**PAŃSTWOWA WYŻSZA SZKOŁA ZAWODOWA   
W NOWYM SĄCZU**

**INSTYTUT TECHNICZNY**

**PRACA DYPLOMOWA**

**PORTAL WSPOMAGAJĄCY KOMUNIKACJĘ   
ZE STUDENTAMI**

**Autor: Jakub Bogdański**

**Kierunek: Informatyka stosowana**

**Nr albumu: 26050**

**Promotor: dr hab. Witold Przygoda**

**NOWY SĄCZ 2019**

Spis treści.

[1. Wstęp. 4](#_Toc30411421)

[1.1. Uzasadnienie wyboru tematu. 4](#_Toc30411422)

[1.2. Cel i zakres pracy. 5](#_Toc30411423)

[1.3. Przedstawienie użytych narzędzi. 6](#_Toc30411424)

[2. Wzorzec projektowy MVC. 7](#_Toc30411425)

[2.1. Model. 7](#_Toc30411426)

[2.2. Widok. 7](#_Toc30411427)

[2.3. Kontroler. 7](#_Toc30411428)

[3. Baza danych. 8](#_Toc30411429)

[3.1. Schemat bazy danych. 8](#_Toc30411430)

[3.2. Połączenie z bazą danych. 9](#_Toc30411431)

[3.3. Określenie relacji w bazie danych. 9](#_Toc30411432)

[3.4. Relacje polimorficzne. 10](#_Toc30411433)

[3.5. Tworzenie relacji polimorficznych. 10](#_Toc30411434)

[3.6. Migracje. 11](#_Toc30411435)

[3.7. Schemat migracji. 12](#_Toc30411436)

1. Wstęp.

W czasach postępującej informatyzacji oraz powszechnego dostępu do sieci internetowej, większość wiadomości, które ze sobą wymieniamy są w formie cyfrowej. Dobrym przykładem mogą być chociażby wiadomości e-mail lub portale społecznościowe cieszące się ogromną popularnością. Informatyzacja dosięgła również instytucje takie jak urzędy, szkoły i uczelnie wyższe.

W skład pracy wchodzą rozdziały:

Rozdział 2. Wzorzec projektowy MVC.

Rozdział 3. Baza danych.

Rozdział 4.

Rozdział 5.

Rozdział 6.

* 1. Uzasadnienie wyboru tematu.

Głównym powodem, dla którego zdecydowałem się realizować portal wspomagający komunikację ze studentami była potrzeba dostarczenia przejrzystego systemu wiadomości między wykładowcą i studentami. Istniejące rozwiązanie występują w niewielkiej skali oraz niekiedy brakuje im niektórych funkcjonalności.

Często na stronach uczelni wyższych istnieją strony prowadzone przez samych pracowników, w których sami muszą zmieniać kod pliku HTML. Rozwiązanie to sprawia, że strony te wyglądają niezbyt schludnie, szczególnie jeśli mówimy o instytutach nietechnicznych. Nierzadko wykładowcy mają minimalistyczne podejście do tychże stron, które tworzą jako listę linków, gdzie każdy ma do nich dostęp nawet osoby spoza uczelni.

Portal wspomagający komunikację ze studentami, który tworzę w ramach tej pracy dyplomowej sprawi, że wykładowcy nie będą musieli tracić czasu na edytowanie plików odpowiedzialnych za treść na stronie, a studentom da czytelny dostęp do materiałów z zajęć oraz dobrze działający system komunikacji.

* 1. Cel i zakres pracy.

Celem niniejszej pracy jest zaprojektowanie oraz implementacja portalu komunikacyjnego między prowadzącymi zajęcia i studentami.

Podstawowe funkcjonalności portalu:

- system kont użytkowników – w skład, którego wchodzą logowanie, rejestracja nowych użytkowników, edycja istniejących kont oraz blokowanie/usuwanie kont.

- tworzenie oraz zarządzanie grupami – prowadzący lub administrator będą mogli tworzyć grupy. Grupy te będą podzielone na: laboratoryjne, ćwiczeniowe, wykładowe. Użytkownicy z odpowiednimi uprawnieniami będą mogli definiować własne grupy. System grup będzie działał z systemem kontroli dostępu.

