Титульный лист

МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ

Государственное бюджетное профессиональное образовательное учреждение Московской области «Ногинский колледж»

Курсовой проект

по МДК.09.01 Проектирование и разработка веб-приложений

ПМ.09 Проектирование, разработка и оптимизация веб-приложений

Тема:

ИСПОЛЬЗОВАНИЕ ФРЕЙМВОРКА LARAVEL В РАЗРАБОТКЕ ВЕБ-ПРИЛОЖЕНИЯ «Социальная сеть»

Разработчик: студент группы 3ИСПР2

Красовский Н.Р.

(подпись)

Оценка защиты курсового проекта

« »

Дата защиты

« » 2024

Руководитель проекта:

преподаватель Степанов С.О.

(подпись)

Ногинск, 2024 г

# Содержание

Оглавление

[Содержание 2](#_Toc169525396)

[Введение 3](#_Toc169525397)

[Основная часть 5](#_Toc169525398)

[**1.общие сведения о разработке веб-приложения на фреймворке** 5](#_Toc169525399)

[2.инсталляция всех необходимых программ и фреймворка, предпроектная подготовка к работе 6](#_Toc169525400)

[**2.1 Frontend часть (HTML, JavaScript и SCSS.)** 8](#_Toc169525401)

[**3.Backend часть (laravel)** 12](#_Toc169525402)

[заключение 15](#_Toc169525403)

[используемые источники 16](#_Toc169525404)

[приложения 17](#_Toc169525405)

# Введение

**Актуальность**:Курсовая работа актуальна тем что сегодня практически все люди используют социальные сети для общения между собой,что делает социальную сеть удобной для общения людей с общим кругом интересов.

**Цель**:Главной задачей на курсовой проект было создание социальной сети с помощью таких инструментов как Bootstrap и Laravel с требуемым функционалом.

В современном информационном обществе социальные сети стали неотъемлемой частью повседневной жизни миллионов людей по всему миру. Они перестали быть просто платформой для общения и развлечений, превратившись в мощный инструмент в деловой сфере, политике, образовании, маркетинге и других областях. Актуальность изучения влияния социальных сетей на современное общество не может быть недооценена, поскольку они оказывают значительное воздействие на нашу повседневную жизнь.

Социальные сети привлекают людей, преследующих различные цели: поддержание контакта со старыми знакомыми и поиск новых,поиск работы, продвижение своего бизнеса, профессиональное общение;обмен информацией и медиаконтентом с другими пользователями.

В социальных сетях ценят аутентичность и креативность, предпочитая контент, который отражает их уникальные интересы и взгляды. Что требует от брендов переосмысления текущих стратегий, адаптации к новым формам общения, включая неформальный тон и интерактивные форматы.

Система социальных сетей наблюдается тенденция к созданию персонализированных цифровых пространств в социальных сетях и мессенджерах. Пользователи стремятся к более контролируемому и упорядоченному цифровому опыту, активно используя функции для структуризации своего пространства, такие как папки, отключение звука уведомлений и создание отдельных чатов. Это позволяет им сосредоточиться на важном для них контенте и управлять потоком информации.

Для установки компилятора SASS, нужно зайти в программу Microsoft Visual Studio Code (далее - VSCode), зайти в раздел Extensions, далее в поиске написать “Live Sass Compiler” и нажать кнопку Install,таким же образом устанавливаем “Format HTML in PHP”

В общем,рынок социальных сетей демонстрирует стабильный рост и разнообразие,предлагая клиентам самые разнообразные функции для общения. Для успешной работы в этой сфере важно следить за актуальными трендами, улучшать сервис и адаптироваться к изменяющимся потребностям клиентов.

**Задачи:**

1. Выбрать тему курсовой работы.
2. Изучить Laravel .
3. Изучить Bootstrap.
4. Сделать сайт с помощью выбранных инструментов.
5. Написать и защитить курсовой проект.

# Основная часть

## **1.общие сведения о разработке веб-приложения на фреймворке**

В мире веб-разработки существует целое множество инструментов и технологий, которые помогают создавать и воплощать идеи в интернете. Однако, если вы хотите построить что-то по-настоящему великое, то необходимо обратиться к фреймворкам.

Laravel - это известный фреймворк, который помогает разработчикам создавать веб-сайты и веб-приложения на языке PHP.

В отличие от CMS, которые предлагают ограниченный набор готовых решений, Laravel даёт полный контроль над каждой линией кода. Разработчики могут создавать уникальные функции, настраивать и оптимизировать процессы под наши нужды и требования, создавая продукт, который отражает индивидуальность и взгляд на мир.

Laravel используется для создания различных типов веб-приложений, включая блоги, электронную коммерцию, социальные сети, управление контентом и т.д. Он предоставляет множество готовых компонентов и широкий функционал: маршрутизацию, авторизацию, работу с базой данных, шаблонизацию и пр. С помощью Laravel любой новичок веб-разработки может получить массу нового, полезного опыта.

Маршрутизация Laravel предоставляет мощный механизм маршрутизации, который управляет тем, как приложение отвечает на различные HTTP-запросы. Разработчики могут легко определять маршруты в одном файле или распределять их по группам для лучшей организации. Маршрутизация поддерживает такие функции, как:

привязка модели к маршруту, что позволяет автоматически инжектировать модельные экземпляры в методы контроллера.Группировка маршрутов для применения мидлваре, префиксов и пространств имен, что упрощает управление большими приложениями.

Шаблонизатор Blade — одна из звездных особенностей Laravel, Blade предлагает лаконичный синтаксис для вставки PHP-кода в HTML и обладает такими возможностями, как:наследование шаблонов, что позволяет определять макеты и расширять их в дочерних шаблонах.Компоненты и слоты для повторного использования кода и улучшения структуры приложения.

В целом Laravel соблюдает отличный баланс между функциональностью, гибкостью и простотой в использовании. Правда,и у него есть свои особенности и границы применения. Поэтому его нельзя назвать «самым лучшим PHP-фреймворком».

## **2.инсталляция всех необходимых программ и фреймворка, предпроектная подготовка к работе**

Установку необходимых программ и фреймворка стоит начать с PHP и Composer на компьютер.

PHP является основным языком программирования в Laravel, обеспечивая все аспекты разработки веб-приложений от обработки запросов до взаимодействия с базами данных и пользовательским интерфейсом. Перейдите на страницу загрузки PHP для Windows и установите необходимую версию или самую новую.

XAMPP предназначен для создания локальной серверной среды на вашем компьютере. Это позволяет разработчикам создавать и тестировать веб-приложения без необходимости использовать удалённые серверы. Для установки XAMPP достаточно скачать исполняемый файл с официального сайта apachefriends.org. После установки вы можете запустить XAMPP Control Panel для управления компонентами сервера (Apache, MySQL и другие).

Composer используется для управления зависимостями в Laravel. Установите его, следуя официальной инструкции на официальном сайте сайте.

После установки PHP и Composer вы можете создать новый проект Laravel с помощью команды create-project от Composer в командной строке:composer create-project laravel/laravel example-app.

