Webová aplikácia Food Sharing

(Dokumentácia)

1. Popis úlohy

I. Scenário – opis problému

Množstvo ľudí dennodenne vyhadzuje kvantá potravín, kvôli ich prekročenej dobe spotreby, prameniacej z nadbytočného kupovania potravín, prípadne si niektorí sami vyestujú viac než spotrebujú. Odpoveďou na takého nevyvážené vlastníctvo potravín by bola výmena potravín na úrovni jednotlivca, ktorá však vyžaduje nejakého sprostredkovateľa. V súčasnoti je častým riešením vytváranie komunitných skupín na sociálnych sieťach, čo má avšak množstvo nevýhod, predovšetkých neprehľadnosť ponúk, či žiadne predpísané pravidlá vo výmenách, chýbanie možností uložiť si ponuky, či dať na javo záujem o ponuku.

II. Riešenie

Navrhnuté riešenie spočíva vo webovej aplikácií, ktorá by prepájala ‘predajcov‘ (= osoby ponúkajúce komodity) s ‘kupcami’ (= osoby poberajúce ponuky).
Predajca môže ľahko vytvárať ponuky, kupec ľahko odpovedať, pričom užívateľ musí byť schopní naraz zaujať zaujať rolu predajcu aj kupcu (pre rôzne ponuky).

III. Zadanie – prevedenie riešenia

Akceptačné podmienky:

1. Užívateľ sa vie zaregistrovať.
2. Užívateľ sa vie prihlásiť.
3. Užívateľ vie vytvárať ponuky,
    a. pričom nemusí vypĺňať zložité formuláre.
4. Užívateľ vie kontaktovať predajcu.
5. Užívateľ si prehľadne vidí ponuky.
6. Myšlienka šetrenia potravinami je efektne predaná užívateľovi.
7. Užívateľ si vie pozrieť svoje ponuky.

8. Užívateľ si vie zhromažďovať ponuky, ktoré ho zaujali.
9. Užívateľ vie filtrovať ponuky.
10. Užívateľ sa vie odhlásiť.
11. Užívateľ si vie personalizovať vzhľad pre lepší ‘user experience’.
12. Užívateľ je povinný splniť formát vstupných parametrov.
13. Užívateľ nie je schopný zareagovať na svoj vlastný príspevok.

IV. Spôsob prevedenia zadania

Prevedenie bude realizované PHP webovou aplikáciou, s využitím frameworku Laravel, s využitím MySQL databázy.
Aplikácia bude deploynutá na Heroku server.