- system kontroli dostępu – użytkownicy będą podzieleni na odpowiadające im role tj. administrator, wykładowca, student. Każda z tych grup będzie posiadać inne uprawnienia na stronie. Administrator będzie mógł korzystać ze wszystkich opcji dostępnych na portalu. Wykładowca będzie mógł: tworzyć grupy, dodawać lub usuwać użytkowników z grup, wysyłać wiadomości do studentów, udostępniać ogłoszenia oraz pliki grupom. Student będzie mógł: wymieniać korespondencje z prowadzącym zajęcia, pobierać materiały przeznaczone dla niego lub grupy do której należy, czytać ogłoszenia przeznaczone dla niego

- prywatne tablice ogłoszeń – system wiadomości między wykładowcą, a studentem. System pozwoli na powiadomienia przez e-mail oraz wysyłanie dowolnych załączników w wiadomości.

- grupowe tablice ogłoszeń – komunikacja prowadzącego zajęcia z grupą studentów. Prowadzący będzie miał możliwość udostępniania materiałów oraz ogłoszeń dla całej grupy.

- publiczne ogłoszenia – wykładowca będzie mógł udostępnić informację lub pliki dla wszystkich użytkowników systemu.

Dodatkowe funkcjonalności portalu:

- wyszukiwarka użytkowników – działająca asynchronicznie wyszukiwarka pozwalająca wykładowcy wyszukać studentów według numeru albumu lub danych osobowych.

- profil użytkownika – pozwala zobaczyć podstawowe dane na temat wybranego użytkownika

* 1. Przedstawienie użytych narzędzi.

Do stworzenia niniejszej pracy wykorzystano podstawowe technologie wykorzystywane przy tworzeniu stron internetowych takie jak: PHP, HTML, MySQL, AJAX, CSS, JavaScript. Dodatkowo zostały użyte rozszerzenia: Laravel oraz Bootstrap.

PHP – popularny, skryptowy język programowania witryn internetowych. Duża część składni PHP wywodzi się z C, Javy oraz języka Perl. Technologia ta pozwala na dynamiczne generowanie stron internetowych. PHP działa po stronie serwera.

HTML – język znaczników służący do tworzenia stron internetowych. Jedno z najbardziej podstawowych narzędzi wykorzystywanych przy tworzeniu witryn. HTML odpowiada za strukturę strony internetowej. Standardy języka ustalane są przez specyfikację W3C. Aktualną wersją jest HTML5.

CSS – jest językiem wykorzystywanym do opisywania wyglądu stron internetowych. Podobnie do HTML jest opisany przez standardy W3C.

JavaScript – skryptowy język programowania służący głównie do tworzenia stron internetowych. Kod JavaScriptu wykonywany jest po stronie klienta i jest on interpretowany przez przeglądarkę internetową

AJAX – technologia pozwalająca na tworzenie asynchronicznych stron internetowych. AJAX pozwala na tworzenie stron, które odświeżają się automatycznie bez konieczności ingerencji użytkownika.

MySQL – strukturalny język zapytań rozwijany przez firmę Oracle. Służy do zarządzania relacyjnymi bazami danych. MySQL jest udostępniany na licencji GNU GPL.

Bootstrap – jedno z najpopularniejszych rozszerzeń służących do tworzenia wyglądu stron internetowych. Bootstrap ułatwia tworzenia responsywnych portali. Dodatkowym jego atutem jest otwarto źródłowość.

Laravel – system stworzony w języku PHP ułatwiający tworzenie portali internetowych. Jego główną cechą jest wykorzystanie wzorca Model View Controller.

1. Wzorzec projektowy MVC.

Wzorzec projektowy MVC (z ang. Model-View-Controller) pozwala w sposób przejrzysty zorganizować zarys aplikacji. Wzorzec ten jest stosowany w przypadku tworzenia aplikacji z graficznym interfejsem użytkownika. Założenia MVC często wykorzystywane są podczas tworzenia portali internetowych, wówczas mają one nieco zmienioną formę. Poprzez zastosowanie technologii HTTP przy tworzeniu stron nie ma możliwości użycia aktywnych modeli.

* 1. Model.

We wzorcu MVC model jest odpowiedzialny za przechowywanie danych, które są wykorzystywane przez kontroler. Modele najczęściej przedstawione są za pomocą klas danego języka programowania.

* 1. Widok.

Zadaniem widoku jest obsługa graficznego interfejsu użytkownika. W skład widoku mogą wchodzić tzw. widoki częściowe, odpowiadają one za odpowiednie fragmenty strony. Widok komunikuje się z kontrolerem poprzez przekazywanie danych z formularzy oraz pobieranie żądania odświeżenia. Widok w żaden sposób nie powinien modyfikować danych.