После создания проекта запустите локальный сервер разработки Laravel с помощью команды serve в Laravel Artisan:cd example-app и php artisan serve

После запуска сервера разработки Artisan ваше приложение будет доступно в вашем веб-браузере по адресу http://localhost:8000.

Изучение документации:Посетите официальный сайт Laravel и изучите документацию, чтобы получить более глубокое понимание возможностей фреймворка.

Изучение Bootstrap — это популярный фреймворк для разработки адаптивных и стильных веб-интерфейсов. Он предоставляет готовые компоненты, сетки и стили, которые значительно ускоряют процесс верстки и делают веб-страницы совместимыми с различными устройствами и браузерами. Что бы использовать Bootsrap нужно вставить в Html часть:

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous"><script src=<https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js> integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous"></script>.

Так же можно установить его при помощи команды на свой проект npm: install bootstrap@5.3.3

Так же в файле .env нужно прописать название,имя пользователя и пароль от Базы данных

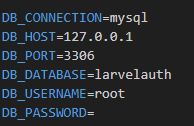


Рис.1.env

## **3.Разработка сайта**

### 3**.1 Frontend часть (HTML, JavaScript,BOOTSTRAP и SCSS.)**

В интернете я нашел бесплатный figma проект peapods-v1.0-(Community) и с него начал брать дизайн сайта.

Сперва для верстки страниц я решил её на логические блоки (шапка, основное содержание, боковая панель, подвал и т.д.).

Для работы нам нужно создать файлы html и scss.Открываем в редакторе VScode.

Начнем с шапки в ларавел и php можно создавать отдельные блоки создавать отдельно от основного кода у меня шапка получила название navbar.blade.php .

В Bootstrap есть сетка при помощи которой я разделил контент на маленькие части с ссылками на главную страницу,друзей,общий чат,поиск пользователей и меню стилизованной под иконку профиля пользователя.

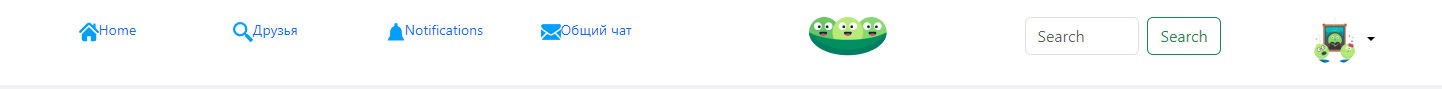


Рис.2. navbar

Затем я перешел к созданию основной секции которую пользователь увидит сразу при авторизации на сайт этот блок я назвал osn.blade.php.В этом блоке пользователь может увидеть свою иконку ,пост,тренды,рекомендованные события и предлагает добавить случайных людей в друзья.

Далее создал страницу с авторизацией и регистрацией включающей в себя форму для авторизации и кнопку с выпадающей формой регистрации

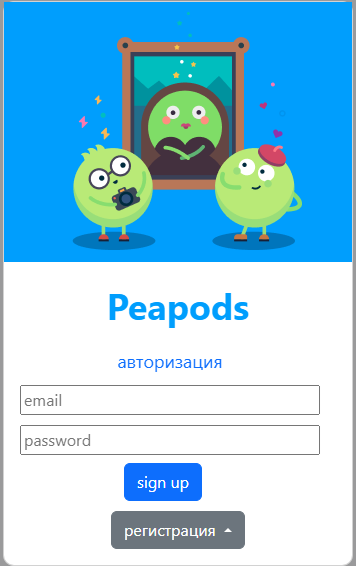


Рис.3.авторизация-регистрация

Страница профиля пользователя включает в себя имя,биографию,телефон,почту,город,возраст и пол пользователя которые пользователь ввел.

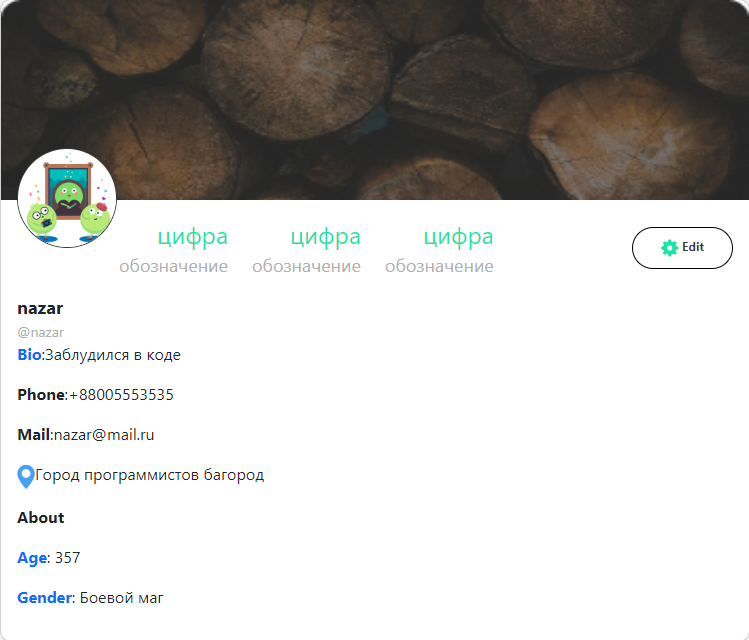


Рис.4 профиль пользователя

Кнопка edit открывает меню редактирования данных пользователя с формой где можно изменить все данные пользователя от фотки до города пользователя

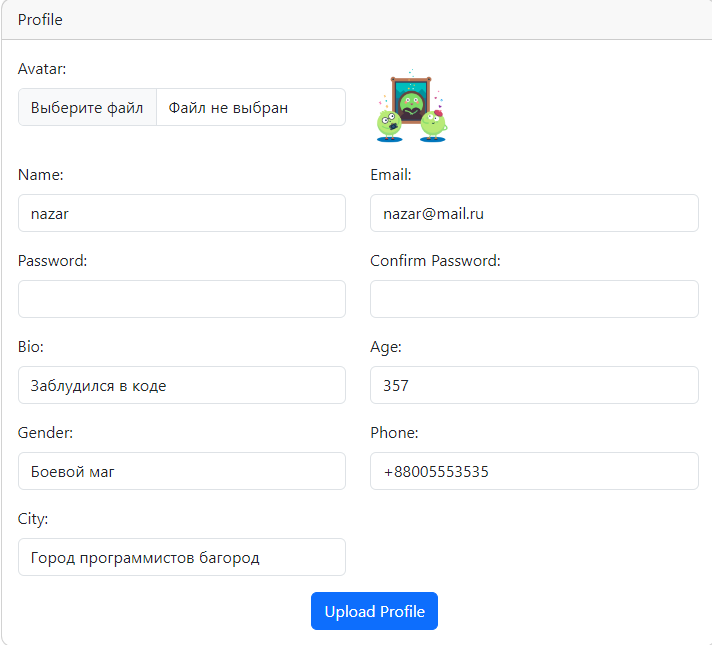


Рис.5.редактирование пользователя

Страница друзья включает в себя всех друзей пользователя ,все запросы в друзья,на этой странице можно перейти в профиль своих друзей и начать приватный чат с другом .

