## The name of the company:

myPet

## Motto:

Find the animal of your dreams with us

## Names of all team members and IDs:


1.**Nazrulla Sadullaev**

**ID:** u1610167

**Section:** 004

2.**Azimjon Akhmadov**

**ID:** u1610032

**Section:** 004

3.**Oybek Fayziev**

**ID:** u1610178

**Section:** 004

4.**Nodirbek Nabiev**

**ID:** u1610169

**Section:** 004


## LINK TO GITHUB REPO:

https://github.com/iuthub/design-project-dreamteam

## LINK TO WEB SITE:

https://mypet.uz

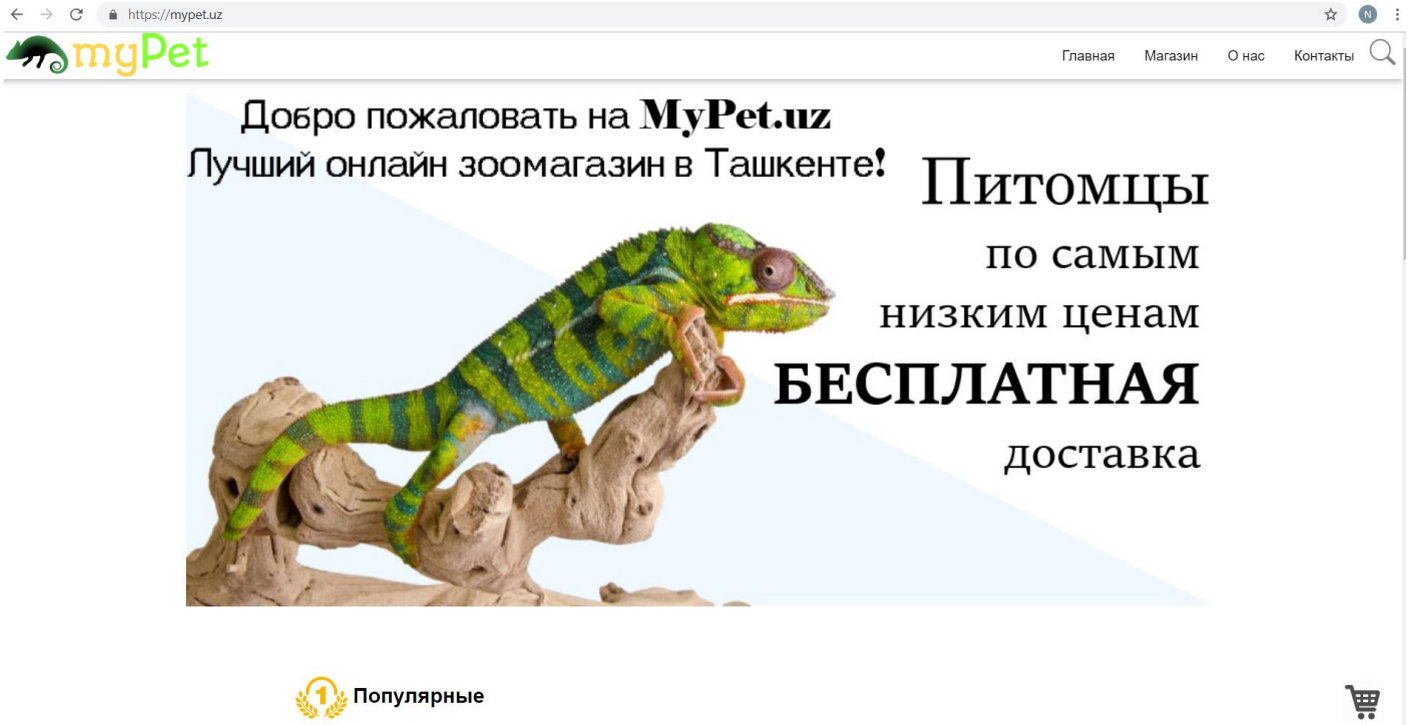
## The description of the company:

MyPet.uz company is a brand new online animal store, that has a variety of pets for any customer. The main goal of the company is to sell different kinds of animal to the public. The animal can be delivered from Russian or they can be of the local origin. Our team is the main maintainer of the website, we follow the orders that are placed through the official email mypet.uz@mail.ru. We try to satisfy the order within a day and don't bear any responsibility after the selling has been made. Any user can enter our website and start shopping immediately, no waste of time for the user due to the absence of the registration. User can enter the website and immediate add something to the cart and later on the checkout, an email is sent to mypet.uz@mail.ru and is processed within a day by our team. This makes the business process much easier and comfortable. There is a free delivery option for those, who make an order for more than 200000 sums and paid delivery for order that are less. There is also a private stock house for some types of animals at Azimjon Akhmadov's place. Depending on the order the animal is delivered from Azimjon's house or from a retailer of the company that supplies animals from abroad (Russia) or from Uzbekistan. It has been a little over two weeks since the website started functioning and our team has already made some profit by selling some rare exotic animal that we have in-place. MyPet.uz is expected to become one of the most successful online animal stores in Uzbekistan.

# Screenshots of the features:

**Azimjon Akhmadov, u1610032**

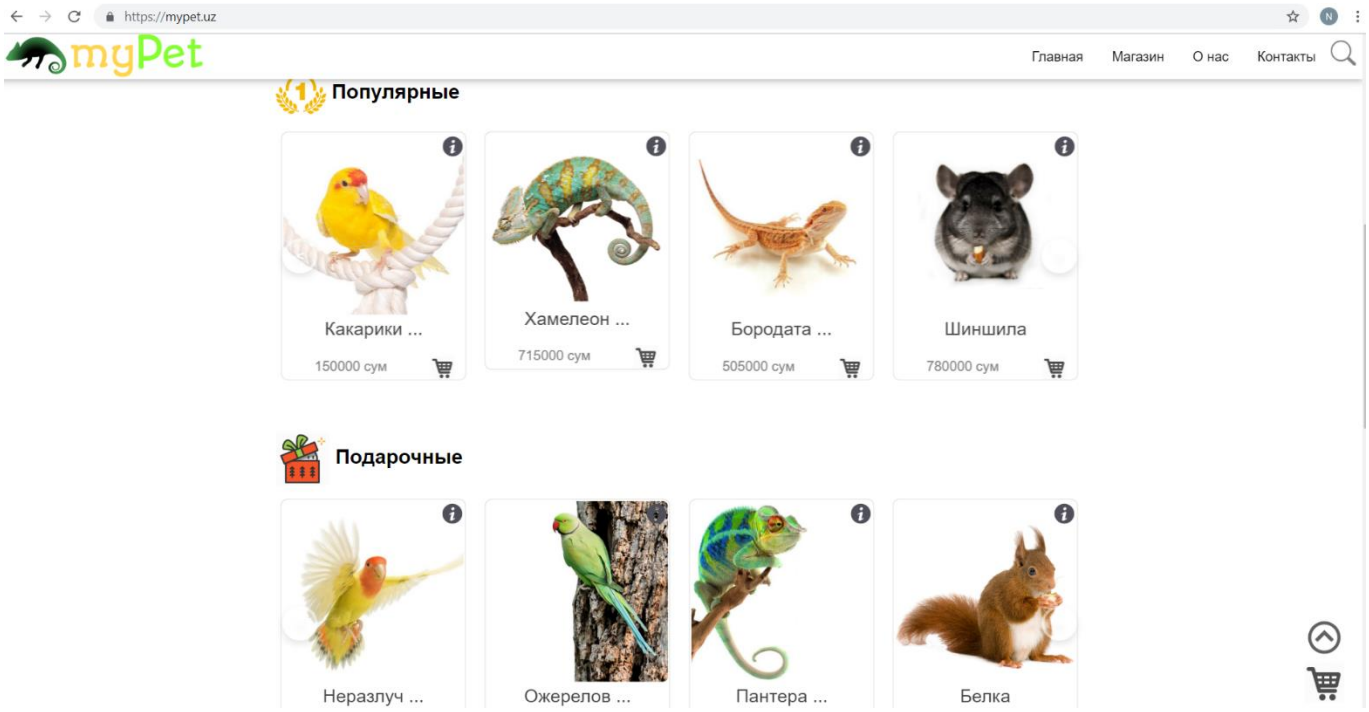
The things that I did were the front-end functionalities included in Laravel for landing page, shop page, contact page and about us page.



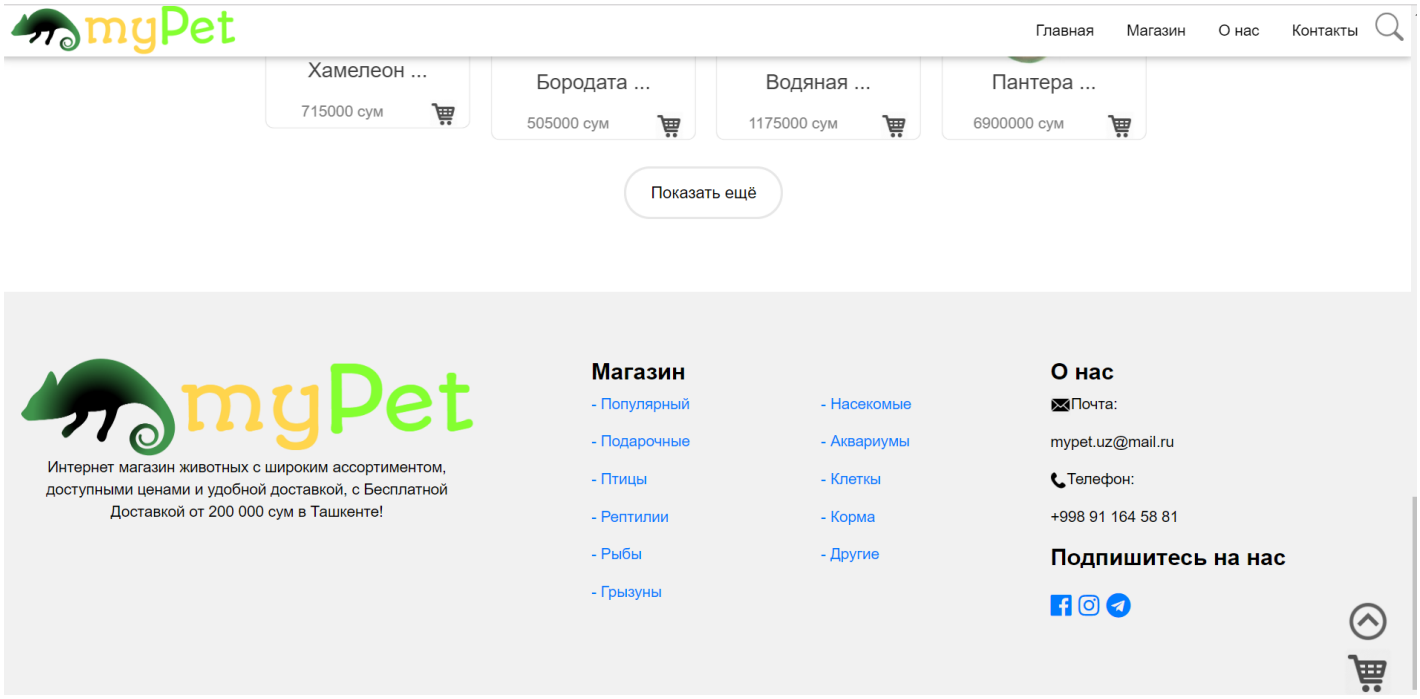

This is the main page that is entered by default through the public folder index.php file. It contains a landing page about the company and lower some categories of animal that could be interesting for the newcomer of the website.