* 1. Kontroler.

Ostatnim elementem wzorca projektowego MVC jest kontroler. Kontroler odbiera, analizuje oraz przetwarza dane wysłane przez użytkownika za pomocą widoku. Kontroler wchodzi w interakcję z modelem poprzez wysyłanie danych do niego. Kontroler ma bezpośrednie powiązanie z modelem i widokami, z którymi kooperuje. W przypadku aplikacji internetowych zgodnie ze wzorcem MVC kontroler może wysyłać zapytania do bazy danych.

1. Baza danych.

Portal umożliwiający komunikację ze studentami korzysta z bazy danych zarządzanej za pomocą systemu zarządzania bazami danych MySQL. Warto odnotować, że relacje występujące w bazie danych zdefiniowane są na poziomie aplikacji w tak zwanych modelach.

* 1. Schemat bazy danych.

Tabele bazy danych oraz relacje pomiędzy nimi zostały przedstawione na schemacie poniżej(Rysunek 1).



Rysunek 1. Diagram ERD. (DO ZMIANY/TYMCZASOWY)

* 1. Połączenie z bazą danych.

Konfiguracja połączenia z bazą danych odbywa się poprzez rozszerzenie PHP – Laravel. W tym rozszerzeniu wszelakie ustawienia bazy danych znajdują się w pliku .env. Za połączenie z bazą danych odpowiada następujący fragment kodu:

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=portal

DB\_USERNAME=root

DB\_PASSWORD=

Na powyższym przykładzie można zauważyć, że konfiguracja odbywa się za pomocą zmiennych środowiskowych.

* 1. Określenie relacji w bazie danych.

Relacje między poszczególnymi tabelami zostały określone w migracjach oraz w modelach za pomocą odpowiednich metod. Do zdefiniowania relacji należy utworzyć metodę w modelu, która zwraca wynik metody odpowiedzialnej za typ relacji, kolejno dla relacji:

- jeden do jednego – korzysta z metody hasOne, odwrotność relacji belongsTo.

- jeden do wielu – używa metody hasMany, odwrotność relacji belongsTo.

- wiele do wielu – korzysta z metody belongsToMany zarówno dla określenia relacji jak i jej odwrotności.

Każda z powyższych metod przyjmuje jako parametr podstawowy odwołanie do modeli, z którymi zachodzi relacja. Dodatkowo możemy podać parametry jeśli chcemy skorzystać ze zdefiniowanych przez siebie kluczy. Metoda belongsToMany może przyjmować dodatkowo nazwę dla tabeli łączącej.

Jeżeli nie korzystamy z dodatkowych parametrów należy pamiętać o odpowiednim nazewnictwie tabel oraz jej kolumn.

Na poniższym przykładzie można zauważyć sposób określania relacji. Za przykład posłuży fragment modelu odpowiedzialnego za użytkownika – User.php, w którym została zdefiniowana relacja typu „wiele do wielu” z grupami.

public function groups()

{

return $this->belongsToMany('App\Group');

}

Podobnie prezentuje się fragment odpowiedzialny za odwrotność tej relacji. W modelu grup - Group.php korzystamy z tej samej metody belongsToMany, która przyjmuje model User jako parametr.

public function users()

{

return $this->belongsToMany('App\User');

}

* 1. Relacje polimorficzne.

Szczególnym typem relacji występującym w bazie danych są tak zwane relacje polimorficzne. W niniejszym projekcie relacje polimorficzne wykorzystywane są do obsługiwania plików. Użytkownik korzystający z aplikacji może załączyć plik do wiadomości lub postu. Zastosowanie relacji polimorficznej sprawia, że zamiast dwóch tabel dla plików z wyróżnieniem relacji(np. file\_post, file\_message), można stworzyć jedną tabelę, która będzie posiadała informację, z którym modelem będzie plik będzie powiązany.

Poniższa tabela(Rysunek 2) przedstawia strukturę tabeli odpowiedzialnej za pliki.



Rysunek 2. Struktura tabeli o nazwie file.

Tabela w relacji polimorficznej posiada dodatkowe pola wskazujące id modelu jego typ. W bazie danych jako typ przechowywana jest nazwa modelu z przedrostkiem ‘App\’, na przykład ‘App\Post’.