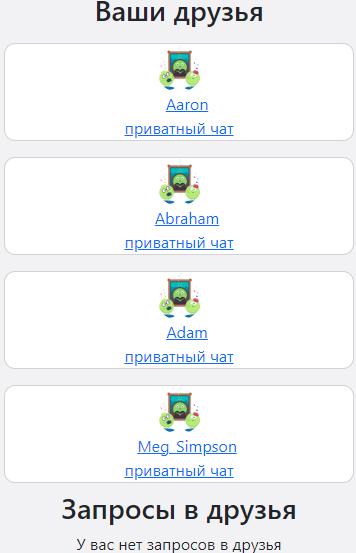


Рис.6.друзья

В приватном чате с другом указана дата и время отправки сообщения и само содержание сообщения для отправки сообщений внизу блока расположена форма с пустым блоком для ввода сообщения и кнопка отправить.

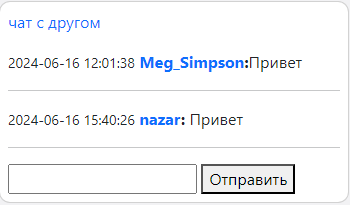


Рис.7.Приватный чат

Админ панель в которую можно перейти кликнув по логотипу социальной сети если у вас есть доступ его можно получить кликнув на иконку местанахождения.



Рис.8.Выдача админки

На этой странице можно изменить любые данные всех пользователей,а также можно забанить и разбанить пользователя.

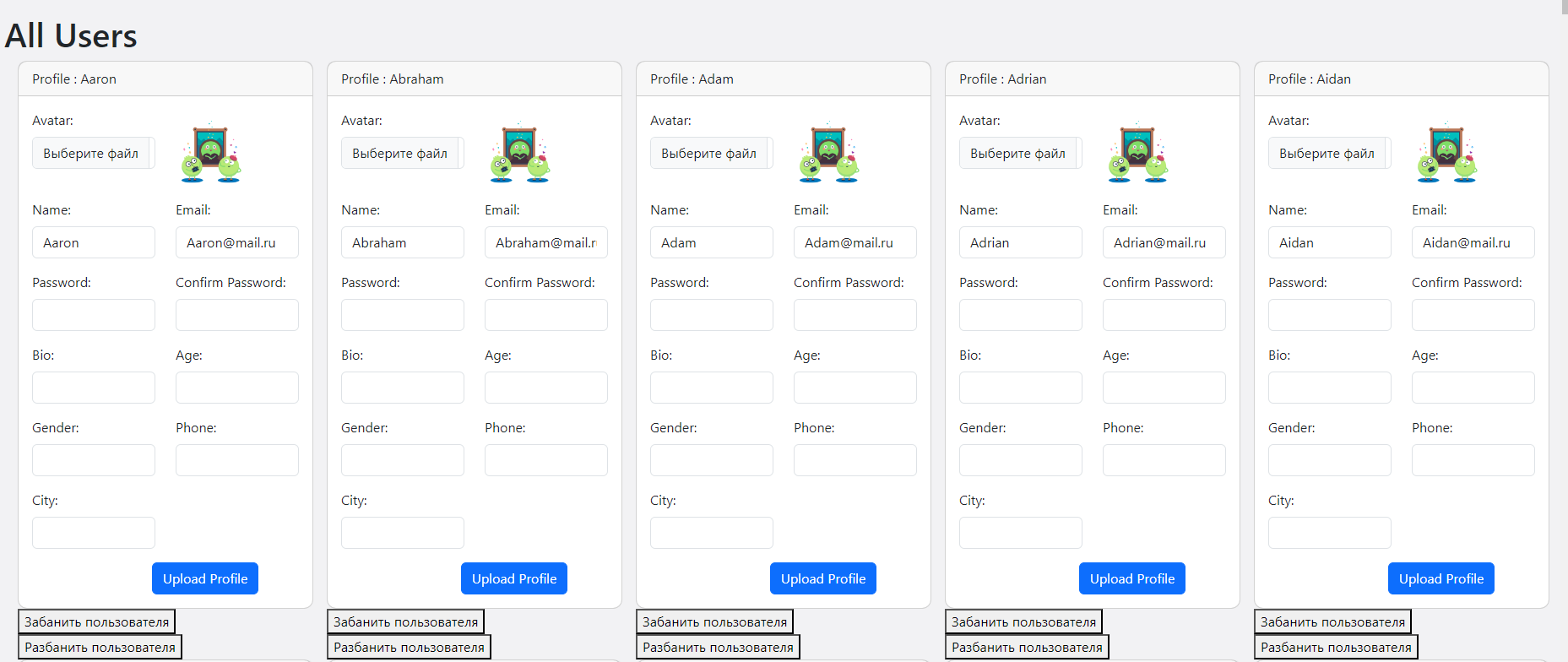


Рис.9.админ панель

Администратор может изменить личные данные пользователя, такие как имя, email, пароль, биография, изменить аватар пользователя.

Сохранение изменений: Нажатие кнопки "Upload Profile" сохранит все внесенные изменения.

Блокировка и разблокировка: Администратор может заблокировать (забанить) или разблокировать (разбанить) пользователя, управляя его доступом к системе.

Страница для забаненых пользователей.

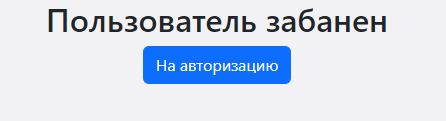


Рис.10.Страничка бана

### 3.2 Backend часть (laravel)

Для начала работы нужно создать файлы, в котором будет писаться весь код.

Основной файл для вывода всех страниц web.php в нем прописываются все пути и действия на странице и какое действие сделать из контроллера. Методы контроллера соответствуют действиям, которые приложение должно выполнять. Например, вы можете определить методы для отображения профиля пользователя, обновления профиля, удаления пользователя и т.д.

Для создания контроллера нам нужно прописать в консоли следующею команду



Рис.11.создание контроллера

В нем и был написан весь код backend.Первой функцией которую я начал делать это авторизация.Это была самая легкая задача так как по этой теме есть много гайдов.

Сначала была создана миграция которая позволяет создать таблицу в базе данных



Рис.12.создание миграции

В миграции прописывается все необходимые колонки

public function up(): void

{

Schema::create('users', function (Blueprint $table) {

$table->id();

$table->string('name');

$table->string('email')->unique();

$table->string('password');

$table->boolean('banned')->default(0);

$table->timestamps();

});

Например в этой таблице прописывается имя, ,почта, ,пароль,забанен ли пользователь и время создания аккаунта.

После этого нам нужно вписать в web.php следующие пути которые нам позволят обрабатывать запросы отправленные формами для авторизации,регистрации и выхода из аккаунта.

