NTW

Lista 3 – Zadanie 1 (50 pkt.)

Proszę przygotować aplikację WEB w wybranym frameworku MVC dla PHP. Aplikacja ma mieć formę systemu do testowania wiedzy za pośrednictwem testów jednokrotnego wyboru. Aplikacja ma być zrealizowana zgodnie z założeniami:

- W systemie jest jedno konto użytkownika typu nauczyciel, oraz dowolna ilość kont użytkowników typu uczeń. Nauczyciel może:
  - zarządzać swoim kontem,
  - zarządzać (dodawać, usuwać, edytować) uczniów,
  - grupować uczniów w klasy,
  - zarządzać własnym bankiem pytań oraz poprawnych dla nich odpowiedzi (pytaniami zarządzamy niezależnie od testów),
  - tworzyć zestawy pytań w postaci testu (pytania wybieramy z banku pytań) i przypisywać go do wybranych uczniów lub do całych klas,
  - przeglądać wyniki osiagnięte przez uczniów.
- uczeń loguje się do aplikacji, wybiera jeden z przeznaczonych dla niego i nie rozwiązanych jeszcze testów i odpowiada na przedstwione mu pytania, na końcu system wyjaśnia mu popełnione błędy i podsumowuje ilosc poprawnie rozwiązanych pytań.
- wyświetlając test aplikcja losuje kolejność zadawanych pytań oraz kolejność wyświetlanych odpowiedzi w każdnym z nich.

Uwaga!
Aplikacja nie musi „pięknie wyglądać” ale musi być wygodna i możliwie intuicyjna w obsłudze.

Legenda kolorów

Opisu zadań: GOTOWE WORK IN PROGRESS TODO

Rozwiązania: ZMIANA FRAGMENT KODU ZOSTAŁ OMINIĘTY

Tworzenie projektu

```
$ composer create-project laravel/laravel l3z1
```

```
$ cd l3z1/
```

```
$ php artisan breeze:install
```

Zmiany w sposobu używania template

```
resources/views/layouts/app.blade.php
```

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
```

```
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta name="csrf-token" content="{{ csrf_token() }}">

<title>{{ config('app.name', 'Laravel') }}</title>

<!-- Bootstrap -->
<link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">

<!-- Fonts -->
<link rel="preconnect" href="https://fonts.bunny.net">
<link href="https://fonts.bunny.net/css?family=figtree:400,500,600&display=swap" rel="stylesheet" />

<!-- Scripts -->
@vite(['resources/css/app.css', 'resources/js/app.js'])
</head>
<body class="font-sans antialiased">
<div class="min-h-screen bg-gray-100">
@include('layouts.navigation')

// usunięto kila linii niepotrzebnych

<!-- Page Content -->
<main>
@yield('content')
</main>
</div>
</body>
</html>
```

resources/views/dashboard.blade.php

```
@extends('layouts.app')

@section('content')
<div class="py-12">
<div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
<div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">
<div class="p-6 text-gray-900">
Jesteś zalogowany.
</div>
```

```
</div>
</div>
</div>
@endsection
```

Zarządzanie użytkownikami

Rozbudowa modelu User

```
$ php artisan make:migration add_role_to_users_table --table=users
```

💻 database/migrations/2024_04_25_195820_add_role_to_users_table.php

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
/**
* Run the migrations.
*/
public function up(): void
{
Schema::table('users', function (Blueprint $table) {
$table->string('role')->default('student');
});
}

/**
* Reverse the migrations.
*/
public function down(): void
{
Schema::table('users', function (Blueprint $table) {
$table->dropColumn('role');
});
}
};
```

```
$ php artisan migrate
```

app/Models/User.php

```php
<?php
namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
use HasFactory, Notifiable;

/**
* The attributes that are mass assignable.
*
* @var array<int, string>
*/
protected $fillable = [
'name',
'email',
'password',
'role',
];

/**
* The attributes that should be hidden for serialization.
*
* @var array<int, string>
*/
protected $hidden = [
'password',
'remember_token',
];

/**
* Get the attributes that should be cast.
*
* @return array<string, string>
*/
```

```
protected function casts(): array
{
return [
'email_verified_at' => 'datetime',
'password' => 'hashed',
];
}
}
```