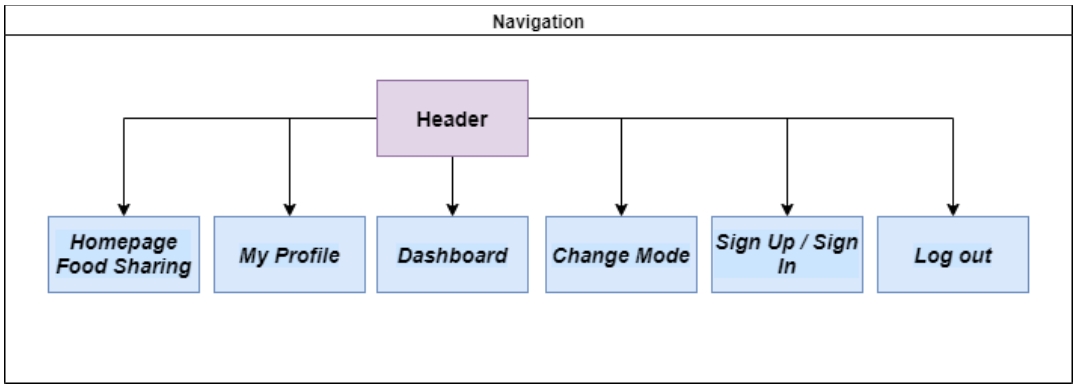
2. Užívateľská príručka

I. Popis funkčnosti webu – Manuál

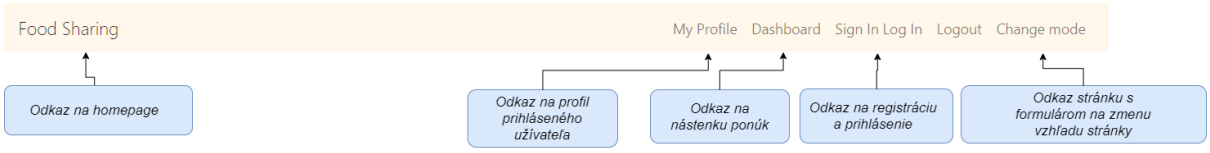
Jednotlivé stránky zdieľajú rovnaký základ štruktúry:

- header
- kontent
- footer

Keďže header je prítomný na každej stránke, užívateľ sa vie dostať kamkoľvek na jeden klik:



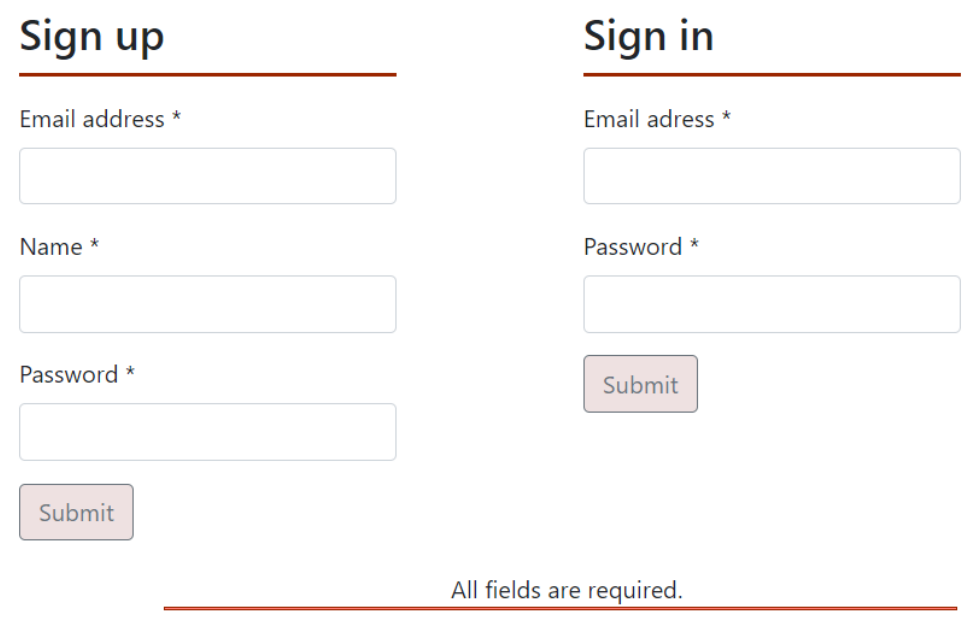
Obr.: Schéma headeru



Obr.: Header samotný, s popisom

Obr.: Homepage.

Na hlavnej stránke čaká užívateľa uvítací text. Pod ním má k dispozícii buttony na registráciu a prihlásenie.