I did all the necessary front end functionality of the website including bootstrap functionality:

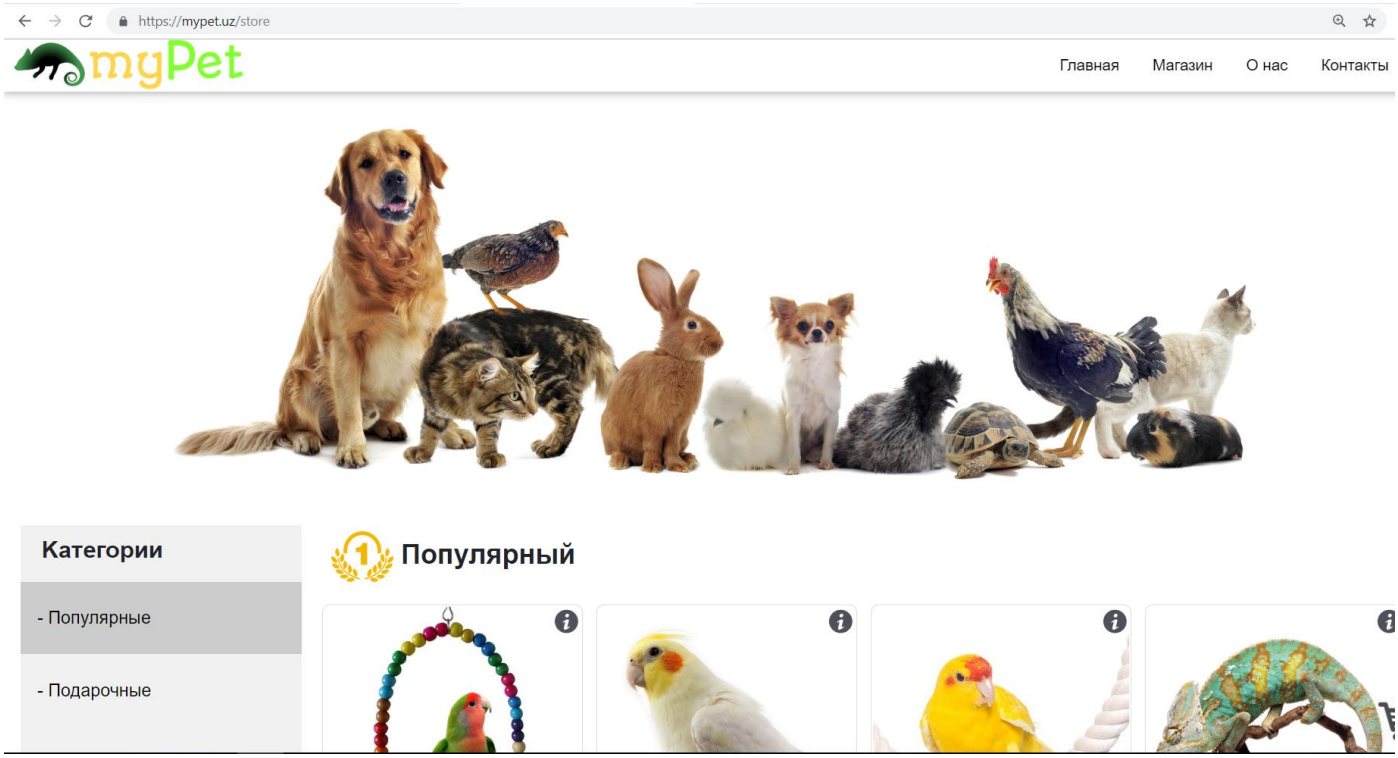

As you can see the animal items shrunk down to two, although there are four items in the desktop version:



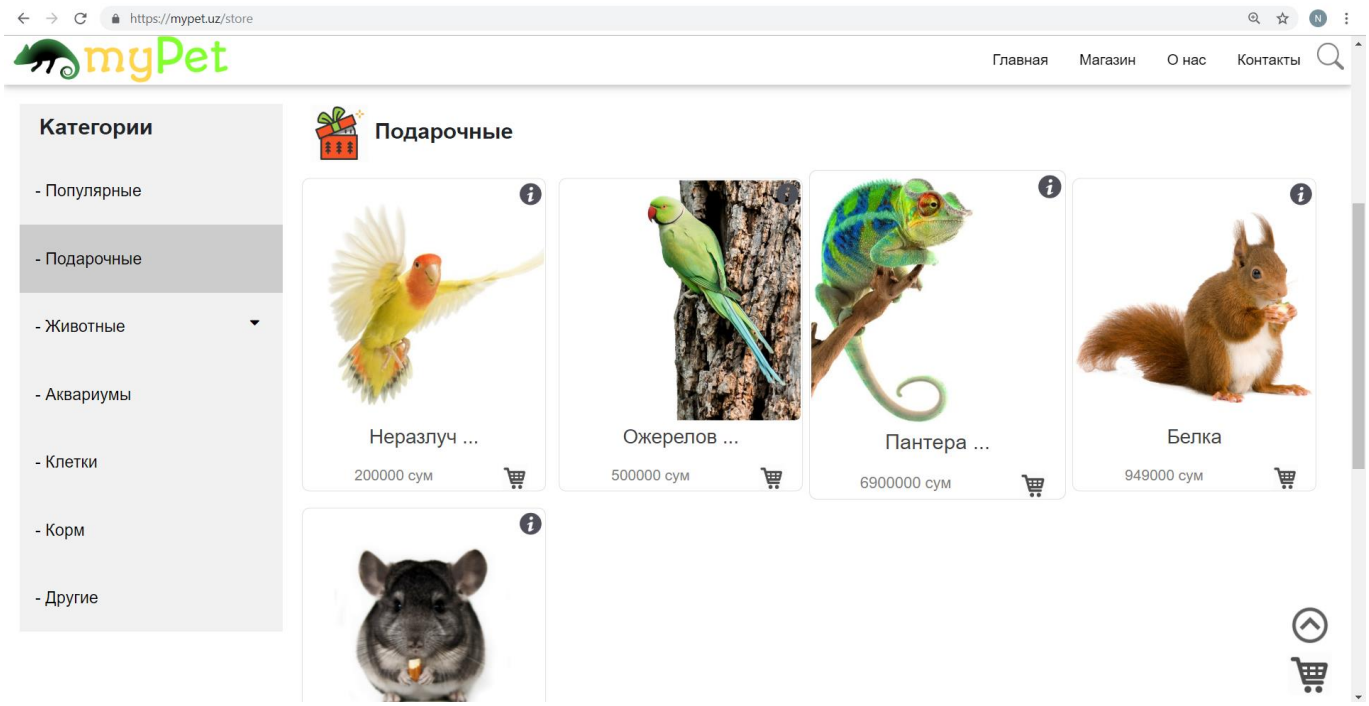

All of this was implemented by the help of bootstrap functionalities such as col-md-1 col-md-10 col-md-1, by the usage of row classes and so on.



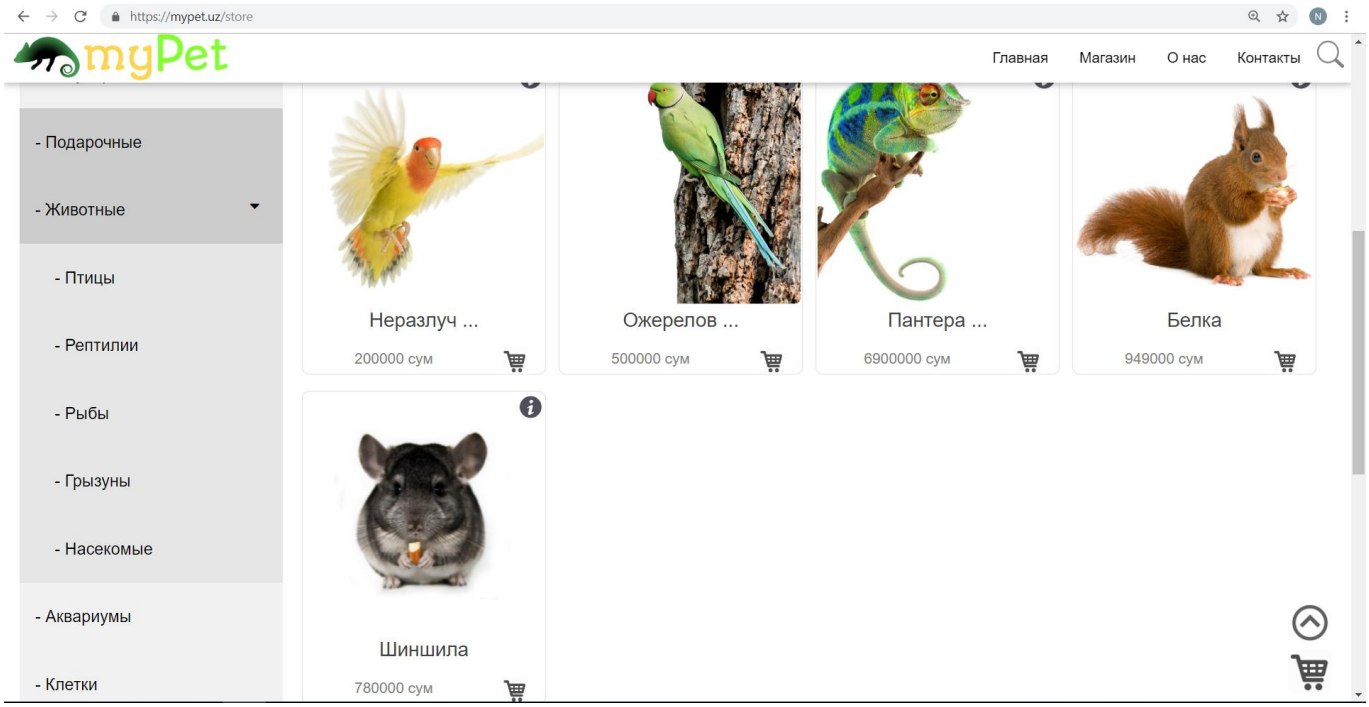

In this picture you can see the footer design and the fixed bar at the top of the website, which makes it easier for the user to navigate through the site.



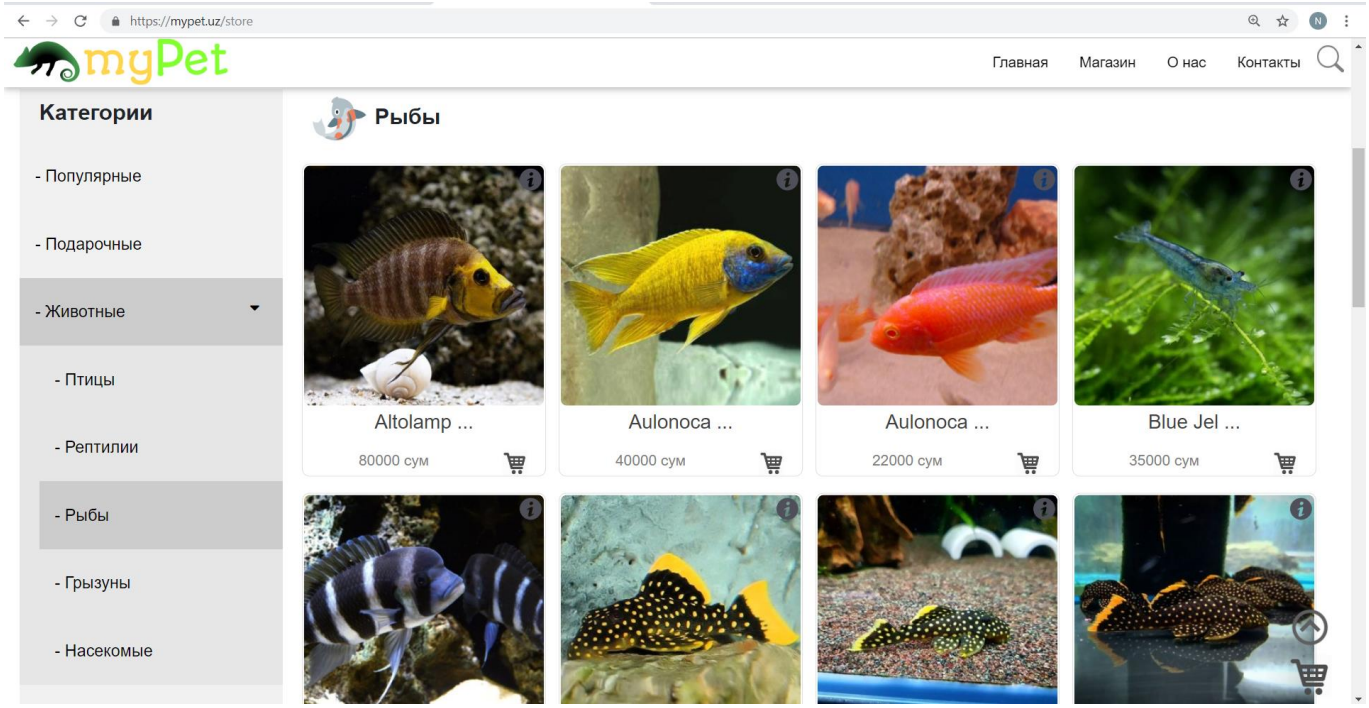

This is a shop page, which contains all the animals by categories and receives them by using the database functionalities of Laravel Eloquent ORM.