Dodawanie nauczyciela

```
$ php artisan make:seeder AdminUserSeeder
```

database/seeders/AdminUserSeeder.php

```
<?php

namespace Database\Seeders;

use App\Models\User;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class AdminUserSeeder extends Seeder
{
/**
* Run the database seeds.
*/
public function run(): void
{
// Sprawdź, czy użytkownik o roli "teacher" o nazwie "admin" już istnieje
$admin = User::where('name', 'admin')->where('role', 'teacher')->first();

// Jeśli nie istnieje, stwórz nowego użytkownika
if (!$admin) {
User::create([
'name' => 'admin',
'email' => 'admin@example.com',
'password' => bcrypt('password'),
'role' => 'teacher',
]);
}
}
```

```
}
```

```
$ php artisan db:seed --class=AdminUserSeeder
```

Uprawnienia nauczyciela

```
$ php artisan make:middleware TeacherMiddleware
```

app/Http/Middleware/TeacherMiddleware.php

```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

class TeacherMiddleware
{
/**
* Handle an incoming request.
*
* @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response) $next
*/
public function handle(Request $request, Closure $next): Response
{
if ($request->user() && $request->user()->role === 'teacher') {
return $next($request);
}

return redirect('/')->with('error', 'Brak uprawnień do tego zasobu.');
}
}
```

Routing

routes/web.php

```php
<?php
use App\Http\Controllers\ProfileController;
use Illuminate\Support\Facades\Route;
use App\Http\Middleware\TeacherMiddleware;
use App\Http\Controllers\UserController;
```

```php
Route::get('/', function () {
return view('welcome');
});

Route::get('/dashboard', function () {
return view('dashboard');
})->middleware(['auth', 'verified'])->name('dashboard');

Route::middleware('auth')->group(function () {
Route::get('/profile', [ProfileController::class, 'edit'])->name('profile.edit');
Route::patch('/profile', [ProfileController::class, 'update'])->name('profile.update');
Route::delete('/profile', [ProfileController::class, 'destroy'])->name('profile.destroy');
});

Route::middleware(['auth', TeacherMiddleware::class])->group(function () {
Route::get('/users', [UserController::class, 'index'])->name('users.index');
Route::get('/users/create', [UserController::class, 'create'])->name('users.create');
Route::post('/users', [UserController::class, 'store'])->name('users.store');
Route::get('/users/{user}/edit', [UserController::class, 'edit'])->name('users.edit');
Route::patch('/users/{user}', [UserController::class, 'update'])->name('users.update');
Route::delete('/users/{user}', [UserController::class, 'destroy'])->name('users.destroy');
});

require __DIR__.'/auth.php';
```

Kontroler

```
$ php artisan make:controller UserController
```

app/Http/Controllers/UserController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class UserController extends Controller
{
// Wyświetlenie listy użytkowników
public function index()
{
```

```
$users = User::all();
return view('users.index', compact('users'));
}

// Formularz do dodawania nowego użytkownika
public function create()
{
return view('users.create');
}

// Zapisanie nowego użytkownika
public function store(Request $request)
{
// Walidacja danych wejściowych
$validatedData = $request->validate([
'name' => 'required|string|max:255',
'email' => 'required|string|email|max:255|unique:users',
'password' => 'required|string|min:8',
'role' => 'required|in:teacher,student' // Wymagane pole 'role' z dopuszczalnymi wartościami 'teacher' lub 'student'
]);

// Stworzenie nowego użytkownika
$user = User::create([
'name' => $validatedData['name'],
'email' => $validatedData['email'],
'password' => bcrypt($validatedData['password']),
'role' => $validatedData['role'],
]);

return redirect()->route('users.index')->with('success', 'Użytkownik został dodany pomyślnie.');
}