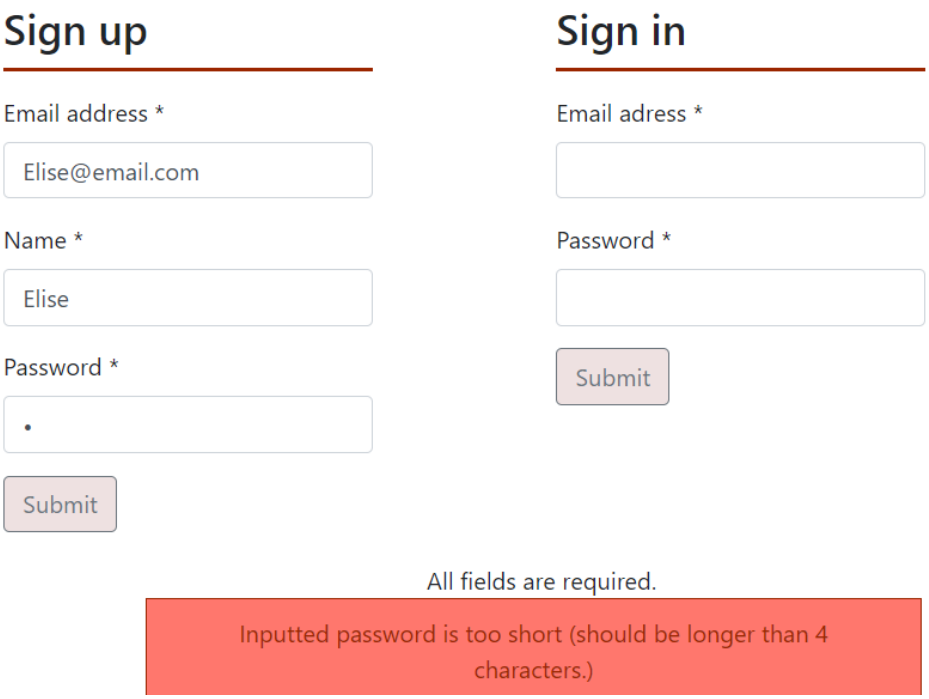
Obr.: Formuláre na registráciu a prihlásenie.

Formuláre na registráciu a prihlásenie vyžadujú od užívateľa email, meno a heslo, s tým, že musia byť splenené nasledujúce podmienky:

- všetky vstupné polia musia byť vyplnené

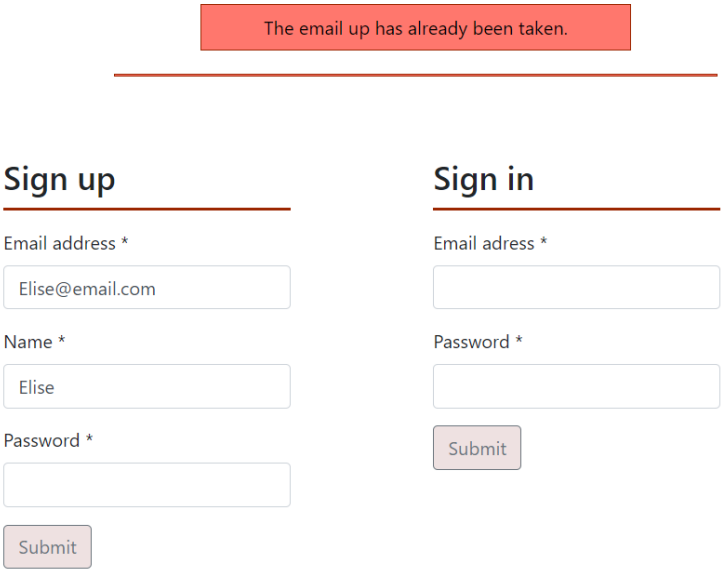
- heslo musí byť dlhšie ako 4 znaky
- email musí byť dlhší ako 4 znaky
- email musí obsahovať ‘@‘ znak

Pri nedodržaní týchto podmienok, validácia vstupných údajov javascriptom vyhodí chybu s popisom vzniknutej chyby.



Obr.: Chyba vyhodená javascriptom pri validacii registrácie.

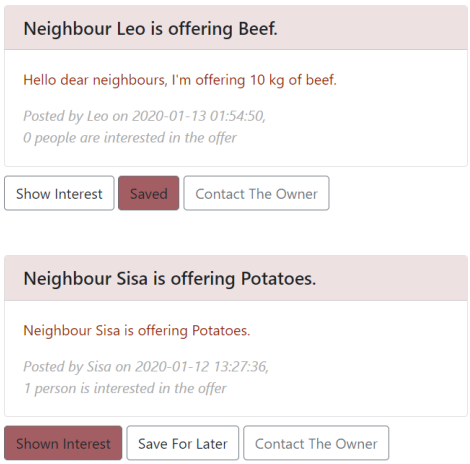
Na strane servera sa naviac overuje napríklad aj unikátnosť emailovej adresy:



Obr.: Chyba vyhodená serverom pri validacii registrácie.