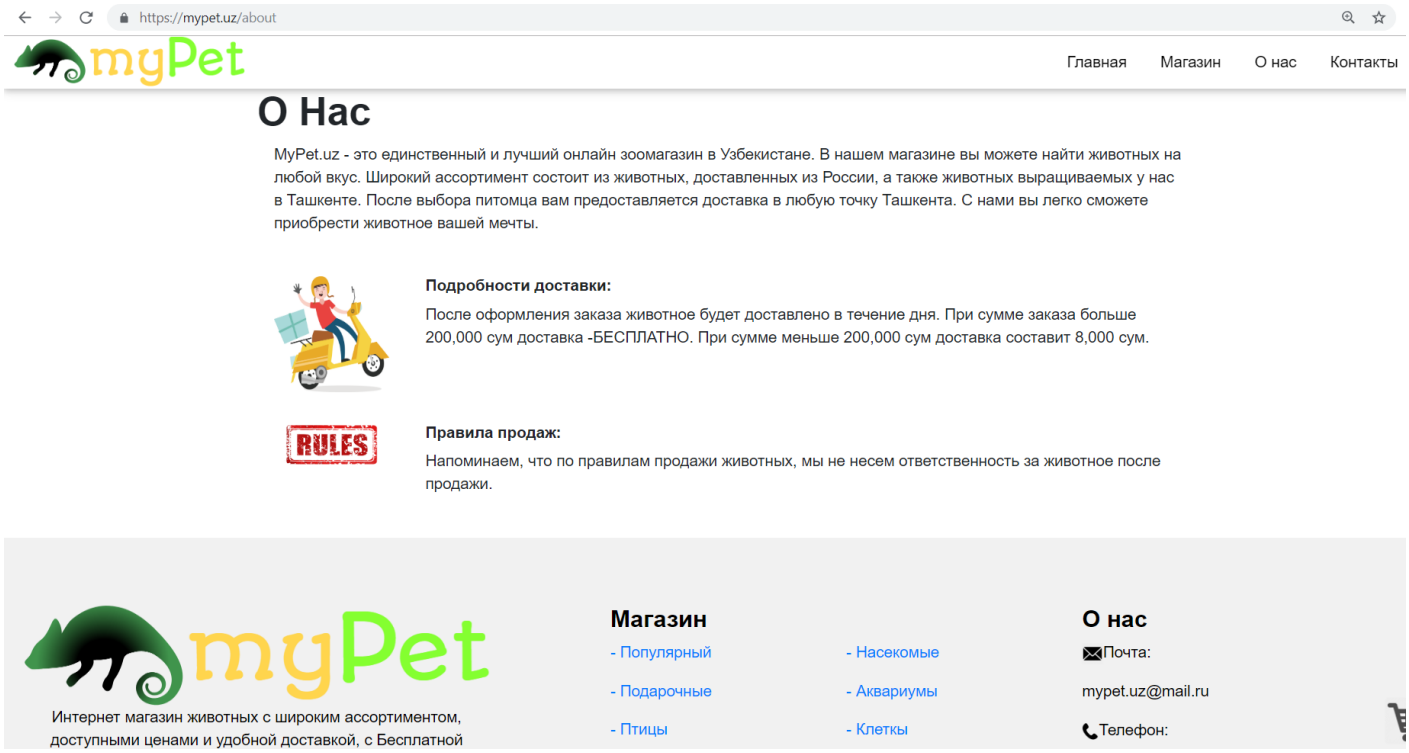



You can see the nav-bar at your left including all the categories.

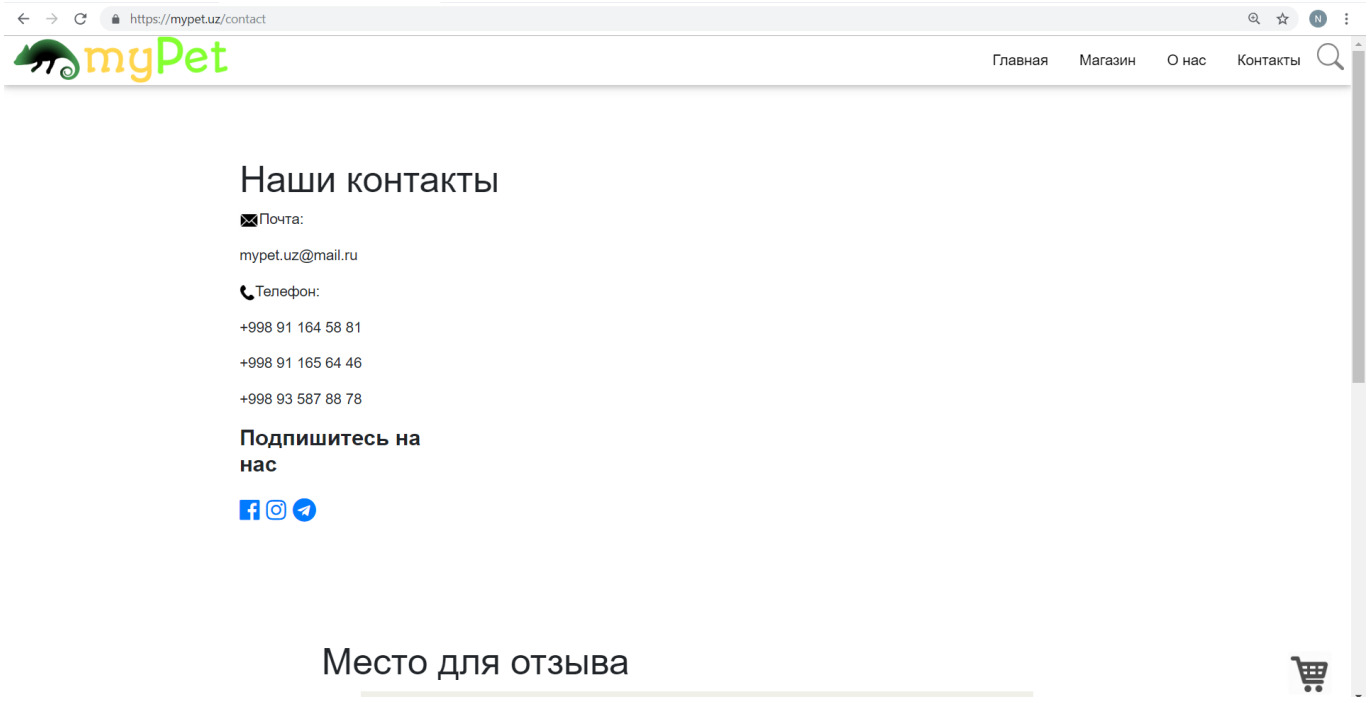


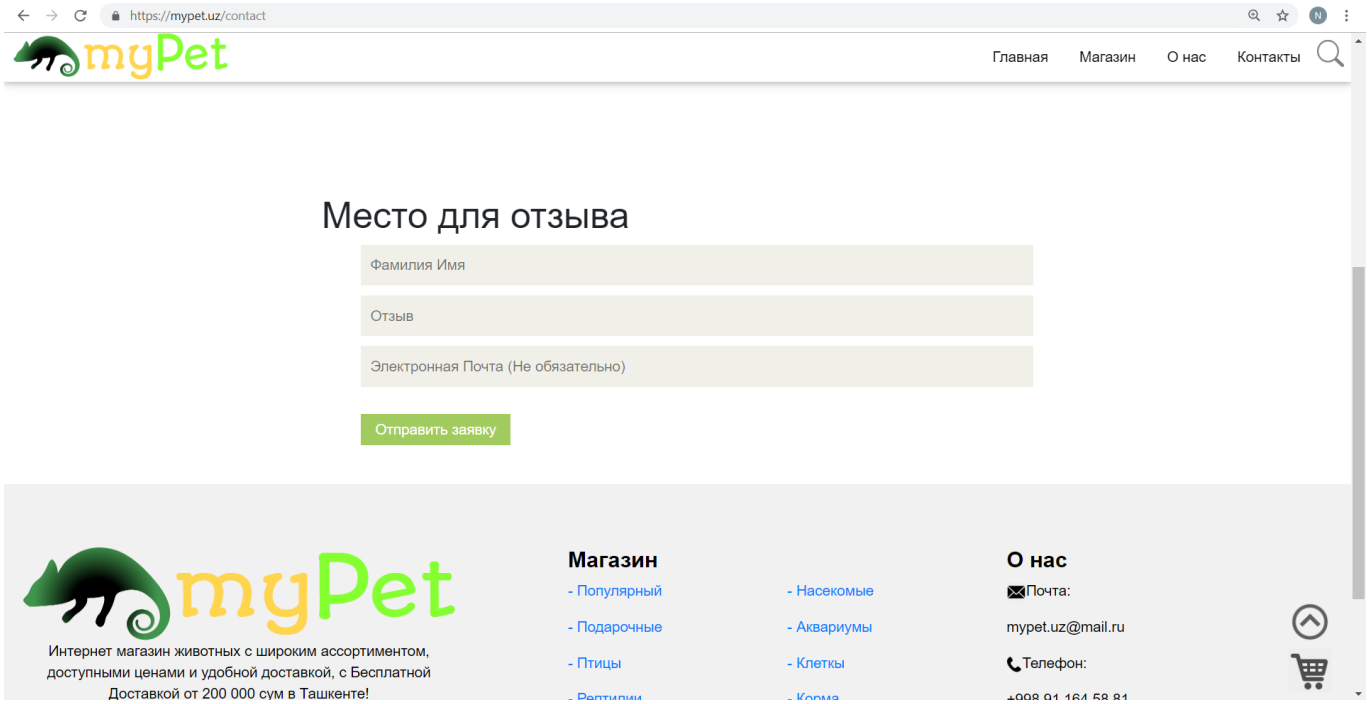

Here you can see the drop down in the nav bar, this is implemented by using Javascript DOM manipulation.






Here is the page with the detailed description of our company, that is already functioning as a business venture.



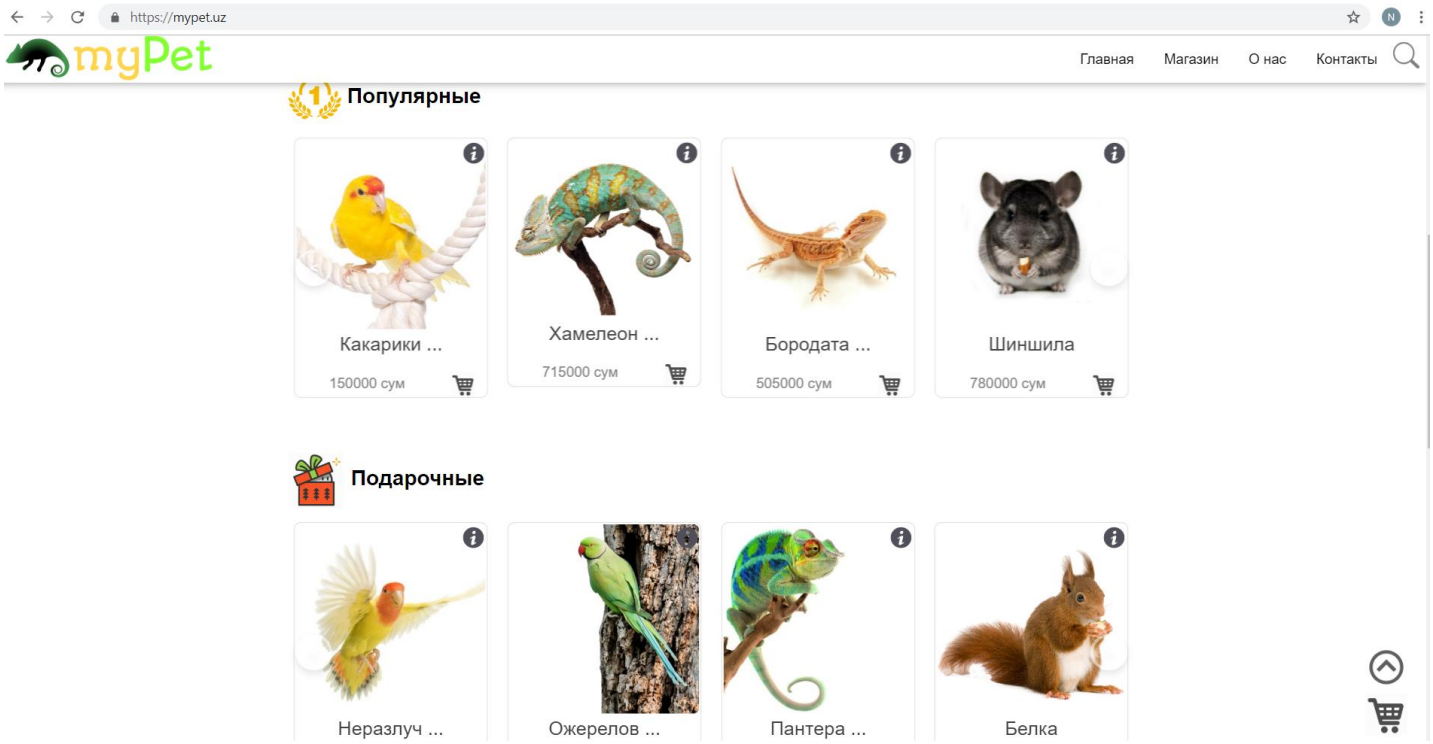



These are the photos demonstrating the page where users can leave a feedback about the website, which is sent directly to the mypet email address, which is mypet.uz@mail.ru.