// Formularz do edycji użytkownika
public function edit(User $user)
{
return view('users.edit', compact('user'));
}

// Aktualizacja danych użytkownika
public function update(Request $request, User $user)
{
// Walidacja danych wejściowych
$validatedData = $request->validate([
```

```
'name' => 'required|string|max:255',
'email' => 'required|string|email|max:255|unique:users,email,' . $user->id,
'password' => 'nullable|string|min:8',
'role' => 'required|in:teacher,student'
]);

// Jeśli próba zmiany SWOJEJ roli na studenta, zablokuj aktualizację i zwróć błąd
if (Auth::user()->is($user) && $validatedData['role'] === 'student') {
return redirect()->back()->withErrors(['role' => 'Nie możesz zmienić swojej roli na studenta.']);
}

// Aktualizacja danych użytkownika
$user->update([
'name' => $validatedData['name'],
'email' => $validatedData['email'],
'password' => isset($validatedData['password']) ? bcrypt($validatedData['password']) : $user->password,
'role' => $validatedData['role'],
]);

return redirect()->route('users.index')->with('success', 'Dane użytkownika zostały zaktualizowane.');
}

// Usunięcie użytkownika
public function destroy(User $user)
{
if (Auth::user()->is($user)) {
return redirect()->back()->withErrors(['role' => 'Nie możesz usunąć siebie.']);
}

$user->delete();
return redirect()->route('users.index')->with('success', 'Użytkownik został usunięty.');
}
}
```

Widoki

CRUD

```
$ php artisan make:view users.index
```

```
resources/views/users/index.blade.php
```

```
@extends('layouts.app')
```

```
@section('content')
<div class="container">
@if (session('success'))
<div class="alert alert-success">
{{ session('success') }}
</div>
@endif

<div class="row justify-content-center">
<div class="col-md-8" style="min-width:600px;">
<div class="card">
<div class="card-header">Lista użytkowników</div>

<div class="card-body">
<table class="table">
<thead>
<tr>
<th scope="col">#</th>
<th scope="col">Nazwa</th>
<th scope="col">E-mail</th>
<th scope="col">Rola</th>
<th scope="col">Akcje</th>
</tr>
</thead>
<tbody>
@foreach ($users as $user)
<tr>
<th scope="row">{{ $user->id }}</th>
<td>{{ $user->name }}</td>
<td>{{ $user->email }}</td>
<td>{{ $user->role }}</td>
<td>
<a href="{{ route('users.edit', $user->id) }}" class="btn btn-sm btn-primary">Edytuj</a>
<form action="{{ route('users.destroy', $user->id) }}" method="POST" style="display: inline;">
@csrf
@method('DELETE')
<button type="submit" class="btn btn-sm btn-danger" onclick="return confirm('Czy na pewno chcesz usunąć tego
użytkownika?')">Usuń</button>
</form>
</td>
</tr>
@endforeach
```

```
</tbody>
</table>
<div class="row justify-content-center mb-3">
<a href="{{ route('users.create') }}" class="btn btn-success">Dodaj nowego użytkownika</a>
</div>
</div>
</div>
</div>
</div>
</div>
@endsection
```

CRUD

```
$ php artisan make:view users.create
```

💻 resources/views/users/create.blade.php

```
@extends('layouts.app')

@section('content')
<div class="container">
<div class="row justify-content-center">
<div class="col-md-8">
<div class="card">
<div class="card-header">Dodaj nowego użytkownika</div>

<div class="card-body">
<form method="POST" action="{{ route('users.store') }}">
@csrf

<div class="form-group">
<label for="name">Nazwa</label>
<input type="text" name="name" id="name" class="form-control" required>
</div>

<div class="form-group">
<label for="email">E-mail</label>
<input type="email" name="email" id="email" class="form-control" required>
</div>

<div class="form-group">
<label for="password">Hasło</label>
```

```
<input type="password" name="password" id="password" class="form-control" required>
</div>