* 1. Tworzenie relacji polimorficznych.

Tworząc relacje polimorficzne należy stworzyć odpowiednie kolumny w bazie danych. Odpowiednio przygotowaną tabelę można zobaczyć na Rysunku 2.

Tworzenie relacji tego typu zostało opisane na podstawie relacji tabel odpowiedzialnych za pliki, posty oraz wiadomości niniejszego projektu

W modelu File należy utworzyć metodą zwracającą wynik metody morphTo.

public function filetest()

{

return $this->morphTo();

}

W modelach Message oraz Post metodą odpowiadająca za polimorficzną relację jest metoda morphMany, która przyjmuje jako parametry model oraz nazwę metody z tego modelu odpowiedzialną za definicje relacji.

public function files()

{

return $this->morphMany('App\File', 'filetest');

}

Odwołanie do relacji odbywa się podobnie jak w przypadku normalnej relacji.

* 1. Migracje.

Migracje w dodatku Laravel działają na podobnej zasadzie jak systemu kontroli wersji. Pozwalają w łatwy sposób zarządzać schematem bazy danych.

Tworzenie migracji odbywa się za pomocą polecenia:

php artisan make:migration create\_files\_table

Za każdym razem przy wywołaniu tego polecenia do nazwy pliku z migracją dopisywana jest informacja o czasie utworzenia tejże migracji. Migracje wywoływane są w kolejności od najstarszej do najnowszej.

Wykonanie migracji odbywa się poleceniem:

php artisan migrate

Dodatkowo to polecenie może przyjmować flagę --force, wymusza ona wykonanie migracji nawet w przypadku gdy może to powodować utratę niektórych danych.

W przypadku błędu każdą migrację można cofnąć za pomocą polecenia:

php artisan migrate:rollback

Wszystkie migracje można wycofać następującym poleceniem:

php artisan migrate:reset

Polecenie php artisan migrate:refresh działa na takiej samej zasadzie jak reset, z tą różnicą, że po resecie wykonywana jest opcja migrate.

Usuwanie wszystkich tabeli a następnie wykonanie migracji odbywa się za pomocą polecenia: php artisan migrate:fresh

* 1. Schemat migracji.

Migracje mają format klasy, która rozszerza klasę Mirgation. Stworzony plik migracji domyślnie składa się z dwóch metod: up, down. Pierwsza z nich wywoływana jest przy okazji uruchomienia migracji, natomiast druga w przypadku cofanie tej migracji.

Migracje powiązanie są z fasadą Schema, która dostarcza narzędzia do zarządzania tabelami bazy danych.

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

$table->bigIncrements('id');

$table->char('title', 64);

$table->string('content');

$table->integer('from\_id')->unsigned();

$table->integer('to\_id')->unsigned();

$table->dateTime('date');

$table->boolean('status');

$table->foreign('from\_id')->references('id')->on('users');

$table->foreign('to\_id')->references('id')->on('users');

});

Powyższy kod przedstawia zawartość metody up dla pliku migracji odpowiedzialnego za tworzenie tabeli messages.

Na powyższym przykładzie można zauważyć sposób tworzenia tabeli oraz jej kolumn. Tworzenie tabeli odbywa się za pomocą metody create, która jaka parametry przyjmuję nazwę tworzonej tabeli oraz funkcję, która przyjmuje jako parametr przyjmuje obiekt typu Blueprint, który jest używany do określenia nowej tabeli.

Do utworzenia kolumn należy wywołać metodę elementu typu Blueprint. Najczęściej używane metody to:

* $table->bigIncrements(‘nazwa’) – tworzenie klucza głównego typu UNSIGNED BIG INT
* $table->char(‘nazwa’, ilość znaków) – pole znaków o określonej długości
* $table->string(‘nazwa’) – typ tekstowy
* $table->integer(‘nazwa’) - określa liczbę całkowitą, może dodatkowo wywoływać metodę unsigned, która oznacza liczby tylko nieujemne
* $table->dateTime(‘nazwa’) – pole daty i czasu.
* $table->boolean(‘nazwa’) -> pole przyjmujące wartości true lub false

W przypadku tworzenia relacji polimorficznej należy skorzystać z metody $table->morphs(‘nazwa’). Tworzy ona automatycznie pola „nazwa\_type” oraz „nazwa\_id” określające model oraz klucz obcy do tego modelu.