Route::get('register', [rcontroller::class,'create'])->name('register');

Route::post('register', [rcontroller::class, 'authstore']);

Route::post('/welcome', [rcontroller::class, 'store'])->name('login');

Route::get('/logout', [rcontroller::class, 'destroy'])->name('logout');

После создания таблицы,приступил к созданию регистрации и авторизации.Код регистрации выглядит следующим образом:

public function authstore(Request $request)

{

$request ->validate([

'name' => ['required', 'string'],

'email' => ['required', 'string', 'email', 'unique:users'],

'password' => ['required', 'confirmed', 'min:4']

]);

$user = User::create([

"name"=> $request->name,

"email"=> $request->email,

"password"=> hash::make($request->password)

]);

auth::login($user);

return redirect('osn');

}

В начале мы принимает данные имени пользователя его почты и пароль из формы.Затем хешируем пароль для обеспечения безопасности, чтобы в случае утечки данных злоумышленники не могли использовать их. После этого обращаемся к базе данных и заполняем введенные данные. Происходит автоматическая авторизация в аккаунт пользователя. Если имя пользователя уже занято, сайт выдаст ошибку о том, что такое имя пользователя уже существует.

Авторизация сделать проще потому что данные у нас уже есть выглядит это так:

public function store(Request $request)

{

$request ->validate([

'email' => ['required', 'string'],

'password' => ['required', 'string']

]);

if (Auth::attempt($request->only('email','password'))) {

$user = User::find(auth()->id());

if($user->banned == 1){

Auth::logout();

return redirect('banned');

}

else { redirect('osn');}

}

else {

return back();

}

}

Здесь мы просто сверяем данные данные с тем что записано в бд,а так же проверяем забанен ли пользователь и если пользователь не забанен он перенаправляет нас на основную страничку.

Код для выхода из аккаунта пользователя выглядит следующим образом

public function destroy(Request $request){

Auth::logout();

return redirect()->route('welcome');

}

В этой функции мы считываем нажатие кнопки logout и с помощью метода logout и внутренней функции laravel которая считывает авторизированного пользователя производим выход из аккаунта

В web.php нужно прописать метод post что бы принять данные из формы

Route::post('/profile',[rcontroller::class,'profilestore'])

->name('user.profile.store');

Код позволяющий редактировать свой профиль на странице пользователя выглядит следующим образом

public function profilestore(Request $request)

{

$request->validate([

'name' => 'required',

'email' => 'required',

'confirm\_password' => 'required\_with:password|same:password',

'avatar' => 'image',

]);

$input = $request->all();

if ($request->hasFile('avatar')) {

$avatarName = time().'.'.$request->avatar

->getClientOriginalExtension();

$request->avatar->move(public\_path('avatars'), $avatarName);

$input['avatar'] = $avatarName;

} else {

unset($input['avatar']);

}

if ($request->filled('password')) {

$input['password'] = Hash::make($input['password']);

} else { unset($input['password']);}

auth()->user()->update($input);

return back()->with('success', 'Profile updated successfully.');

}

Здесь мы проверяем ввели ли имя и почту пользователя ,а также указываем для формы для ввода что данные 2 полей совпадают и картинка пользователя отправлена как картинка.



Рис.13.Обновление профиля

Система общего чата позволяет всем пользователям отправлять свои сообщения. Для этого нужно создать отдельную миграцию где нужно указать для создания базы данных столбы такие как id сообщения ,user\_id идентификатор пользователя ,content в этой колонке будет сохранятся само сообщение которое написали, колонка chatroom позволяет нам узнать что это сообщение было написано в приватном чате или нет где приватный чат обозначается особым числом и если оно пустое то сообщение уходит в общий чат и timestamps это колонка запоминает время автоматически когда было отправлено сообщение.

public function up(): void

{

Schema::create('messages', function (Blueprint $table) {

$table->id();

$table->unsignedBigInteger('user\_id');

$table->text('content');

$table->string('chatroom')->nullable();

$table->timestamps();

});

}

Для того что бы контроллеры работали в файле web.php нужно прописать путь и контроллер который отвечает за эти функции где Route::get отвечает за получение из базы данных сообщений и Route::post который обрабатывает отправку сообщений пользователями

Route::get('/chat', [rcontroller::class,'chatindex'])->name('chat');

Route::post('/send-message', [rcontroller::class,'sendMessage'])->name('send-message');

Для страницы общего чата мне понадобится 2 контроллера где в первом из них я буду отображать все сообщения пользователей которые туда писали и игнорировать сообщения которые были написаны в приватном чате.

public function chatindex()

{

$messages = Message::with('user')->orderBy('created\_at')->get()

->whereNull('chatroom');

return view('chat', compact('messages'));

}

В функции chatindex мы в переменную messages сохраняем данные которые находим в таблице messages такие как имя пользователя , сортируем по дате создания и где колонка chatroom пустая.

Второй контроллер отвечает за функцию отправки сообщения в базу данных и её сохранения.

public function sendMessage(Request $request)

{

$message = new Message();

$message->user\_id = auth()->id();

$message->content = $request->content;

$message->save();

return redirect()->back();

}

Код добавления,показывания,удаление,принятие запросов в друзья и сам путь к сранице друзья прописывается в web.php следующим образом:

Route::view('/friend', 'friend')->name('friend')->middleware('auth');

Route::get('/friend', [rcontroller::class,'friendindex'])->name('friendindex')

->middleware('auth');

Route::get('/friend/add/{username}', [rcontroller::class,'getadd'])

->name('friend.add')->middleware('auth');

Route::get('/friend/accept/{username}', [rcontroller::class,'getaccept'])

->name('friend.accept')->middleware('auth');

Route::post('/friend/delete/{username}', [rcontroller::class,'postdelete'])

->name('friend.delete')->middleware('auth');

Route::get('/user/{username}',[rcontroller::class,'getprofile'])

->name('profile.index');

В общем используется 5 функций из контроллера.

Функция первая friendindex показывает всех друзей пользователя,код выглядит следующим образом:

public function friendindex(){

$friends = Auth::user()->friends();

$requests= Auth::user()->friendrequests();

return view ('friend',[

'friends'=>$friends,

'requests'=>$requests

]);

}

В этой функции мы записываем в переменную $friemds всех друзей пользователя и в переменную $requests все запросы в друзья а после возвращаем данные на страницу друзья.