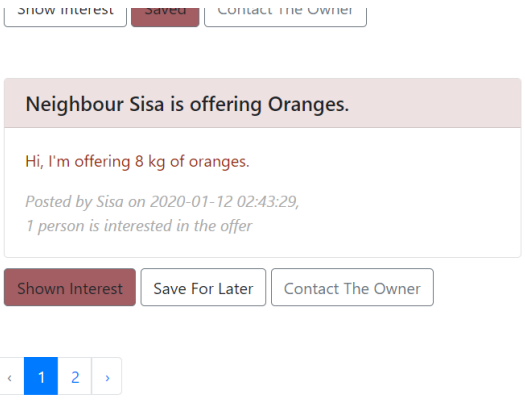
Po úspešnej validácii je novovytvorený user prihlásený a presmerovaný na nástenku s ponukami. Jednotlivé ponuky obsahujú:

- názov
- obsah ponuky
- sprievodné informácie
    - dátum pridania
    - počet ľudí, ktorí vyjadrili záujem
- buttony na:
    - vyjadrenie záujmu
    - uloženie ponuky na neskôr
    - kontaktovanie predajcu

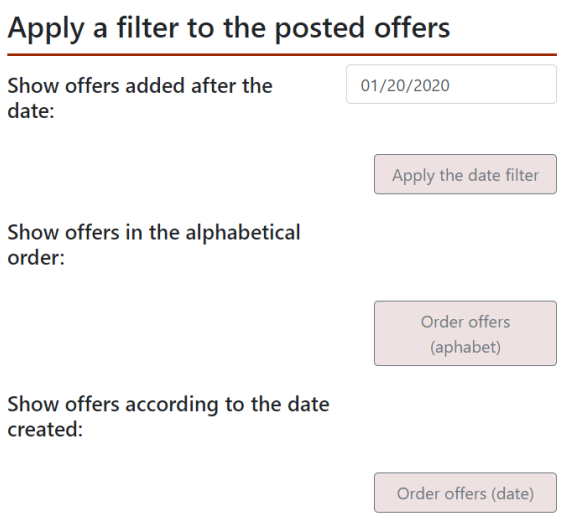


Obr.: Nástenka s ponukami.

Pri väčšom množstve ponúk sa zobrazujú iba po 5 kusoch, pričom sa dá prepínať medzi jednotlivými stránkami:

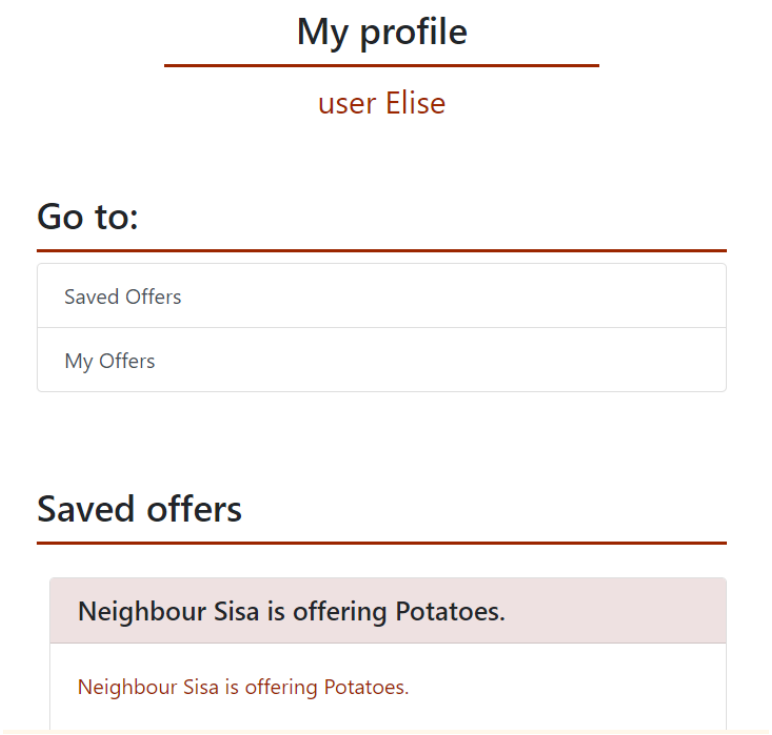


Na prehľadnejšie zobrazenie ponúk má užívateľ k dispozícii filtrovanie a zoradenie podľa abecedy a dátumu pridania.