Podczas korzystania z migracji tworzącej relację polimorficzną należy spełnić określone wymagania serwera. W przypadku korzystania z MySQL wersja musi być nowsza niż 5.7.7.

W przeciwnym razie otrzymamy następujący błąd:

SQLSTATE[42000]: Syntax error or access violation: 1071 Specified key was too long; max key length is 767 bytes

Jego rozwiązaniem może być aktualizacja MySQL do najnowszej wersji lub ręczne ustawienie domyślnej długości zmiennej typu tekstowego.

W celu określenia relacji na poziomie bazy danych należy skorzystać z metody

$table->foreign():

$table->foreign('from\_id')->references('id')->on('users');

Powyższy kod pokazuje sposób określania klucza obcego dla kolumny o nazwie „from\_id”. Należy wskazać do jakiej kolumny odnosi się wybrana kolumna w obcej tabeli.

Dodatkowo można określić zachowanie klucza obcego podczas usuwania lub zmiany wartości w tabeli. W tym celu należy zastosować metodę onDelete lub onUpdate, która za parametr przyjmuje określone zachowanie.

$table->foreign('user\_id')->references('id')->on('users')->onDelete('set null');

Powyższy przykład opisuje ustawienie wartości null przy usunięciu użytkownika. Należy pamiętać, żeby klucz obcy był ustawiony jak nullable.

Metody onDelete oraz onUpdate mogą przyjmować wartości znane z systemu zarządzania bazą danych MySQL, to jest:

* Cascade – usuwa element powiązany
* Set null – ustawia wartość null dla klucza obcego
* Restrict – oznacza odrzucenie wykonywanej akcji

1. Uprawnienia użytkowników.

Użytkownicy korzystający z systemu przypisani są do grup różniącymi się poziomami dostępu do funkcjonalności dostępu.

* 1. Hierarchia użytkowników.

Najniższy poziom dostępu do informacji na stronie internetowej będą mieli użytkownicy niezalogowani tak zwani „goście”. Goście będą mieli dostęp jedynie do plików i materiałów oznaczonych przez udostępniającego jako publiczne.

Kolejni w hierarchii są studenci. Podczas rejestracji będą mogli oni dołączyć do grupy wykładowej odpowiadającej ich specjalizacji. Oprócz tego mogą wysyłać wiadomości wraz z załącznikami do innych użytkowników systemu. Użytkownicy zalogowani mogą wybrać czy chcą być powiadamiani drogą mailową o nowej wiadomości w systemie. Oprócz komunikacji z innymi użytkownikami student może zobaczyć do jakich grup należy oraz posty przeznaczone dla tych grup. Oprócz tego mogą zobaczyć pliki, które zostały do nich wysłane w postach bez konieczności ich przeszukiwania.

Grupą z wyższymi uprawnieniami jest grupa prowadzących zajęcia. Oprócz przywilejów przypisanych do studentów mogą oni dodatkowo definiować własne grupy oraz dodawać do nich użytkowników. Mogą oni również wyrzucać wybrane osoby z grupy oraz usuwać całe grupy.

Najwyższy poziom uprawnień mają administratorzy. Oprócz uprawnień niższych grup otrzymują oni dodatkowo dostęp do zarządzania uprawnieniami użytkowników. Domyślnie każdy zarejestrowany użytkownik ma rolę studenta. Administrator systemu może ją zmienić. Dodatkowo mają oni podgląd wszystkich grup.

* 1. Rozwiązanie technologiczne.

System uprawnień oparty jest na podstawie zmiennej w tabeli użytkowników. Zmienna ta przyjmuje wartości całkowite od 1 do 3. Każda z nich odpowiada innemu uprawnieniu, kolejno:

* 1 – student
* 2 – prowadzący
* 3 – administrator

Sprawdzanie roli użytkownika odbywa się za pomocą tak zwanych Middleware. Middleware to mechanizm pozwalający filtrować wywołania użytkownika.

Żeby utworzyć klasę middleware należy wywołać następujące polecenie:

php artisan make:middleware RoleTeacher

Polecenie to utworzy plik z klasą RoleTeacher z metodą handle przyjmującą jako parametry: żądanie użytkownika oraz następną stronę.