Функция getadd позволяет отправлять запросы в друзья в ней мы проверяем есть ли запрос или пользователь уже в друзьях если нету то мы отправляем запрос.

public function getadd($username){

$user = User::where('name', $username)->first();

if (!$user){

return redirect()->route('osn')->with('info','пользователь не найден');

}

if (Auth::user()->hasfriendrequestspending($user)

|| $user->hasfriendrequestspending( Auth::user() ) ){

return redirect()

->route('profile.index',['username'=>$user->name])

->with('info','пользователю отправлен запрос в друзья');

}

if (Auth::user()->isfriendwith($user)){

return redirect()

->route('profile.index',['username'=>$user->name])

->with('info','пользователь уже в друзьях');

}

Auth::user()->addfriend($user);

return redirect()

->back()

->with('info','пользователю отправлен запрос в друзья');

}

Для кнопки в добавление в друзья нужно вставить в HTML

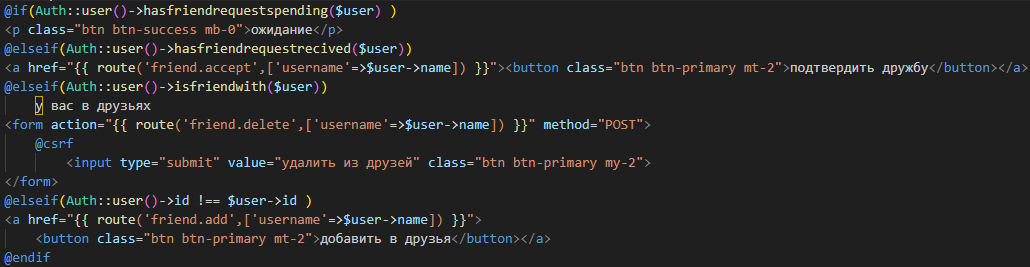


Рис.14.Код для кнопки отображения друзей

Функции getaccept нужна для принятия в друзья ее код выглядит так:

public function getaccept($username){

$user = User::where('name', $username)->first();

if (!$user){

return redirect()->route('osn')->with('info','пользователь не найден');

}

if (! Auth::user()->hasfriendrequestrecived($user)){

return redirect()->route('osn');

}

Auth::user()->acceptfriendrequest($user);

return redirect()

->route('search.results',['username'=>$user->name])

->with('info','запрос в друзья принят');

}

В этом коде мы в переменную $user записываем имя пользователя и если такого пользователя нету нас возвращает на основную страницу так же идет проверка на то что пользователь получил запрос в дружбу и при прохождение этих проверок мы принимаем этот запрос в друзья.

Функция postdelete позволяет удалять друзей :

public function postdelete($username){

$user = User::where('name', $username)->first();

if (!Auth::user()->isfriendwith($user)){

return redirect()->back();

}

Auth::user()->deletefriend($user);

return redirect()->back();

}

Здесь мы проверяем есть ли пользователь в друзьях и если его нету в ином случае просто удаляем друга

Последняя функция в этом списке это профиль друзей getprofile и её функционал реализован в контроллере следующим образом

public function getprofile($username){

$user = User::where('name', $username)->first();

if (!$user){

abort(404);

}

return view('profile.index',compact('user'));

}

В этом коде мы находим имя пользователя и переходим на его страницу.

Пред последняя функция позволяет стать администратором и для этого нам нужно добавить на HTML ссылку для обновления прав пользователя



Рис.15.Обновление прав пользователя

Для того что бы эта ссылка нужно в web.php прописать следующий роут который будет вызывать нужную функцию

Route::get('/admix/{userId}', [rcontroller::class,'admin'])->name('adminx');

Добавим для этой ссылки функционал следующим образом в контроллере пропишем:

public function admin(Request $request,$userId){

$user = User::find($userId);

$user->admin = 1;

$user->save();

return redirect()->back()->with('success', 'Пользователь успешно забанен.');

}

И последняя группа функций сделана для функционала админ панели где администратор может увидеть всех пользователей,редактировать их аккаунты,забанить и разбанить пользователя.  
 Для того что бы реализовать весь этот функционал нужно добавить в файл web.php следующие роуты:

Route::view('/admin', 'admin')->name('admin')->middleware('auth');

Route::get('/admin', [rcontroller::class,'showAllUsersadmin']);

Route::post('/admin', [rcontroller::class, 'profilestoreadmin'])

->name('admin.profile.store');

Route::post('/admin/update/{userId}', [rcontroller::class, 'profilestoreadmin'])

->name('admin.profile.update');

Route::post('/admin/banned/{userId})', [rcontroller::class,'banned'])

->name('admin.profile.banned');

Route::post('/admin/unbanned/{userId})', [rcontroller::class,'unbanned'])

->name('admin.profile.unbanned');

Здесь у нас есть путь для 5 функций для админки где переменная userId передается при помощи url.

Первая функция showAllUsersadmin нужна для показа всех пользователей это самая простоя из всех перечисленных .

public function showAllUsers()

{

$users = User::all();

return view('admin', ['users' => $users]);

}

В этой функции мы возвращаем переменную с пользователями на страницу админ

Вторая функция в админке profilestoreadmin нужная для того что бы изменять профили пользователей в большой части она похожа на другую функцию profilestore .

public function profilestoreadmin(Request $request,$userId)

{

$user = User::find($userId);

$request->validate([

'name' => 'required',

'email' => 'required',

'confirm\_password' => 'required\_with:password|same:password',

'avatar' => 'image',

]);

$input = $request->all();

if ($request->hasFile('avatar')) {

$avatarName = time().'.'.$request->avatar->getClientOriginalExtension();

$request->avatar->move(public\_path('avatars'), $avatarName);

$input['avatar'] = $avatarName;

} else {

unset($input['avatar']);

}

if ($request->filled('password')) {

$input['password'] = Hash::make($input['password']);

} else {

unset($input['password']);

}

$user->update($input);

return back()->with('success', 'Profile updated successfully.');

}

Третьей функцией для админ панели будет кнопка бана пользователей с её помощи адмистрация сможет быстро и легко банить пользоватлей.

public function banned(Request $request, $userId){

$user = User::find($userId);

$user->banned = 1;

$user->save();

return redirect()->back()->with('success', 'Пользователь успешно забанен.');

}

В этой функции мы меняем значение banned в базе данных пользователя где 1 значит что пользователь забанен ,а если колонка пустая или не равна 1 то пользователь не забанен

Последней функцией в административной панеле это кнопка разбана пользователя где адмистратор сможет оперативно снять с человека бан

public function unbanned(Request $request, $userId){

$user = User::find($userId);

$user->banned = 0;

$user->save();

return redirect()->back()->with('success', 'Пользователь успешно разбанен.');

}

После нажатии на кнопку пользователь сможет снова посещать наш сайт и пользоваться его функционалом.

Для того что бы вывести страницу оповещающая пользователя о том что она забанен нужно вписать в web.php :

Route::view('/banned', 'banned')->name('banned');

## **4 Загрузка на хостинг**

Backend составляющая полностью готова. Осталось загрузить файлы сайта на сервер, получить для него домен, прикрутить его к серверу и запустить сам сайт на хостинге.

Домен был получен на сервисе beget.com и вся настройка будет происходить там.

Предоставленный домен - <http://nazarkrx.beget.tech>.

В самом beget нужно зайти на аккаунт открыть консоль проекта и в нём скачать мой проект с github с помщью команды git clone “ssh ссылка на репозиторий”,далее нужно создать символическую ссылку которая будет будет переводить с сайт с папки public\_html на папку public.