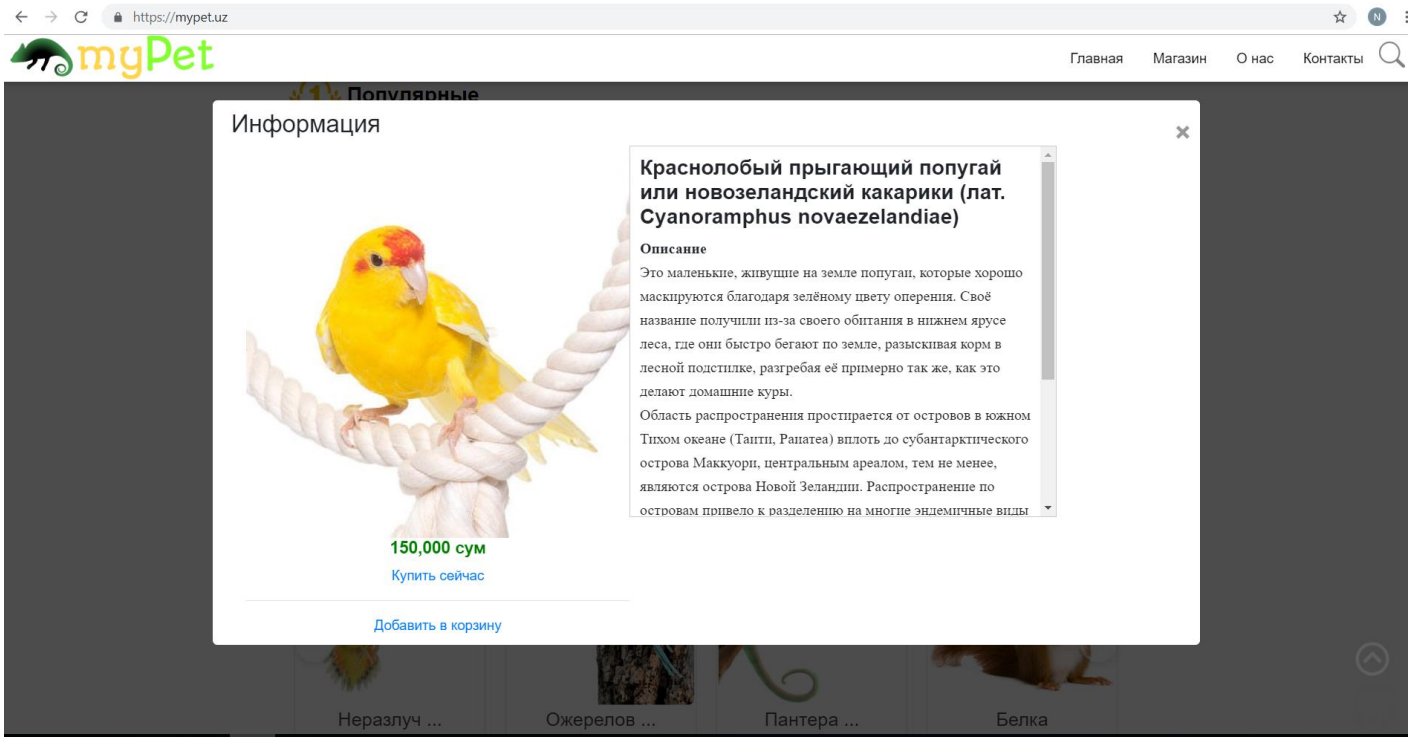
**Nazrulla Sadullaev, u1610167**

The things that I did were the backend functionalities included in Laravel for animal buying and cart management.

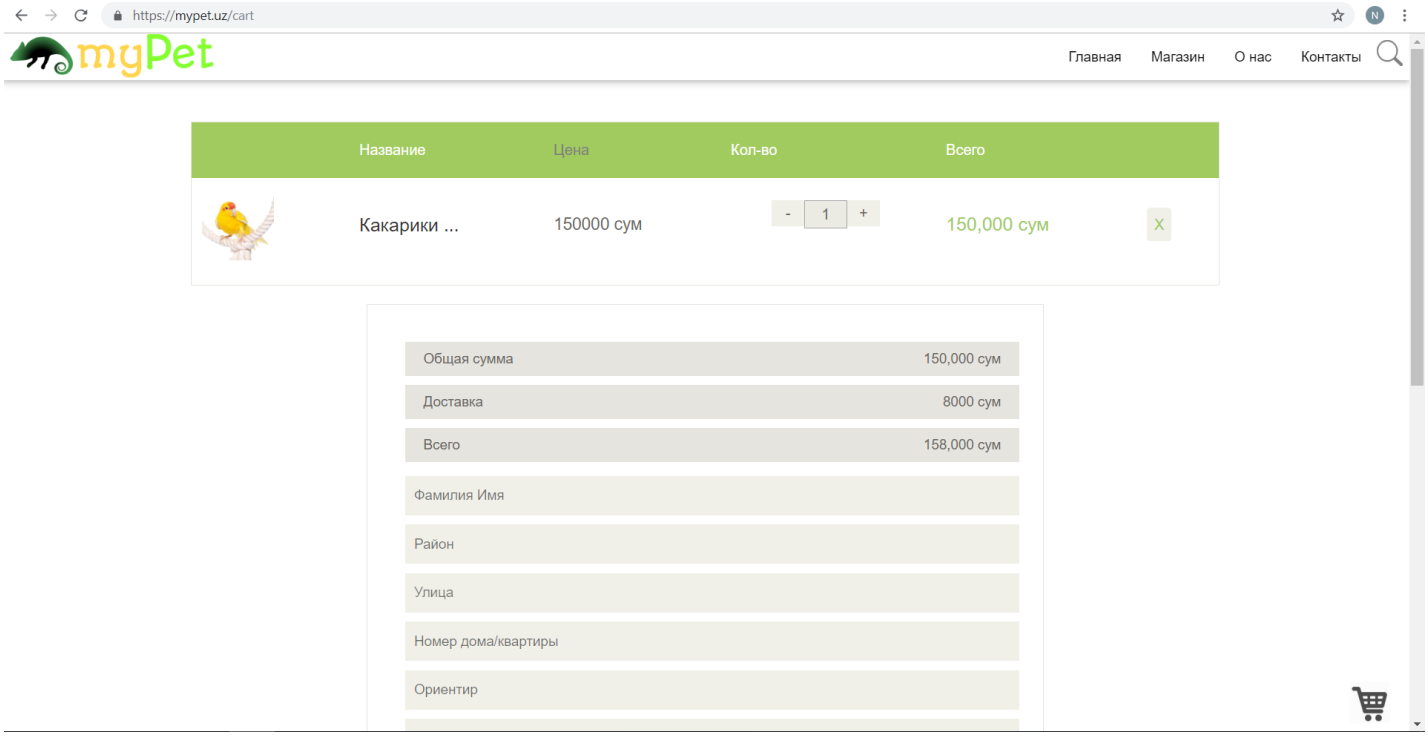
**Animal buying**:




Here you can see the main page with the animals categorized into different sections. In order to buy some animal, I implemented a controller called **CartController,** which has multiple functions.

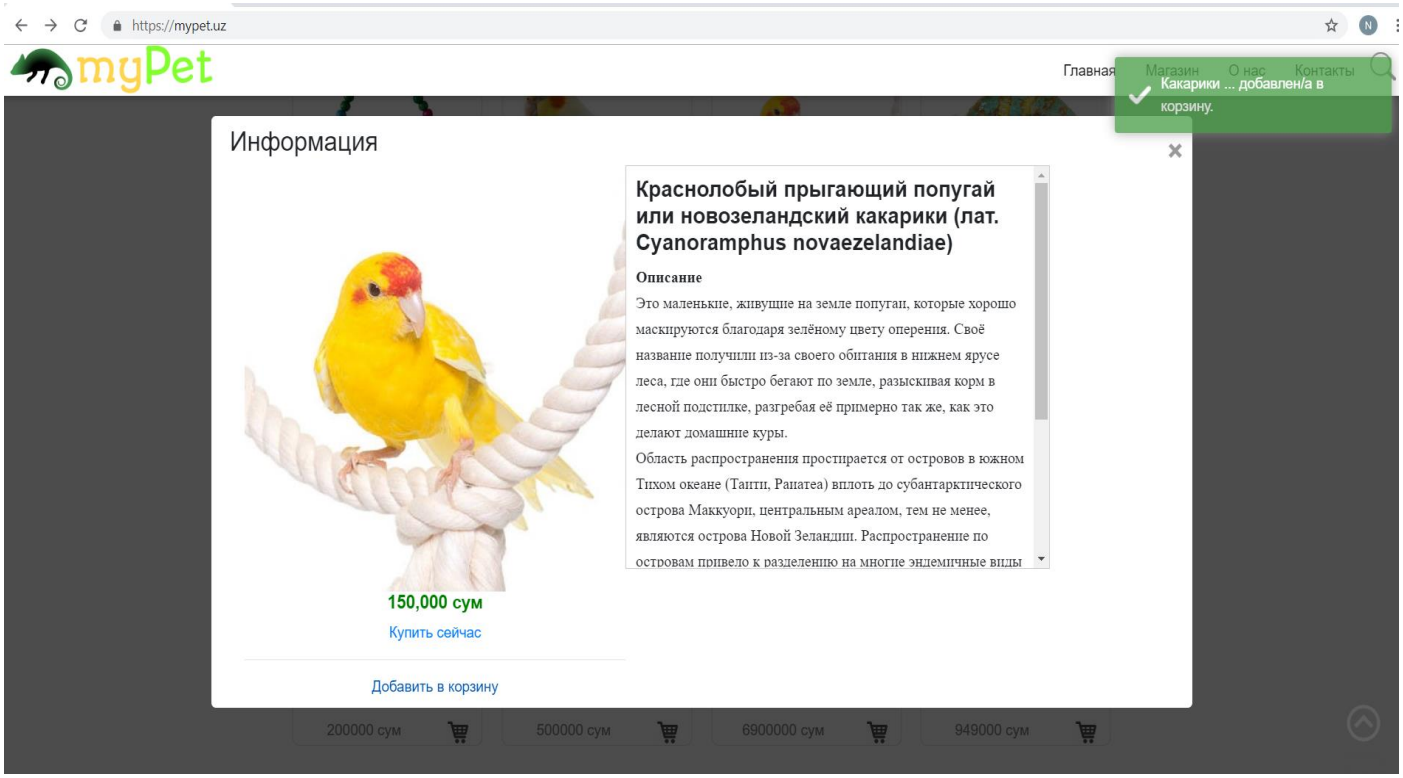



This is the result of pressing on some item. The information is retrieved through the route **“animal/{id}”**, this route is handled in web.php in routes folder. All the necessary information is retrieved from the database using **Eloquent ORM,** which was done by another team member.



