

                        $.each(res, function (index, message) {
                            tableRow = '<tr><td>' + message.content + '</td><td>' + message.userI
d + '</td><td>' + message.created_at + '</td><td><a class="btn btn-danger" href="/messages/'+mess
age.id+'/destroy">Löschen</a></td></tr>';

                            $('#dynamic-row').append(tableRow);
                        });

                    }
                });
            }else{
                $('#dynamic-row').html(original);
            }
        });
        $('input[type=search]').on('search',function(){
            $('#dynamic-row').html(original);
        });
    </script>
```

search()-Methode

```
      public function search(Request $request){

        $messages=Messages::where('content','like','%' . $request->get('searchQuest') . '%')->get
();

        return json_encode($messages);
    }
```

SEO

Keyword-Kombinationen

Wir haben uns folgende Keywords überlegt, unter denen wir gefunden werden wollen. Diese haben wir auf unserer Homepage eingebettet.

- Care2Share

- Leihplattform
- Artikel leihen
- Geld sparen
- wiederverwenden mieten
- Lösung Problem
- nachhaltig, Zukunft