Obr.: Filtrovanie ponúk.

V sekcii My Profile si užívateľ môže pozrieť sebou uložené ponuky a svoje príspevky.

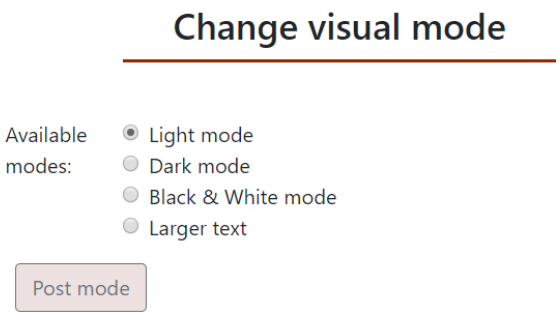


Obr.: Sekcia My Profile.

Sekcia Change mode ponúka zmenu vzhľadu stránky, s tým, že zmena sa zachováva po dobu trvania celej session:

- light mode
- dark mode
- larger mode

- black & white mode



Obr.: Zmena vzhľadu stránky

3. Popis implementácie

popis hlavních rysů programu (funkčnost hlavních skriptů)

- Popis architektury, obsluha formulářů, zabezpečení, algoritmy zpracování dat, UML, popis využití frameworků.

I. Popis architektúry

Webová aplikácia využíva architektúru MVC, teda Model-View-Controller, čo podmienilo aj štruktúrovanie priečinkov a zložiek.

View je úplne separované od logiky, premenné sú do view predávané cez premennú Request $request, ktorá je POST requestom predaná serveru, accesovaná v Controllery, a nesie v sebe užívateľom zadané hodnoty z formulárov. Hodnoty sa ukladajú do MySQL databázy, tak, že databázové tabuľky korešpondujú s modelami.

Model

Model predstavuje dátové objekty, kontrétne sú to:

- Model Interest
  - s atribútami:
    - id
    - created_at - dátum pridania
    - user_id
    - offer_id
- Model Offer
  - s atribútami:
    - id
    - created_at - dátum pridania

        - commodity = meno ponúkanej komodity, potraviny
        - body = text ponuky
        - user_id
        - num_of_interested = počet ľudí, ktorí zaklikli ‘Show interest‘ na danej ponuke
- Model Saved
    - s atribútami:
        - id
        - created_at - dátum pridania
        - user_id
        - offer_id
- Model User
    - s atribútami:
        - id
        - created_at - dátum pridania
        - email
        - name
        - password
        - rememberToken (required)

Controller

Controller je členený do 4 súborov, pričom každý má na starosti rôzne entity:

- InteractionController
    - spracúva POST requesty (ukladanie do databázy nového interestu ohľadom konktrétnej ponuky konktrétnym userom) a GET requesty týkajúce sa Show interest a Save offer buttonov zavolané z View.
- OfferController
    - má na starosti requesty týkajúce sa vytvárania novej ponuky, editovania ponuky, filtrovania ponúk, zoraďovania ponúk, rendrovanie view s posunutím ďalej správnych dát, t.j. ponúk
- SettingsController
    - má na starosti requesty ohľadom menenia vzhľadu stránky
- UserController
    - má na starosti requesty regitrácie, prihlasovanie, odhlasovania

View

Súbory reprezentujúce view sú rozdelené v priečinkoch ‘includes‘ a ‘layouts‘. Vo view je využitý Blade web template system, ktorý umožňuje cez príkazy @include() a @yield() znovupoužívať značné časti kódu.

Príkaz @include() bol využitý na inkludovanie headeru, footeru, css súborov, informatívnych správ; @yield() na zase na obsah stránok.