Теперь нам нужно скачать composer.phar с сайта php и теперь нужно в консоль проекта нужно ввести:php8.3 composer.phar install и запустить его это позволит скачать на проект свой composer**.**

Нужно создать файл .env для этого нам понадобится вписать в консоль следующую команду: cp .env.example .envи php8.3 artisan key:generate .

Далее создать на сайте beget свою базу данных где нам нужно сохранить имя базы данных и пароль.

Последним действием нам нужно внести все данные в файл .env где нужно изменить параметр APP\_DEBUG= на false и внести данные для подключения к базе данных(имя бд и пользователь бд одинаковые).

Теперь мы можем зайти на наш сайт и увидеть, что все полностью работает и функционирует как нужно.

# заключение

В ходе выполнения курсового проекта была разработана социальная сеть с минимальны набором функций необходимых для её функционирования.Основной целью данного проекта было создание интерфейса для пользователя,позволяющая ему просматривать других пользователей добавлять их в друзья и писать в чат.

Были решены такие задачи как

1. **Разработка структуры базы данных**:
   * Была спроектирована и реализована оптимальная структура базы данных, учитывающая все необходимые таблицы, связи и поля для хранения информации о пользователях, друзьях и чатах.
2. **Функционал регистрации и авторизации**:
   * Была реализована возможность регистрации новых пользователей с сохранением их данных в базе данных и последующей авторизации на сайте.
3. **Добавление в друзья**:
   * Пользователи могут устанавливать связи друг с другом, добавляя пользователей в друзья. Этот функционал был успешно интегрирован с соответствующим учетом связей в базе данных.
4. **Просмотр других пользователей**:
   * Реализован возможность просмотра профилей других пользователей с отображением их информации на сайте.
5. **Редактирование профиля**:
   * Пользователи получили функционал для редактирования своего профиля, включая изменение персональных данных, фотографий ,имени,почты и т.д.
6. **Приватный чат и общий чат**:
   * Был реализован функционал приватного чата между пользователями, а также общего чата для общения в группах или общественности.

В результате выполнения данного курсового проекта была достигнута основная цель – разработка социальной сети с функционалом необходимом для общения между пользователями. Все поставленные задачи были успешно выполнены.

Разработанная система успешно обеспечивает удобство и функциональность для пользователей, позволяя им легко управлять своими друзьями и профилем. В ходе выполнения проекта были применены современные подходы и технологии, что гарантирует надежность и масштабируемость системы.

Таким образом, все цели, поставленные в рамках данной курсовой работы, были достигнуты. Полученные результаты демонстрируют успешное выполнение проекта и подтверждают возможность дальнейшего многократного расширения и улучшения социальной сети в будущем.

# используемые источники

Джоэл С, Джефф Э решение проблем / Джоэл С, Джефф Э [Электронный ресурс] // Stack Overflow : [сайт]. — URL: stackoverflow.com (дата обращения: 9.06.2024).

Алексей Буточников Как развернуть Laravel на хостинге Beget / Алексей Буточников [Электронный ресурс] // https://vk.com/ : [сайт]. — URL: https://vk.com/video19465981\_456246119 (дата обращения: 14.06.2024).

Влад Неверов Социальная сеть / Влад [Электронный ресурс] // https://www.youtube.com/ : [сайт]. — URL: https://www.youtube.com/watch?v=\_tDRtjvrysY&ab\_channel=VladNeverov (дата обращения: 10.06.2024).

Тэйлор Отвел Документация / Тэйлор Отвел [Электронный ресурс] // laravel.com : [сайт]. — URL: https://laravel.su/docs/11.x/ (дата обращения: 5.06.2024).