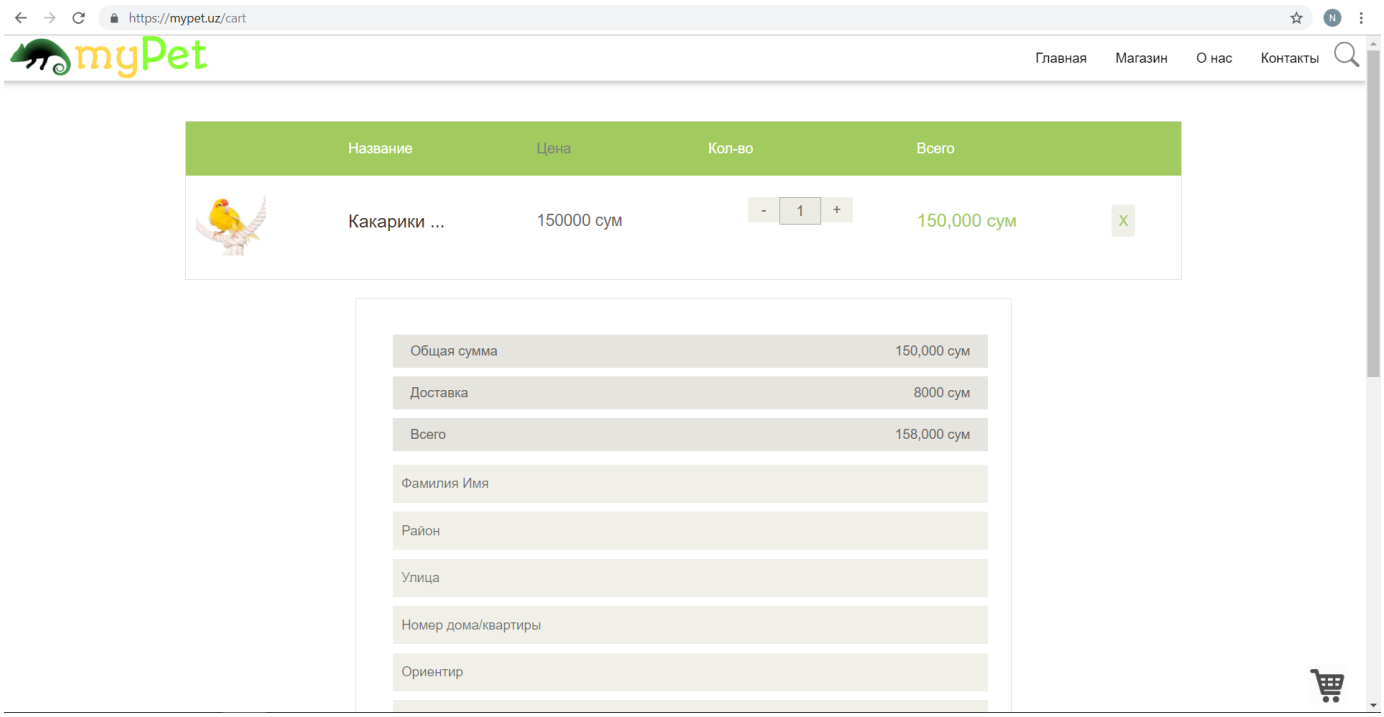

If the button “Купить сейчас” is pressed, the route **“cart/”** is activated and the CartController is handled as a resource in web.php, which led to an activation of the store function according to Laravel rules.

The corresponding page is open with all the necessary fields for order placement.

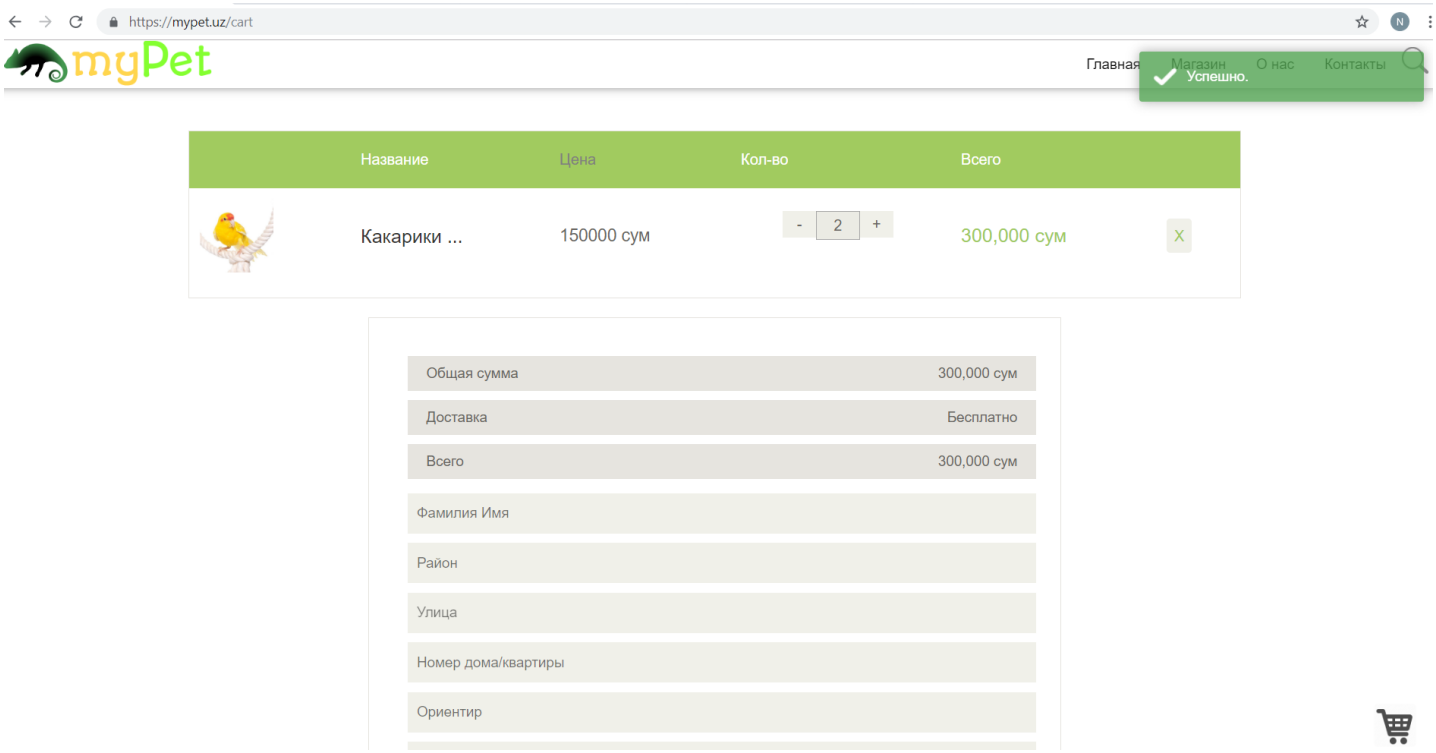



If the button “Добавить в корзину” is pressed, the route **“cart/”** is activated and the CartController is handled as a resource in web.php, which led to an activation of the store function according to Laravel rules.

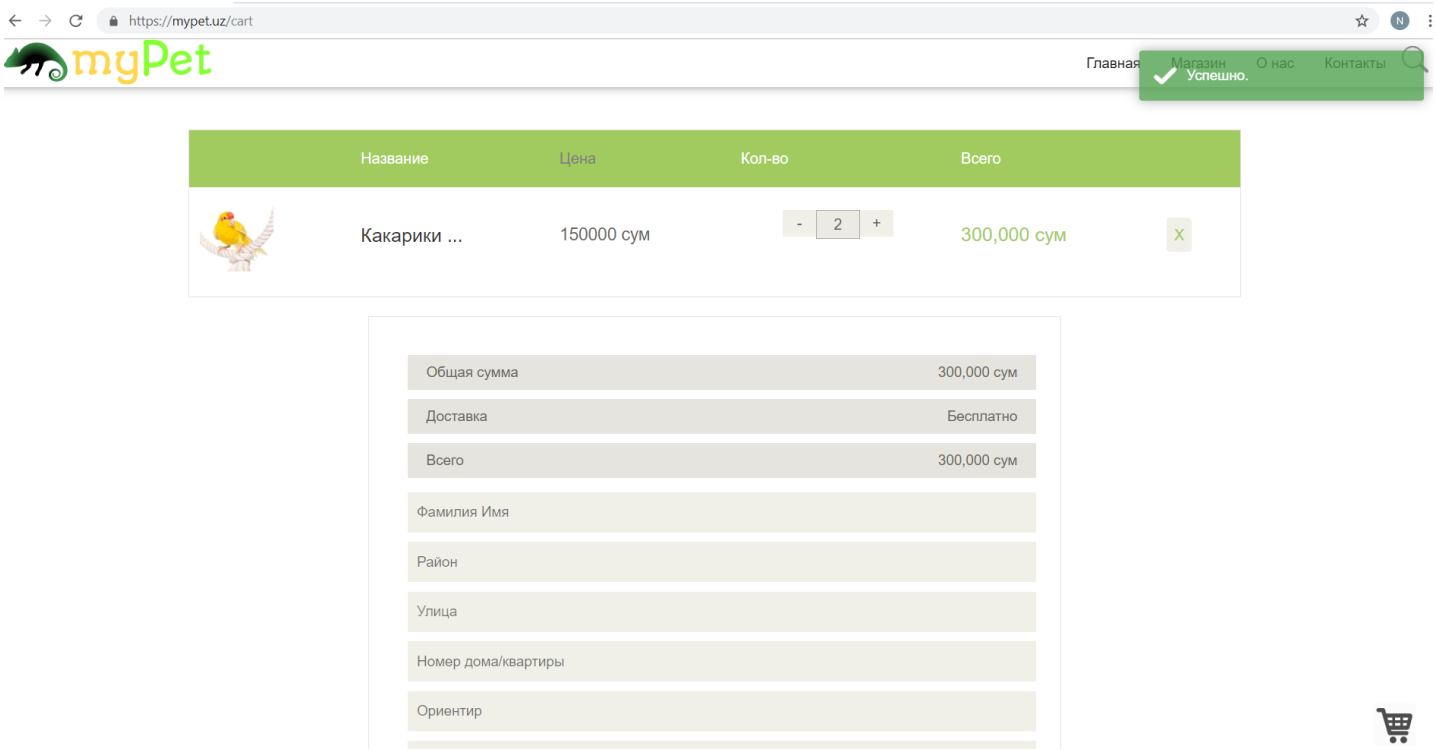
However, the page is not redirected, but a notification is shown about the addition of the animal to the cart, which was implemented by my team member.



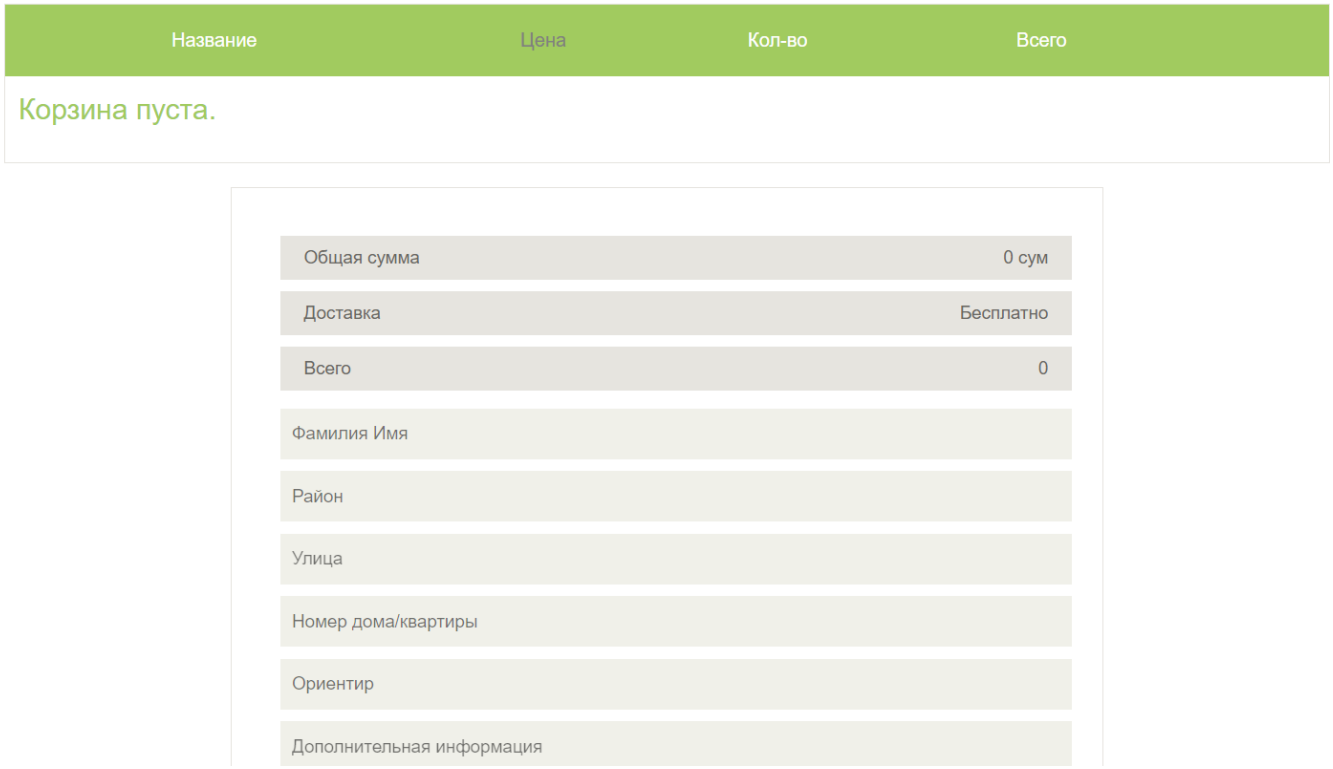

As you previously saw this picture, there are button to **add** and **subtract** an item from the cart.