II. Obsluha formulárov

Každé jedno pole každého formuláru prechádza validáciou javascriptom. Overujú sa rôzne podmienky pre rôzne polia, ako už bolo spomenuté vyššie, napriklad dĺžka hesla, či prítomnosť @ v emaili.

Po neúspešnej validácii javascriptom dáta nie sú odoslané na server, sú znovupredané do polí a zobrazené užívateľovi na opravu. Požadovanú zmenu mu hlása červený error message.

Po úspešnej validácii javascriptom sa odošle POST request na server. Na serveri sa ďalej validuje, dĺžka inputov, ale aj unikátnosť emailu kontrolou predošlých inputov uložených v databáze.

III. Zabezpečenie

Heslo je ukladané až po prejdení funkciou bcrypt(), teda bcrypt($request['password']), ktorá okrem zahashovania hesla pridáva k heslo randomne generovanú ‘soľ’ za účelom zvýšenia bezpečnosti, taktiež ako aj určitý ‘cost factor’. Až takto zmenené heslo je uložené do databázy.

Ďalej, pri pokuse o accessnutie nástenky typnutím kontrétnej URL vedúcej na nástenku, server preveruje, či tak robí už autentifikovaný user (autentifikovaný user je už dostupný vrámci Illuminate\Http\Request objektu, použitím Auth facade máme na to dostupnú funkciu check()), alebo sa niekto pokúša obísť naše zabezpečenie.

IV. Využitie Frameworku

Pri developovaní aplikácie bol použitý framework Laravel, verzia 6.10.1, čo umožňovalo využívanie Blade templating engine a Blade príkazov, napríklad aj konštrukcie if-else; využívanie Auth middleware, helper methods spolu s Eloquent Object Relational Mapping (Offer::orderBy) a facades (Illuminate\Support\Facades\Auth), getovanie stavov modelov z databázy, čo docielilo prehľadneší kód, či migračný systém na vytváranie tabuliek.

Využívanie JQuery na jednoduché getovanie elementov vo formulári, na ktorých sa vykonávala validácia.

V. Spracovanie dát

Dátu (ako pri sign up a log in formulároch) sú po úspešnej validácii javascriptom POST requestom odoslané na server. Na serverovej strane sú overené po prevzatí z View, checkuje sa, či premenná nie je prázdna, ale takisto aj, či nezlyhal zápis do databázy (cez Auth:attemt()).

Dáta sa getujú z databázy cez queries ako: $saved_offers = Offer::whereIn('id', $saved_off_ids_arr)->get(); , čo umožňuje široký záber zadávania podmienok na getovanie.

VI. Triedy a ich funkcie:

class InteractionController

- public function postInterestUpdated(Request $request)

- public function postInterestLoad(Request $request)
- public function postSavedLoad(Request $request)
- public function postSavedUpdated(Request $request)
- public function getContact($offer_id)

class OfferController

- public function getDashboard()
- public function getDashboardWoJS()
- public function postCreateOffer(Request $request)
- public function getDeleteOffer($offer_id)
- public function postEditOffer(Request $request)
- public function postFilterDate(Request $request)
- public function postOrderAlphabet(Request $request)
- public function postOrderDate(Request $request)

class SettingsController

- public function getMode()
- public function postMode(Request $request)

class UserController

- public function postSignUp(Request $request)
- public function postSignIn(Request $request)
- public function getLogOut()
- public function getProfile()

Javascript funkcie

- $('a[href$="#edit-offer"]').on("click", function(event)
- $('#edit-save').on('click', function()
- $('.interest_button').on('click', function(event)
- $('.saved_button').on('click', function(event)
- if($("#load_interaction_buttons").length > 0)

- function load_validation()
- function load_interested(){
- function load_saved()
- function mark_as_saved(element)
- function remove_saved(element)
- function mark_as_interested(element)
- function remove_interest(element)

  + funkcie z validation.js

4. Popis úložiska dát

Ako úložisko slúži ClearDB MySQL databáza hostovaná na Heroku, bežiaca ako doplnková služba.