# Приложения

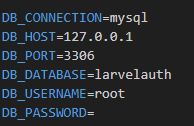


Рис.1.env

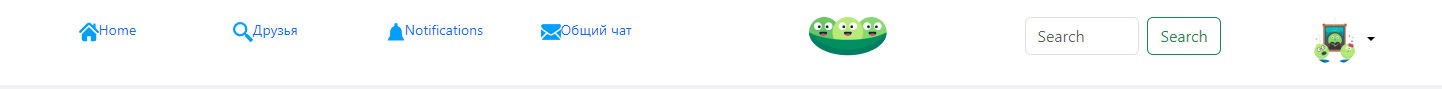


Рис.2. navbar

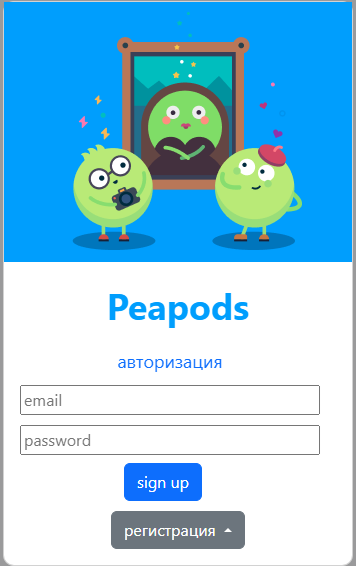


Рис.3.авторизация-регистрация

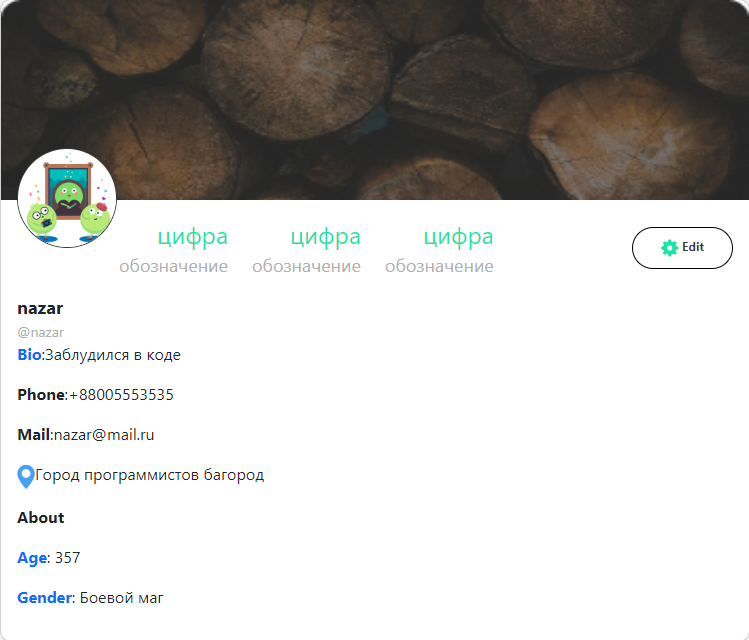


Рис.4 профиль пользователя

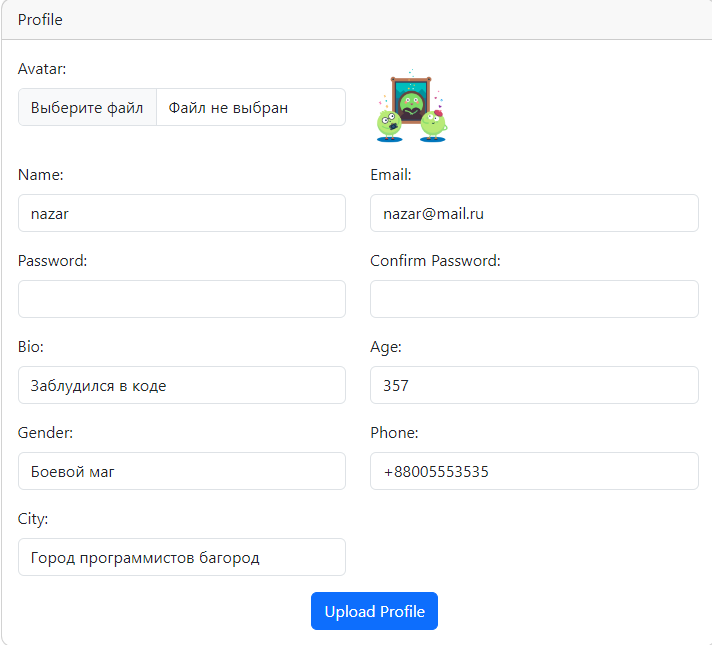


Рис.5.редактирование пользователя

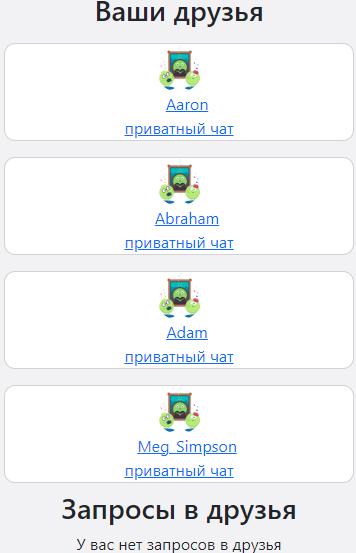


Рис.6.друзья

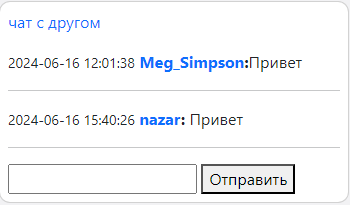


Рис.7.Приватный чат



Рис.8.Выдача админки

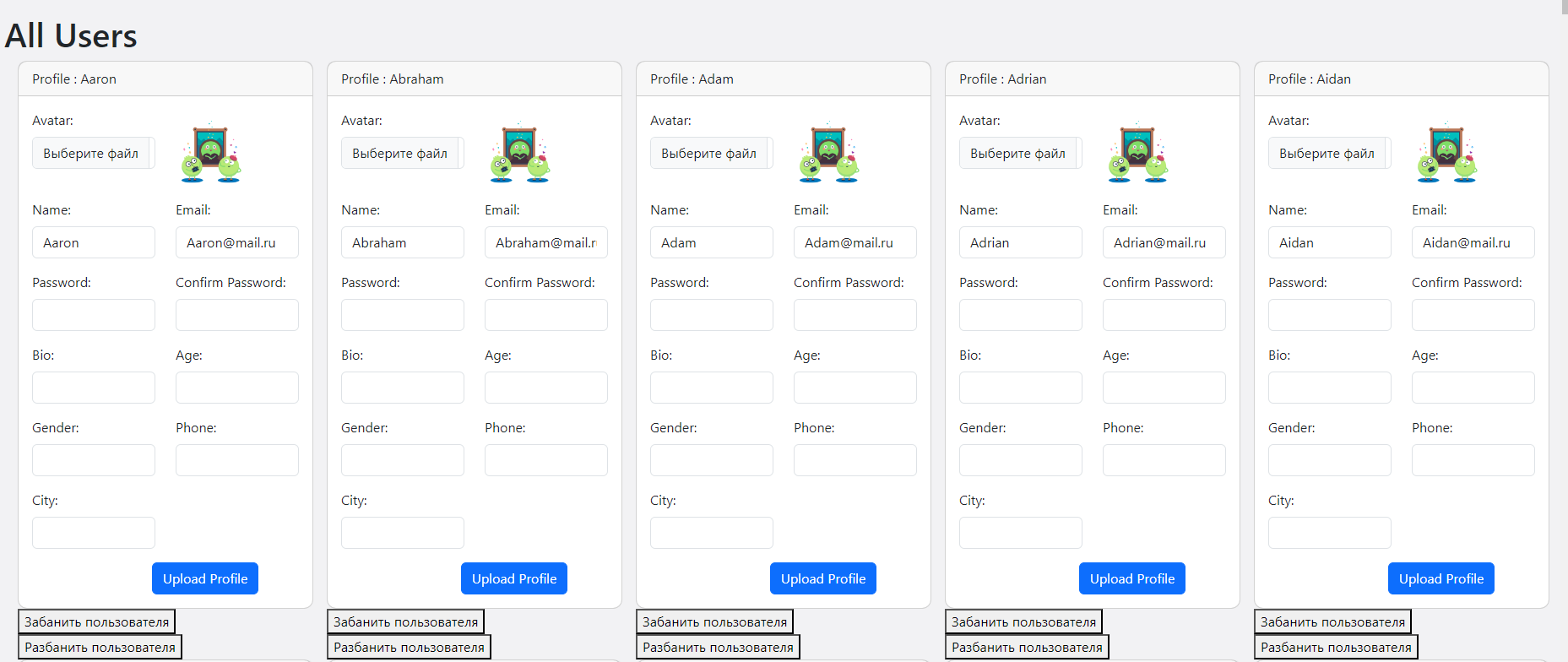


Рис.9.админ панель

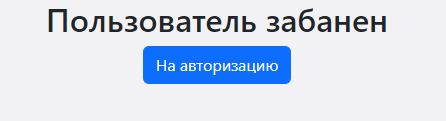


Рис.10.Страничка бана



Рис.11.создание контроллера



Рис.12.создание миграции

public function up(): void

{

Schema::create('users', function (Blueprint $table) {

$table->id();

$table->string('name');

$table->string('email')->unique();

$table->string('password');

$table->boolean('banned')->default(0);

$table->timestamps();

});

Route::get('register', [rcontroller::class,'create'])->name('register');

Route::post('register', [rcontroller::class, 'authstore']);

Route::post('/welcome', [rcontroller::class, 'store'])->name('login');

Route::get('/logout', [rcontroller::class, 'destroy'])->name('logout');

public function authstore(Request $request)

{

$request ->validate([

'name' => ['required', 'string'],

'email' => ['required', 'string', 'email', 'unique:users'],

'password' => ['required', 'confirmed', 'min:4']

]);

$user = User::create([

"name"=> $request->name,

"email"=> $request->email,

"password"=> hash::make($request->password)

]);

auth::login($user);

return redirect('osn');

}

public function store(Request $request)

{

$request ->validate([

'email' => ['required', 'string'],

'password' => ['required', 'string']

]);

if (Auth::attempt($request->only('email','password'))) {

$user = User::find(auth()->id());

if($user->banned == 1){

Auth::logout();

return redirect('banned');

}

else { redirect('osn');}

}

else {

return back();