<div class="form-group">
<label for="role">Rola</label>
<select name="role" id="role" class="form-control" required>
<option value="teacher">Nauczyciel</option>
<option value="student">Uczeń</option>
</select>
</div>

<button type="submit" class="btn btn-primary">Dodaj użytkownika</button>
</form>
</div>
</div>
</div>
</div>
</div>
@endsection
```

CRUD

```
$ php artisan make:view users.edit
```

resources/views/users/edit.blade.php

```
@extends('layouts.app')

@section('content')
<div class="container">
@if ($errors->any())
<div class="alert alert-danger">
<ul>
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div>
@endif

<div class="row justify-content-center">
<div class="col-md-8">
<div class="card">
<div class="card-header">Edytuj użytkownika</div>
```

```
<div class="card-body">
<form method="POST" action="{{ route('users.update', $user->id) }}">
@csrf
@method('PATCH')

<div class="form-group">
<label for="name">Nazwa</label>
<input type="text" name="name" id="name" class="form-control" value="{{ $user->name }}" required>
</div>

<div class="form-group">
<label for="email">E-mail</label>
<input type="email" name="email" id="email" class="form-control" value="{{ $user->email }}" required>
</div>

<div class="form-group">
<label for="password">Hasło</label>
<input type="password" name="password" id="password" class="form-control">
<small class="text-muted">Pozostaw puste, jeśli nie chcesz zmieniać hasła.</small>
</div>

<div class="form-group">
<label for="role">Rola</label>
<select name="role" id="role" class="form-control" required>
<option value="teacher" @if($user->role === 'teacher') selected @endif>Nauczyciel</option>
<option value="student" @if($user->role === 'student') selected @endif>Uczeń</option>
</select>
</div>

<button type="submit" class="btn btn-primary">Zapisz zmiany</button>
</form>
</div>
</div>
</div>
</div>
</div>
@endsection
```

Dodawanie guzika do nawigacji (tylko dla nauczyciela)

resources/views/layouts/navigation.blade.php

```
...
<!-- Navigation Links -->
<div class="hidden space-x-8 sm:-my-px sm:ms-10 sm:flex">
<x-nav-link :href="route('dashboard')" :active="request()->routeIs('dashboard')">
{{ __('Dashboard') }}
</x-nav-link>
<!-- Nowy link do zarządzania użytkownikami -->
@if (Auth::user()->role === 'teacher')
<x-nav-link :href="route('users.index')" :active="request()->routeIs('users.index')">
{{ __('Użytkownicy') }}
</x-nav-link>
@endif
</div>
...
<!-- Responsive Navigation Menu -->
<div :class="{'block': open, 'hidden': ! open}" class="hidden sm:hidden">
<div class="pt-2 pb-3 space-y-1">
<x-responsive-nav-link :href="route('dashboard')" :active="request()->routeIs('dashboard')">
{{ __('Dashboard') }}
</x-responsive-nav-link>
<!-- Nowy link do zarządzania użytkownikami -->
@if (Auth::user()->role === 'teacher')
<x-responsive-nav-link :href="route('users.index')" :active="request()->routeIs('users.index')">
{{ __('Użytkownicy') }}
</x-responsive-nav-link>
@endif
</div>
...
```

Testowanie

```
$ php artisan serve
```

Zarządzanie grupami

Baza danych

```
$ php artisan make:migration create_groups_table --create=groups
```

```
💻 database/migrations/2024_04_26_062632_create_groups_table.php
```

```
<?php
```

```php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
/**
* Run the migrations.
*/
public function up(): void
{
Schema::create('groups', function (Blueprint $table) {
$table->id();
$table->string('name');
$table->timestamps();
});
}

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

```
$ php artisan migrate
```

Model

```
$ php artisan make:model Group
```

app/Models/Group.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use App\Models\User;
```

```php
class Group extends Model
{
use HasFactory;
protected $fillable = ['name'];