	Table	Action	Rows	Type	Collation	Size	Overhead
☐	interests	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_unicode_ci	16 KiB	-
☐	migrations	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_unicode_ci	16 KiB	-
☐	offers	Browse Structure Search Insert Empty Drop	9	InnoDB	utf8mb4_unicode_ci	16 KiB	-
☐	saveds	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_unicode_ci	16 KiB	-
☐	users	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_unicode_ci	16 KiB	-
	5 tables	Sum	30	InnoDB	utf8mb4_general_ci	80 KiB	0 B

Obr.: Tabulky (screenshot z databázy bežiacej ešte len lokálne, nie z ClearDB MySQL).

Dáta sú uložené v 4 tabuľkách: interests, offers, saveds, users, generovaná na základe migration súborov.

```
public function up()
{
    Schema::create( table: 'offers', function (Blueprint $table) {
        $table->bigIncrements( column: 'id');
        $table->timestamps();
        $table->text( column: 'commodity');
        $table->text( column: 'body');
        $table->integer( column: 'user_id');
        $table->integer( column: 'num_of_interested');
    });
}
```

Obr.: Migration súbor pre Model Offer

Na základe migration súbore pre Model Offer sa vytvorí tabuľka so stĺpcami ‘id‘, ‘created_at‘, ‘updated_at‘, ‘commodity‘, ‘body’, ‘user_id‘, ‘num_of_interests‘

Jednotlivé vzťahy medzi modelmi sú udržiavané v databáze, napríklad na modeli Offer môžeme vidieť, ako atribút ‘user_id’ tabuľky Offer odkazuje na inštanciu modela User. Tieto vzťahy sú definované v jednotlivých modeloch nasledovne:

- Interest belongs to User
- Interest belongs to Offer
- Offer belongs to User
- Offer has many Interests
- Offer has many Saveds
- Saved belongs to Offer
- Saved belongs to User
- User has many Offers
- User has many Interests
- User has many Saveds

5. Poznámky ku klientovi a serveru

I. Escaping

Laravel Blade templating engine implicitne escapuje HTML entity vo vnútri zátvoriek {{ }}

(source)

II. Neztrácí se uživatelem vyplněné hodnoty

je zaručené použitím:

```
value="{{Request::old('commodity')}}"
```

III. Stránka použitelná i bez JavaScriptu

IV. Skiny

Defaultne je nastavený light mode. Ak užívateľ zvolí iný mód, zapíše sa táto voľba do dát nesených v session:

```
$request->session()->put('mode', $mode);
```

V. Zaujímavosť v CSS

Využitie Bootstrap open source CSS frameworku, napríklad na navigation bar v headeri, a predovšetkých na buttony, či docielenie vysokej miery responzivity použitím klás ako ‘col-md-6’ pre pre stredne veľké zariadenia.

```
<nav class="navbar navbar-expand-lg navbar-light">
    <a class="navbar-brand display-2" href="{{ route('homepage') }}">Food Sharing</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupport
        <span class="navbar-toggler-icon"></span>
    </button>

    <div class="navbar-nav ml-auto" id="navbarSupportedContent">
        <ul class="nav navbar-nav navbar-right">
            <li class="nav-item">
                <a class="nav-link" href="{{ route('profile')  }}">My Profile</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="{{ route('dashboard')  }}">Dashboard</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="{{ route('home') }}">Sign In Log In</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="{{ route('logout') }}">Logout</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="{{ route('mode') }}">Change mode</a>
            </li>
        </ul>
    </div>
</nav>
```

```
<button type="submit" class="col-md-4 offset-8 btn btn-outline-secondary submit_button">Order offers (date)</button>
```

Obr.: Ukážky využitia Boostrapu

VI. Zaujímavosť

Odkazovanie nezávislé na absolútnej, docielené využitím feature Laravelu, funkcie route() z Facade Route. Ku každému linku sa priradí meno. Následne stačí využívať len tento alias.

```
var urlEdit = '{{ route('edit') }}';
```