As you can see after the pressing of the plus button the item has been added to the cart and the corresponding notification is shown. Whenever the **+** button is pressed, the route **"cart/add/{rowid}/{singleItem}"** is activated and is passed to addOne function of the CartControlled, which does the corresponding routine and update the Cart object.




As you can see after the pressing of the plus button the item has been removed from the cart and the corresponding notification is shown. Whenever the **-** button is pressed, the route **"cart/subtract/{rowid}/{singleItem}"** is activated and is passed to subtractOne function of the CartController, which does the corresponding routine and updates the Cart object.




Whenever the **X** button is pressed the item is removed from the Cart. This is done by the activation of destroy function in CartController.

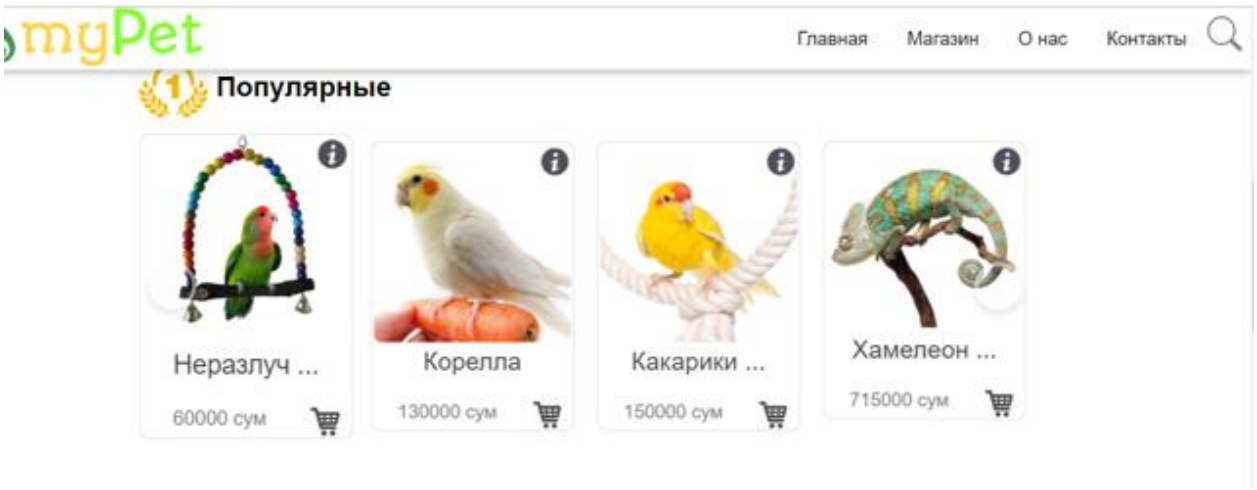
Here is the code of web.php where all the routing has been done:

```
<?php

  Route::resource('', 'AnimalController');
  Route::resource('/cart', 'CartController');
  Route::post('/mail', 'OrderController@sendMail')->name('makeOrder');
  Route::get('/cart/add/{rowId}/{singleitem}', 'CartController@addOne');
  Route::get('/cart/subtract/{rowId}/{singleitem}', 'CartController@subtractOne');

  Route::get('/animal/{id}', "AnimalController@getAnimal")->name('getAnimal');
  Route::get('/form/{id}', 'OrderController@showForm')->name('showForm');

  Route::get('/store', "StoreController@index")->name('store.index');
  Route::get('/contact', 'ContactController@index')->name('contacts.index');
  Route::get('/about', 'AboutUsController@index')->name('about.index');

  // AJAX Pagination
  Route::get('/animals/{type}/{page}', 'StoreController@get_by_page');
```

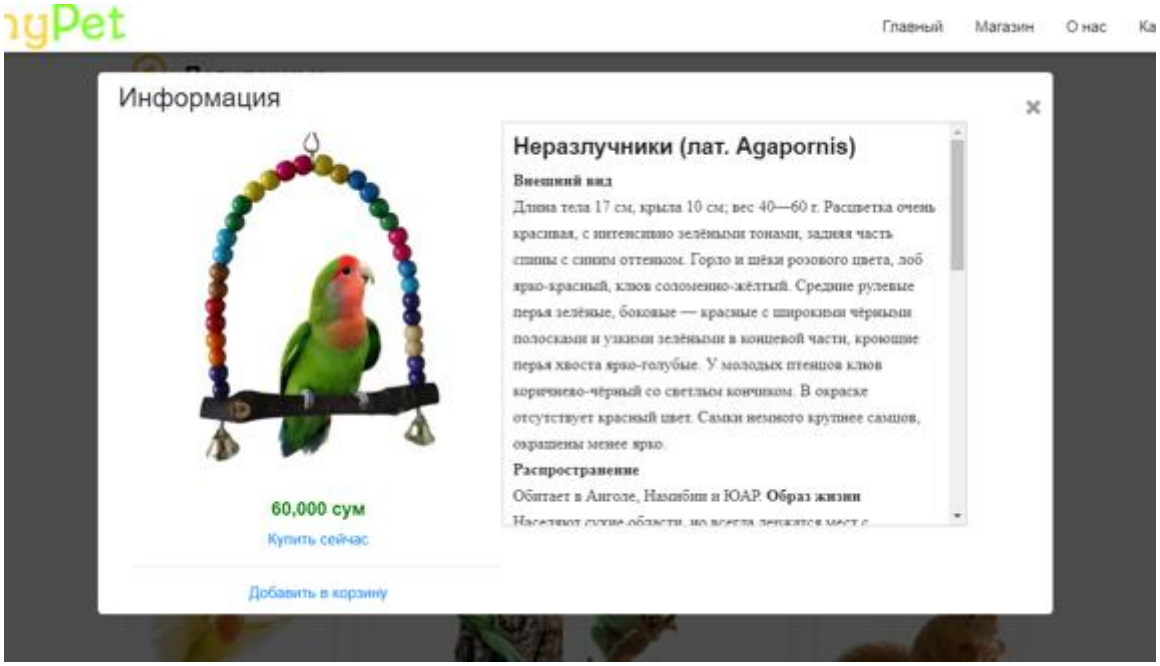
**Oybek Fayziev, u1610178**

My part is adding good design in our application which there are have many functionalities for the users with good design that there is hover over styles when mouse on the product, validations, adding to cards, incrementing and decrementing quantities with bonuses and other functionalities. When product has clicked users will take brief information about that product. Furthermore, there are have two functions which they can add to card their products or they can easily buy them. When they want to purchase quickly, there is a form for users which they store their personal information. On the other hand, they can buy later by clicking button add to card. I have done my part by bootstrap, jquery ajax, js.

Hover over style good design when user mouse on the product that item size will be bigger to differentiate others.

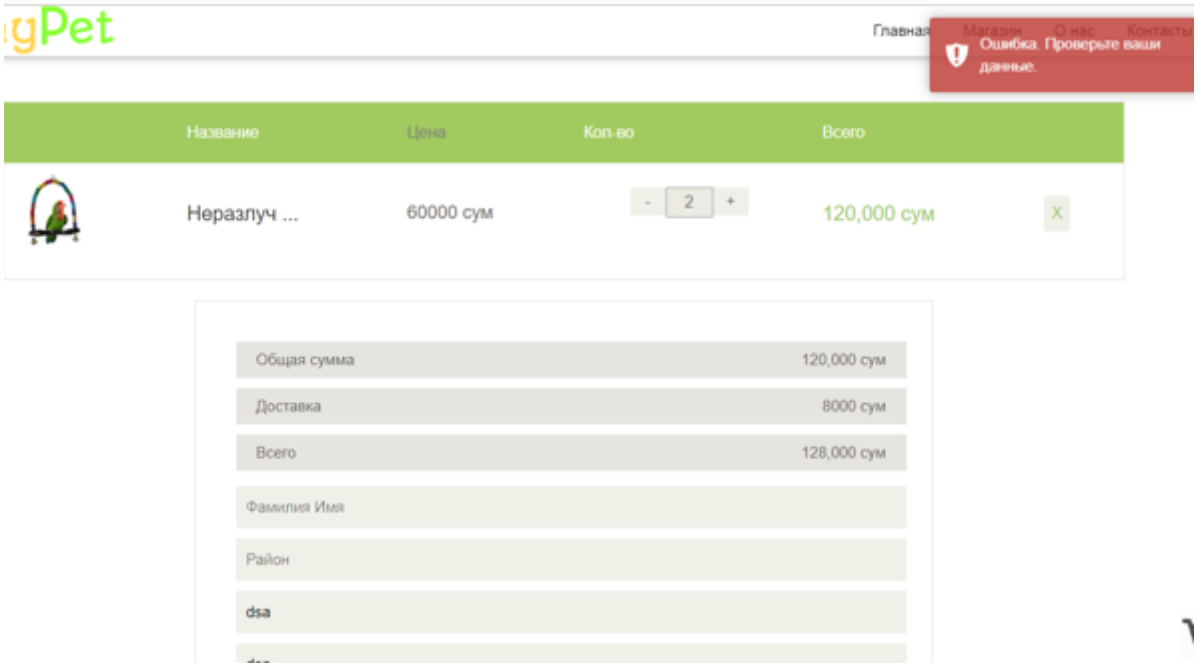



Whenever user click the product, they will get brief information about that animal and there are have two functions which *Купить сейчас* and *Добавить в корзинку.*




When some errors have arisen or some fields are missing, then form will not submit and shows the error. I have used JavaScript for validations.

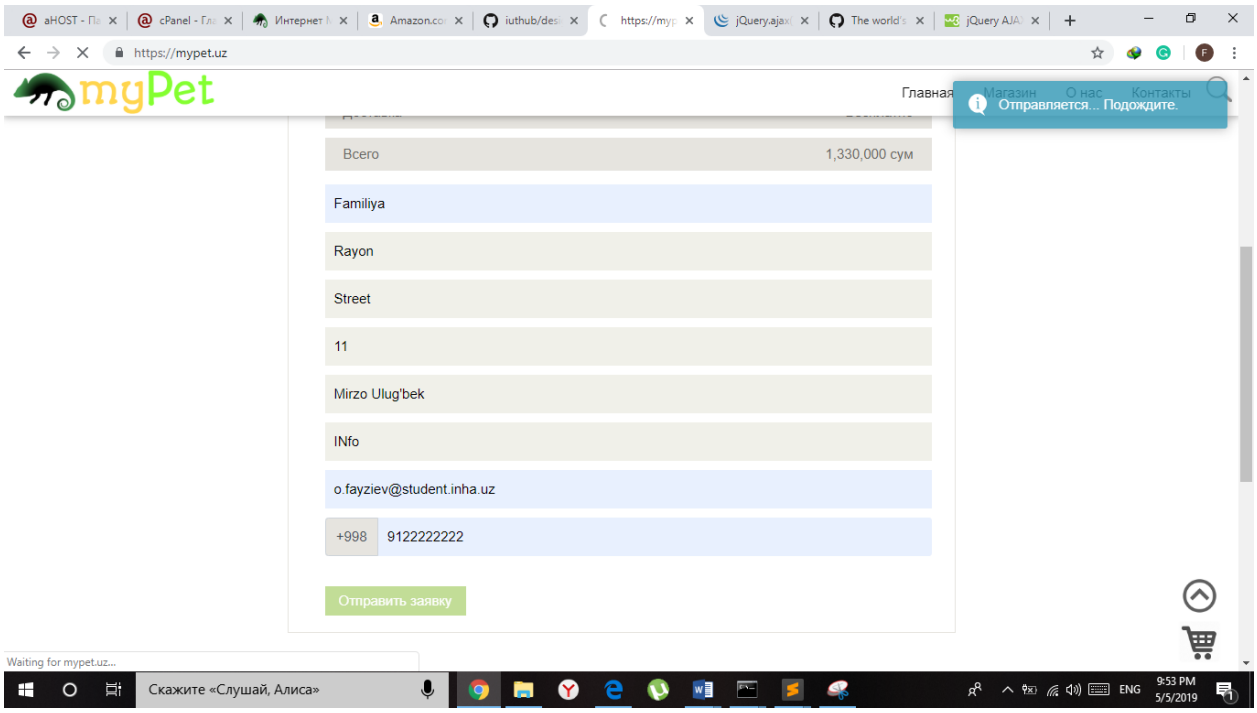
@if(\Illuminate\Support\Facades\Session::has('validation_error')) toastr.error('Ошибка. Проверьте ваши данные.') @endif




On the other hand, if all fields are perfectly, then request stores on the database.