}

}

public function destroy(Request $request){

Auth::logout();

return redirect()->route('welcome');

}

Route::post('/profile',[rcontroller::class,'profilestore'])

->name('user.profile.store');

public function profilestore(Request $request)

{

$request->validate([

'name' => 'required',

'email' => 'required',

'confirm\_password' => 'required\_with:password|same:password',

'avatar' => 'image',

]);

$input = $request->all();

if ($request->hasFile('avatar')) {

$avatarName = time().'.'.$request->avatar

->getClientOriginalExtension();

$request->avatar->move(public\_path('avatars'), $avatarName);

$input['avatar'] = $avatarName;

} else {

unset($input['avatar']);

}

if ($request->filled('password')) {

$input['password'] = Hash::make($input['password']);

} else { unset($input['password']);}

auth()->user()->update($input);

return back()->with('success', 'Profile updated successfully.');

}



Рис.13.Обновление профиля

public function up(): void

{

Schema::create('messages', function (Blueprint $table) {

$table->id();

$table->unsignedBigInteger('user\_id');

$table->text('content');

$table->string('chatroom')->nullable();

$table->timestamps();

});

}

Route::get('/chat', [rcontroller::class,'chatindex'])->name('chat');

Route::post('/send-message', [rcontroller::class,'sendMessage'])->name('send-message');

public function chatindex()

{

$messages = Message::with('user')->orderBy('created\_at')->get()

->whereNull('chatroom');

return view('chat', compact('messages'));

}

public function sendMessage(Request $request)

{

$message = new Message();

$message->user\_id = auth()->id();

$message->content = $request->content;

$message->save();

return redirect()->back();

}

Route::view('/friend', 'friend')->name('friend')->middleware('auth');

Route::get('/friend', [rcontroller::class,'friendindex'])->name('friendindex')

->middleware('auth');

Route::get('/friend/add/{username}', [rcontroller::class,'getadd'])

->name('friend.add')->middleware('auth');

Route::get('/friend/accept/{username}', [rcontroller::class,'getaccept'])

->name('friend.accept')->middleware('auth');

Route::post('/friend/delete/{username}', [rcontroller::class,'postdelete'])

->name('friend.delete')->middleware('auth');

Route::get('/user/{username}',[rcontroller::class,'getprofile'])

->name('profile.index');

public function friendindex(){

$friends = Auth::user()->friends();

$requests= Auth::user()->friendrequests();

return view ('friend',[

'friends'=>$friends,

'requests'=>$requests]);

public function getadd($username){

$user = User::where('name', $username)->first();

if (!$user){

return redirect()->route('osn')->with('info','пользователь не найден');

}

if (Auth::user()->hasfriendrequestspending($user)

|| $user->hasfriendrequestspending( Auth::user() ) ){

return redirect()

->route('profile.index',['username'=>$user->name])

->with('info','пользователю отправлен запрос в друзья');

}

if (Auth::user()->isfriendwith($user)){

return redirect()

->route('profile.index',['username'=>$user->name])

->with('info','пользователь уже в друзьях');

}

Auth::user()->addfriend($user);

return redirect()

->back()

->with('info','пользователю отправлен запрос в друзья');

}

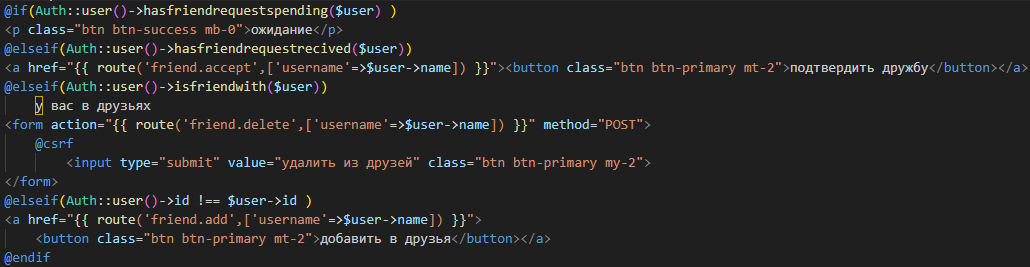


Рис.14.Код для кнопки отображения друзей

public function getaccept($username){

$user = User::where('name', $username)->first();

if (!$user){

return redirect()->route('osn')->with('info','пользователь не найден');

}

if (! Auth::user()->hasfriendrequestrecived($user)){

return redirect()->route('osn');

}

Auth::user()->acceptfriendrequest($user);

return redirect()

->route('search.results',['username'=>$user->name])

->with('info','запрос в друзья принят');

}

public function postdelete($username){

$user = User::where('name', $username)->first();

if (!Auth::user()->isfriendwith($user)){

return redirect()->back();

}

Auth::user()->deletefriend($user);

return redirect()->back();

}

public function getprofile($username){

$user = User::where('name', $username)->first();

if (!$user){

abort(404);

}

return view('profile.index',compact('user'));

}



Рис.15.Обновление прав пользователя

Route::get('/admix/{userId}', [rcontroller::class,'admin'])->name('adminx');

public function admin(Request $request,$userId){

$user = User::find($userId);

$user->admin = 1;

$user->save();

return redirect()->back()->with('success', 'Пользователь успешно забанен.');

Route::view('/admin', 'admin')->name('admin')->middleware('auth');

Route::get('/admin', [rcontroller::class,'showAllUsersadmin']);

Route::post('/admin', [rcontroller::class, 'profilestoreadmin'])

->name('admin.profile.store');

Route::post('/admin/update/{userId}', [rcontroller::class, 'profilestoreadmin'])

->name('admin.profile.update');

Route::post('/admin/banned/{userId})', [rcontroller::class,'banned'])

->name('admin.profile.banned');

Route::post('/admin/unbanned/{userId})', [rcontroller::class,'unbanned'])

->name('admin.profile.unbanned');

public function showAllUsers()

{

$users = User::all();

return view('admin', ['users' => $users]);

}

{

$user = User::find($userId);

$request->validate([

'name' => 'required',

'email' => 'required',

'confirm\_password' => 'required\_with:password|same:password',

'avatar' => 'image',

]);

$input = $request->all();

if ($request->hasFile('avatar')) {

$avatarName = time().'.'.$request->avatar->getClientOriginalExtension();

$request->avatar->move(public\_path('avatars'), $avatarName);

$input['avatar'] = $avatarName;

} else {

unset($input['avatar']);

}

if ($request->filled('password')) {

$input['password'] = Hash::make($input['password']);

} else {

unset($input['password']);

}

$user->update($input);

return back()->with('success', 'Profile updated successfully.');

}

public function banned(Request $request, $userId){

$user = User::find($userId);

$user->banned = 1;

$user->save();

return redirect()->back()->with('success', 'Пользователь успешно забанен.');

}

public function unbanned(Request $request, $userId){

$user = User::find($userId);

$user->banned = 0;

$user->save();

return redirect()->back()->with('success', 'Пользователь успешно разбанен.');

}

Route::view('/banned', 'banned')->name('banned');