Poniższy kod przedstawia metodę handle dla middleware RoleTeacher.

public function handle($request, Closure $next)

{

if(Auth::user()->role>=2)

return $next($request);

else

return redirect()->back();

}

Metoda ta odpowiada za sprawdzanie czy zalogowany użytkownik ma co najmniej uprawnienia prowadzącego zajęcia. W przypadku gdy zalogowana osoba jest prowadzącym lub administratorem system pozwala przejść użytkownikowi do żądanej przez niego podstrony. W przeciwnym razie następuje przekierowanie do poprzedniej strony.

W przypadku gdy należy sprawdzić czy zalogowany użytkownik jest administratorem należy utworzyć odpowiednią klasę. Wówczas metoda handle będzie wyglądać podobnie do tej z powyższego przykładu z tą różnicą, że warunkiem będzie równość z liczbą 3. Poniżej znajduje się fragment kodu klasy IsAdmin

if(Auth::user()->role>=3)

return $next($request);

else

return redirect()->back();

Niektóre z podstron nie wymagają oceny uprawnień. Wtedy konieczne jest zabezpieczenie adresów przed dostępem użytkowników niezalogowanych. Należy wówczas utworzyć middleware sprawdzający czy użytkownik jest zalogowany.

php artisan make:middleware LoggedIn

Metoda handle klasy LoggedIn prezentuje się następująco:

public function handle($request, Closure $next)

{

if(Auth::Check()==true)

return $next($request);

else

return redirect()->back();

}

Na powyższym przykładzie warunkiem do przekierowania dalej jest sprawdzenie czy użytkownik jest zalogowany, odpowiada za to metoda Auth::check(), która zwraca wartości true, gdy użytkownik jest zalogowany lub false, w przeciwnym przypadku.

Middleware dodatkowo mogą być wykorzystywane do zabezpieczenia adresów wywoływanych asynchronicznie. W takim przypadku należy utworzyć klasę, która sprawdzi czy aktualne żądanie zostało wywołane przez technologię AJAX.

public function handle($request, Closure $next)

{

if($request->ajax())

return $next($request);

else

return redirect()->back();

}

Powyższy kod przedstawia metodę handle klasy AjaxMiddleware. W warunku wywoływana jest metoda, która sprawdza czy żądanie wywołane jest asynchronicznie, jeśli tak odbywa się przekierowanie do żądanej podstrony, w przeciwnym razie użytkownik zostanie cofnięty do strony poprzedniej.

Rejestracja middleware odbywa się w pliku Kernel.php, który znajduje się pod ścieżką App\Http. Należy do tablicy o nazwie routeMiddleware dodać wpis z nową klasą. Przykładowy wpis prezentuje się następująco:

'ajax' => \App\Http\Middleware\AjaxMiddleware::class,

Przypisanie middleware do adresu url odbywa się w pliku web.php znajdującym się w folderze routes. Middleware można przypisać na dwa sposoby:

* Dla grupy adresów – wówczas korzystamy z klasy Route wywołując metodę middleware a następnie metodę group. Przykład przedstawia grupę tras wywoływanych przez technologię AJAX:

Route::middleware('ajax')->group(function () {

Route::get('/message/x/{id}', 'AjaxController@messages')->middleware('loggedin');

Route::get('/message/read/{type}/{id}', 'MessageController@read')->middleware('loggedin');

Route::get('/post/filter/{id}', 'AjaxController@posts');

Route::get('/users/x/{id}', 'AjaxController@users')->middleware('admin');

Route::get('/ajax/search{id}', 'AjaxController@search')->name('search');

Route::get('/ajax/user/search', 'AjaxController@searchuser')->name('searchuser');

Route::get('/userinfo/{id}', 'AjaxController@userinfo')->name('userinfo')->middleware('teacher');

Route::get('/group/add/{id}/{uid}', 'GroupController@addto')->name('groups.addto')->middleware('teacher');

Route::get('/message/add/{id}', 'MessageController@add')->name('message.add');

Route::get('/group/info/{id}', 'AjaxController@groups')->name('grouptest');

});

* Dla pojedynczego adresu – wtedy należy wywołać metodę middleware dla zarejestrowanej trasy

Route::post('/message/postreply/{type}/{id}', 'MessageController@postreply')->name('message.postreply')->middleware('loggedin');

* 1. Uprawnienia grup.

W przypadku uprawnień, grupa definiuje dostęp do wyświetlanych danych. Dostęp do plików i postów uzależniony jest od przynależności do grupy zdefiniowanej przez prowadzącego. Jeśli użytkownik nie należy do grupy nie będzie miał możliwości pobrania plików, ani odczytania postu.