While sending order, alerts *'Отправляется... Подождите.'* After submitting data prints *'Заявка принята.'* @if(\Illuminate\Support\Facades\Session::has('mail_sent') && \Illuminate\Support\Facades\Session::get('mail_sent')) toastr.success('Заявка принята.') @endif

ToastrJS library performs an action for notification

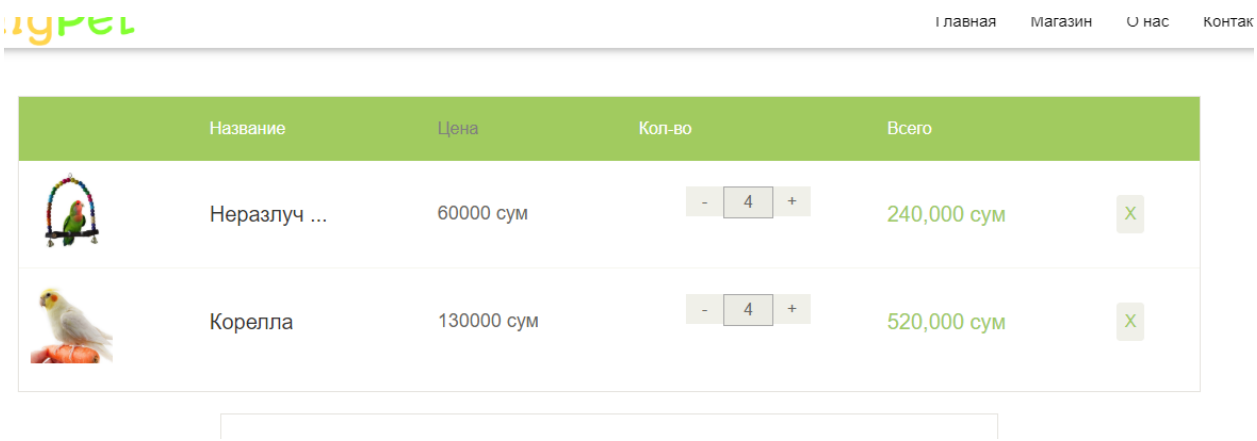



When *Добавит в корзинку* button clicked, it goes to {{--Script to Add to Card--}} in scripts.blade.php file. $.ajax({ url: '/cart', type: 'POST', data: { _token: token, id: $(this).data('id')}, This line save some data to the server and notify the user once it's complete.

If successful stored than prints ' добавлен/а в корзину.'

var data = JSON.parse(response)

toastr.success(data + ' добавлен/а в корзину.')

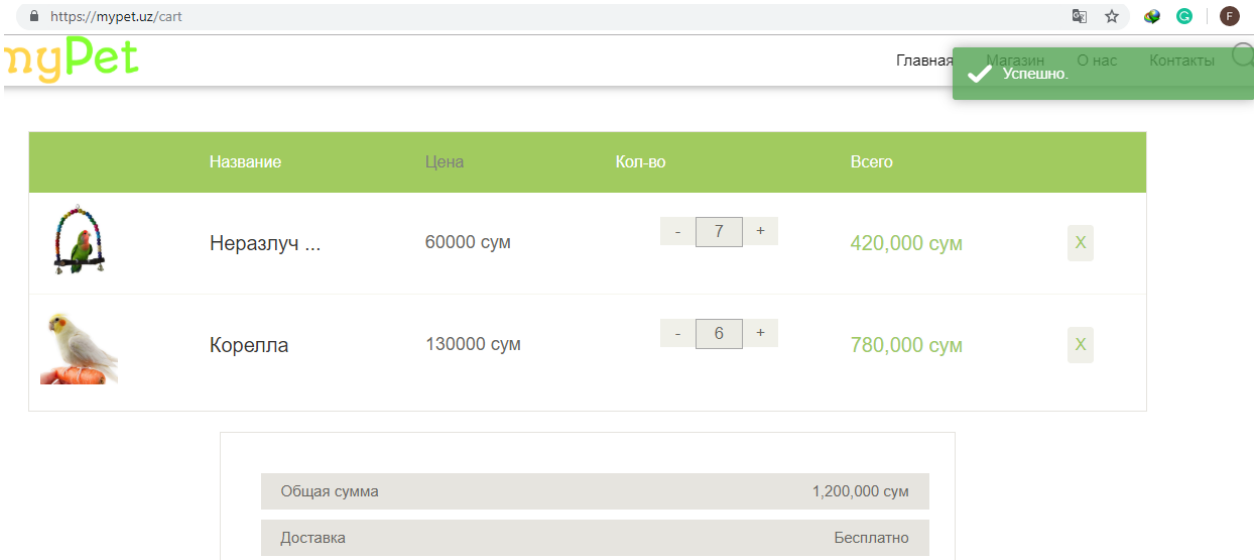



Furthermore, user can add or discard item's quantity.  Calculating total amount with delivery will be calculated from this formula total_amount + tax * 8000. Which tax is the variable initially equal to zero when total amount more than 200 thousand it will equal to one. JS does this staffs.

if(total_sum < 200000){$('.total_area .tax').html('Доставка 8000') tax = 1}

And AJAX will exchange data with the server and update parts of a web page without reloading the whole page.

$.ajax({ url: '/cart/subtract/' + $(this).data('id') + '/' + $(this).data('singleitem'), type: 'GET', success: function(response){toastr.success('Успешно.') $(this).attr('disabled', false)},)}})



**Nodirbek Nabiev, u1610169**

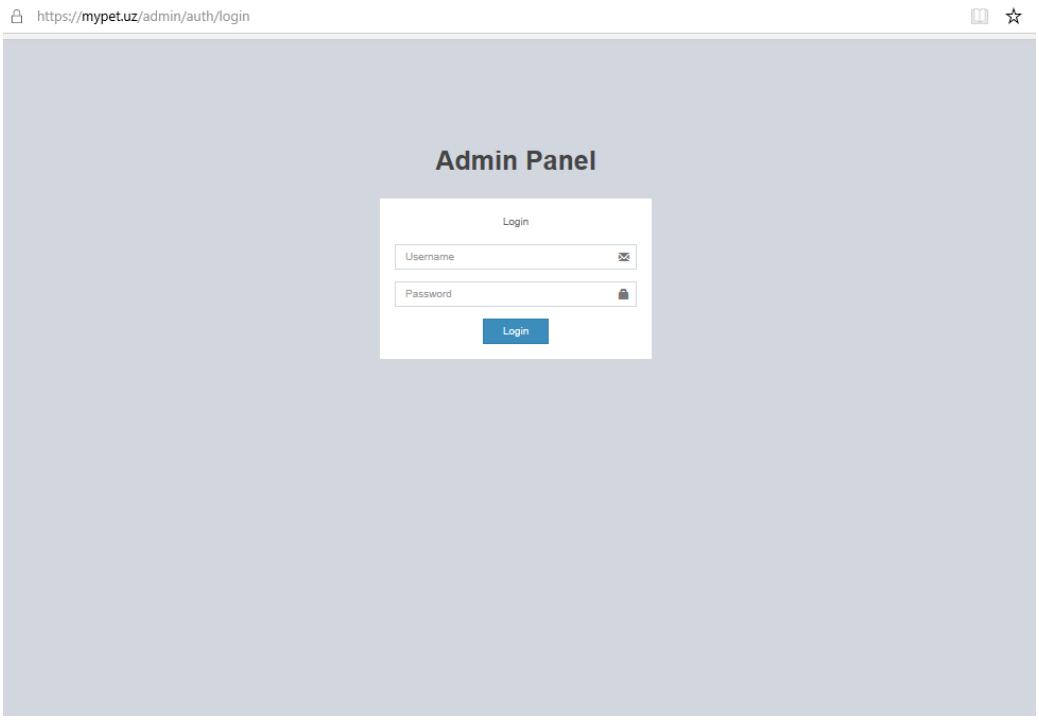
Mainly in this project I was responsible for the things like admin panel for the web-site, database in general and specifically with some usage of  the **Eloquent ORM** and the fields for entering the necessary data which comes up whenever the user wants to buy some product which will be sent to our e-mail.

First of all, the admin panel of the web-site using the Laravel-admin which is the admin builder based on the Laravel framework.

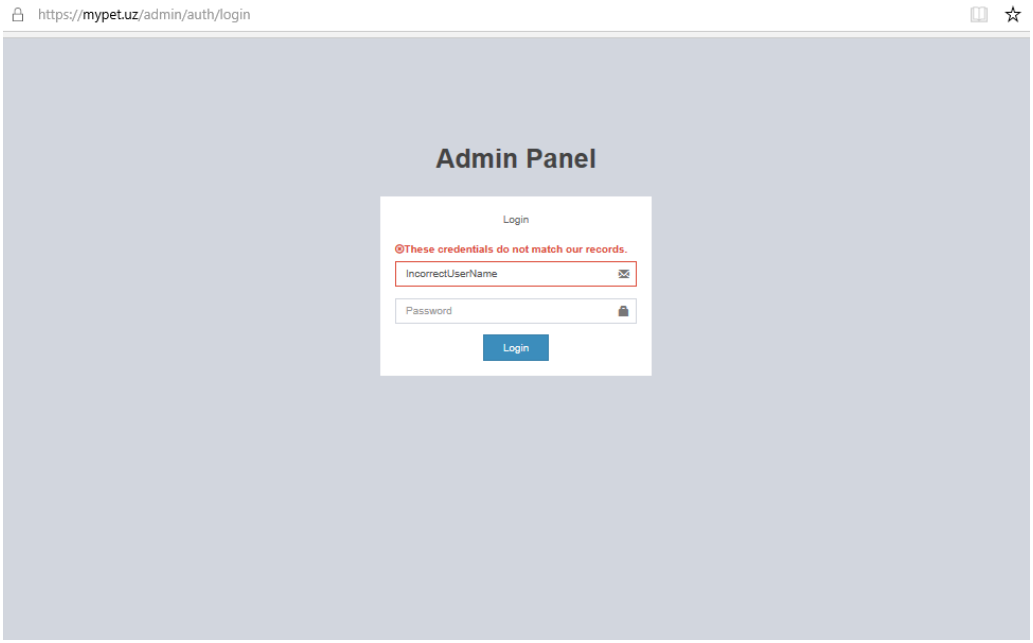
If you want to access the admin panel you should just write in query panel of the browser

https://mypet.uz/admin/auth/login

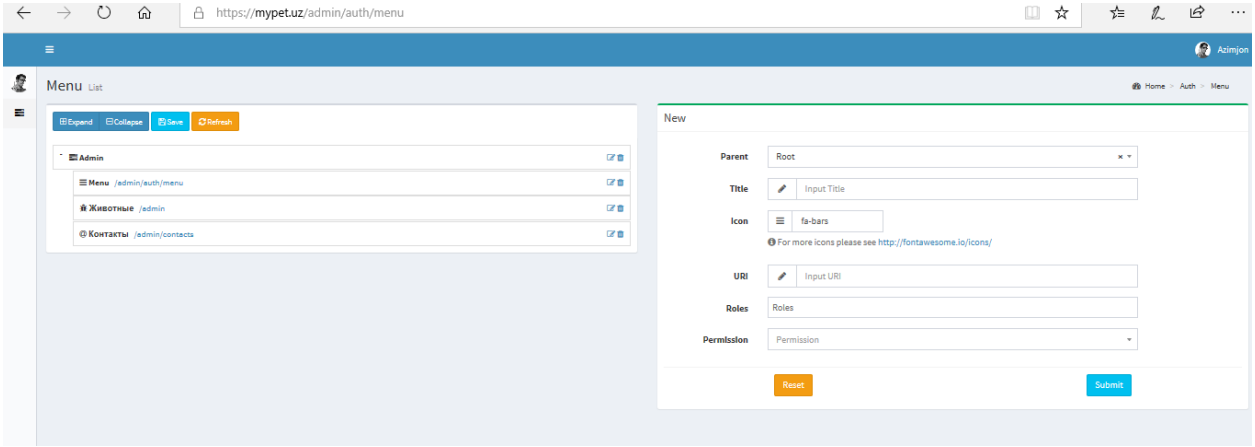
After you can see the authorization window