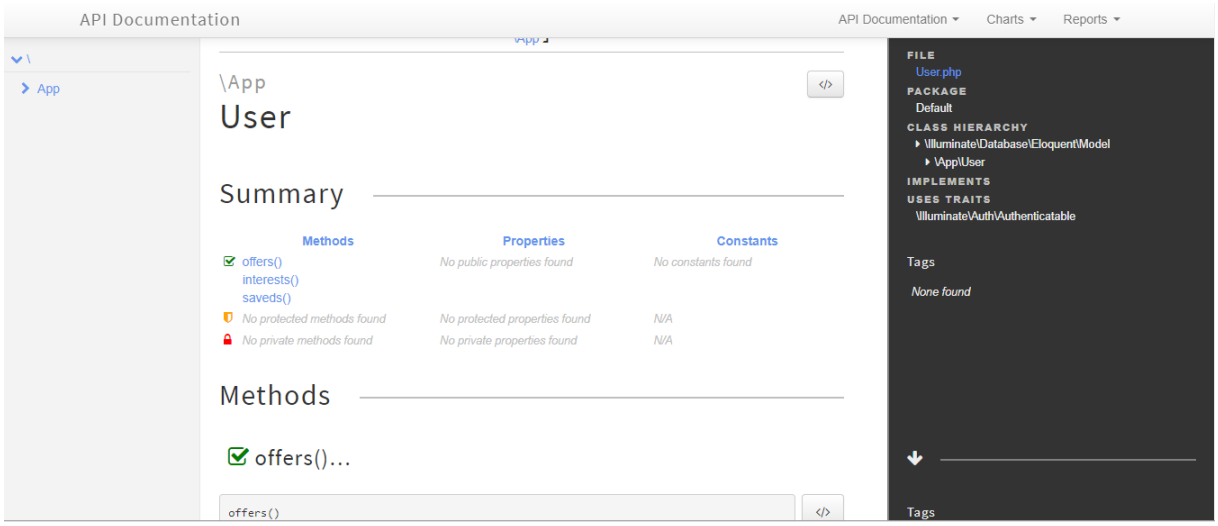
```
Route::post( uri: '/edit', [
    'uses' => 'OfferController@postEditOffer',
    'as' => 'edit'
]);
```

Obr.: Odkazovanie

6. Dokumentácia

Kód bol zdokumentovaný, okrem poznámok v kóde, použitím automaticky generovanej dokumentácie zo zdrojového kódu od Php Documentor.

Dokumentácia sa nachádza v priečinku docs.



Obr.: Dokumentácia

7. Splnenie podmienok – Evaluácia

Reflexia na akceptačné podmienky:

1. **Užívateľ sa vie zaregistrovať.**
    - áno
2. **Užívateľ sa vie prihlásiť.**
    - áno
3. **Užívateľ vie vytvárať ponuky,**
    - **pričom nemusí vypĺňať zložité formuláre.**
        i. áno
4. **Užívateľ vie kontaktovať predajcu.**
    - cez kontakt button
5. **Užívateľ si prehľadne vidí ponuky.**
    - vie ich filtrovať
    - stránkovanie
6. **Myšlienka šetrenia potravinami je efektne predaná užívateľovi.**
    - homepage dizajn
7. **Užívateľ si vie pozrieť svoje ponuky.**
    - v sekcií My Profile
8. **Užívateľ si vie zhromažďovať ponuky, ktoré ho zaujali.**
    - využitím Save buttona a sekcie My Profile
9. **Užívateľ vie filtrovať ponuky.**
    - využitím filter (nie s pagination)
10. **Užívateľ sa vie odhlásiť.**
    - využitím Log Out
11. **Užívateľ si vie personalizovať vzhľad pre lepší ‘user experience’.**
    - cez dostupné módy
12. **Užívateľ je povinný splniť formát vstupných parametrov.**

- validávia na strane klienta aj servera

13. Užívateľ nie je schopný zareagovať na svoj vlastný príspevok.

- buttonu na interakciu sa zobrazia na základe toho, či je užívateľ autorom ponuky (edit, delete), alebo nie (Save, Show interest)

Splnené