If you enter the wrong username or the password, it will say that something is incorrect and you should reenter your username and password

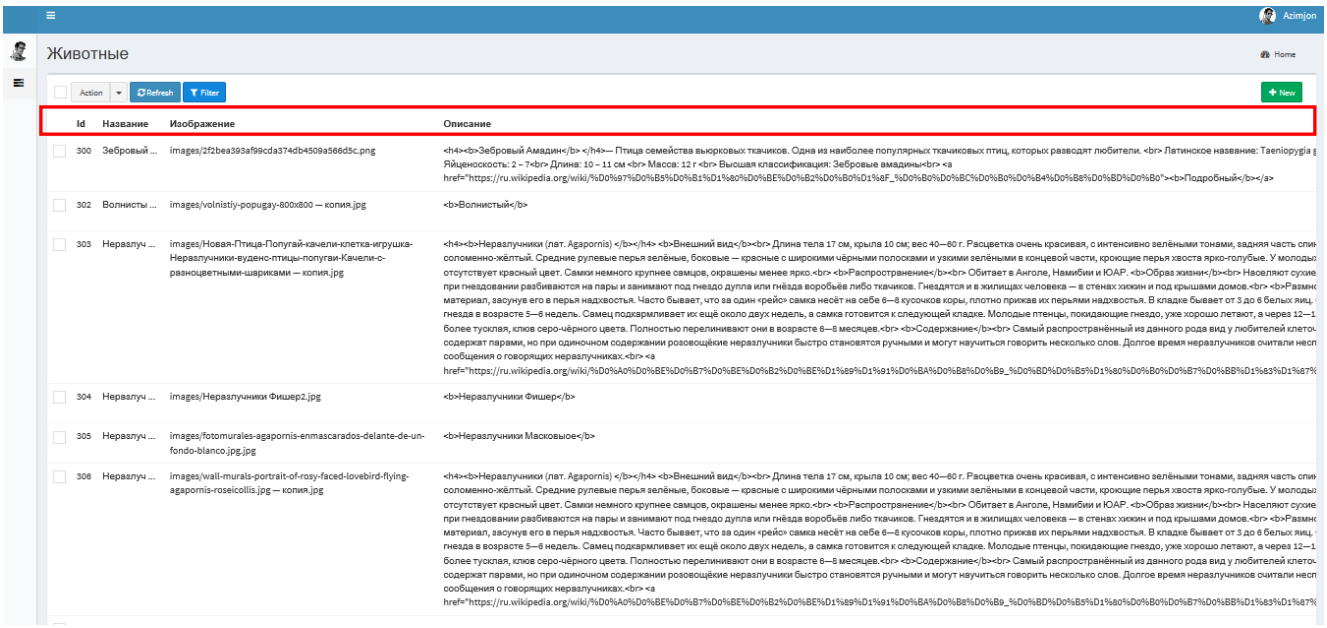



After successful registration into the admin panel, we can see the whole admin panel page.

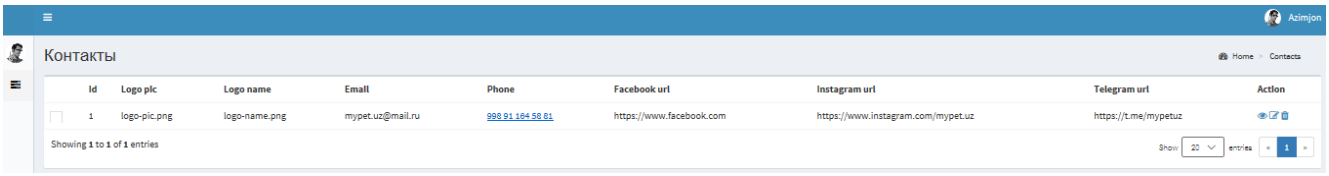



It should be noted that we can remove the built in field using **Encore\Admin\Form::forget(['map', 'editor'])** or extend customer form field **Encore\Admin\Form::extend('php', PHPEditor::class**). In fact, we can also require js and css assets using the following **Admin::css('/packages/prettydocs/css/styles.css');**

If we click to the “Животные” button in the admin panel menu list, we will see all the information about the animals like their “id”, “name”, “name of the image”, “description”, “price”, “action”. In the “action” we can delete or edit the information that already exists. All of them are shown in the following picture:




Next if we choose the “Контакты” we can see some additional information as shown in the picture:




The second thing that I was responsible for is the database. In the table creation process in the database, I have opened table on the migrations file and filled with attributes in this table.

**Example**(Table creation process. Taken from the actual source code)

```
use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;
class CreateUsersTable extends Migration{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up(){
        Schema::defaultStringLength(191);

        Schema::create('users', function(Blueprint $table){
            $table->increments('id');
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
```

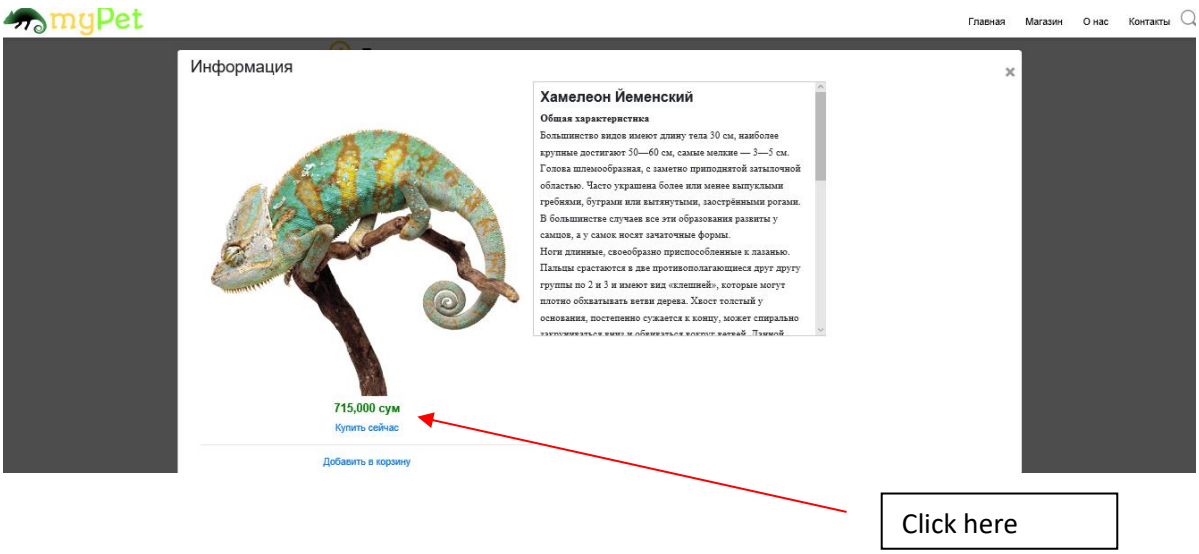
```
        }

        /**
         * Reverse the migrations.
         *
         * @return void
         */
        public function down(){
                Schema::dropIfExists('users');
        }
}
```

Note that the rest of the migrations are followed this condition.

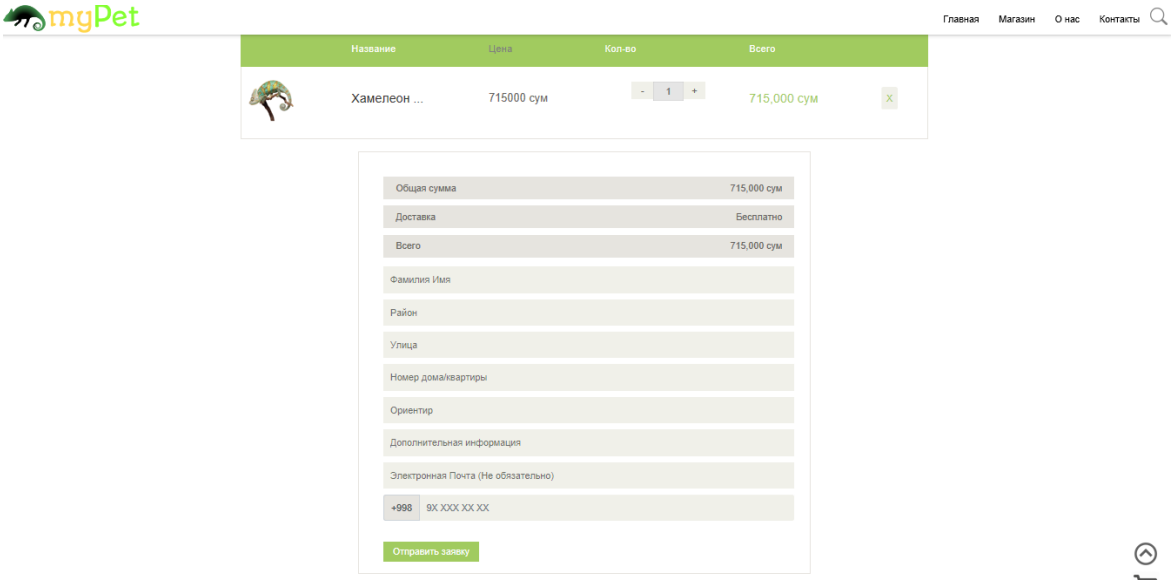
Also, I used the Eloquent ORM which is just the beautiful and simple realization of the “ActiveRecord” in the Laravel framework for working with database systems. Note that the creation of the Eloquent model starts just by extending the “Model” class like class User extends Model {}
**Example** (Part of the script taken from the actual source code)

```
<?php

 namespace App\Http\Controllers;

 use App\Animal;
 use App\Contact;
 use Illuminate\Http\Request;

 class AnimalController extends Controller{
  public $contacts = null;

  public function __construct(){
   $this->contacts = Contact::all()->first();
  }

  /**
   * Display a listing of the resource.
   *
   * @return \Illuminate\Http\Response
   */
  public function index(){
   $animals       = Animal::all();
   $best_animals   = $animals->where('is_best', '=', true)->take(12);
   $gifted_animals = $animals->where('is_gifted', '=', true)->take(12);
   $reptile_animals  = $animals->where('is_reptile', '=', true)->take(12);
   $contacts      = $this->contacts;

   return view('index', compact('best_animals', 'gifted_animals', 'reptile_animals', 'contacts'));
  }
```

The third thing that I did is the window that shows up whenever the customer wants to buy something from our web-site and e-mail sending to our e-mail address.




The following page shows up with the table that is needed to be filled whenever the user want to order some animal:




After submitting the form, the information will be send to our e-mail account and by this we can response the customers. If the data not successfully send than redirects to index page. ShowForm function from OrderController. The following script describes it.

```
if(count(Mail::failures()) == 0){
      Session::flash('mail_sent', true);
    }else{
      Session::flash('mail_sent', false);
    }
    return redirect()->route('index');
```

And the validation process is implemented in the following script:

```
$validator = Validator::make($request->all(), [
      'checkout' => 'required',
      'full_name' => 'required|max:255',
```

```
  'email' => 'present|max:255',
  'region' => 'required|max:255',
  'street' => 'required|max:255',
  'house_num' => 'required|max:255',
  'address' => 'required|max:255',
  'phone' => 'required|max:255',
]);
```