}
```

Routing

routes/web.php

```php
<?php

use App\Http\Controllers\ProfileController;
use Illuminate\Support\Facades\Route;
use App\Http\Middleware\TeacherMiddleware;
use App\Http\Controllers\UserController;
use App\Http\Controllers\GroupController;

Route::get('/', function () {
return view('welcome');
});

Route::get('/dashboard', function () {
return view('dashboard');
})->middleware(['auth', 'verified'])->name('dashboard');

Route::middleware('auth')->group(function () {
Route::get('/profile', [ProfileController::class, 'edit'])->name('profile.edit');
Route::patch('/profile', [ProfileController::class, 'update'])->name('profile.update');
Route::delete('/profile', [ProfileController::class, 'destroy'])->name('profile.destroy');
});

Route::middleware(['auth', TeacherMiddleware::class])->group(function () {
Route::get('/users', [UserController::class, 'index'])->name('users.index');
Route::get('/users/create', [UserController::class, 'create'])->name('users.create');
Route::post('/users', [UserController::class, 'store'])->name('users.store');
Route::get('/users/{user}/edit', [UserController::class, 'edit'])->name('users.edit');
Route::patch('/users/{user}', [UserController::class, 'update'])->name('users.update');
Route::delete('/users/{user}', [UserController::class, 'destroy'])->name('users.destroy');
Route::get('/groups', [GroupController::class, 'index'])->name('groups.index');
Route::get('/groups/create', [GroupController::class, 'create'])->name('groups.create');
```

```
Route::post('/groups', [GroupController::class, 'store'])->name('groups.store');
Route::get('/groups/{group}/edit', [GroupController::class, 'edit'])->name('groups.edit');
Route::patch('/groups/{group}', [GroupController::class, 'update'])->name('groups.update');
Route::delete('/groups/{group}', [GroupController::class, 'destroy'])->name('groups.destroy');
});

require __DIR__.'/auth.php';
```

Kontroler

```
$ php artisan make:controller GroupController
```

app/Http/Controllers/GroupController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Group;

class GroupController extends Controller
{
// Wyświetlenie listy grup
public function index()
{
$groups = Group::all();
return view('groups.index', compact('groups'));
}

// Formularz do dodawania nowej grupy
public function create()
{
return view('groups.create');
}

// Zapisanie nowej grupy
public function store(Request $request)
{
// Walidacja danych wejściowych
$validatedData = $request->validate([
'name' => 'required|string|max:255|unique:groups'
]);
```

```
// Stworzenie nowej grupy
Group::create([
'name' => $validatedData['name']
]);

return redirect()->route('groups.index')->with('success', 'Grupa została dodana pomyślnie.');
}

// Formularz do edycji grupy
public function edit(Group $group)
{
return view('groups.edit', compact('group'));
}

// Aktualizacja danych grupy
public function update(Request $request, Group $group)
{
// Walidacja danych wejściowych
$validatedData = $request->validate([
'name' => 'required|string|max:255|unique:groups,name,' . $group->id
]);

// Aktualizacja danych grupy
$group->update([
'name' => $validatedData['name']
]);

return redirect()->route('groups.index')->with('success', 'Dane grupy zostały zaktualizowane.');
}

// Usunięcie grupy
public function destroy(Group $group)
{
$group->delete();
return redirect()->route('groups.index')->with('success', 'Grupa została usunięta.');
}
}
```

Widoki

CRUD

```
$ php artisan make:view groups.index
```

resources/views/groups/index.blade.php

```
@extends('layouts.app')

@section('content')
<div class="container">
@if (session('success'))
<div class="alert alert-success">
{{ session('success') }}
</div>
@endif
<div class="row justify-content-center">
<div class="col-md-8" style="min-width:300px;">
<div class="card">
<div class="card-header">Lista grup</div>

<div class="card-body">
<table class="table">
<thead>
<tr>
<th scope="col">#</th>
<th scope="col">Nazwa</th>
<th scope="col">Akcje</th>
</tr>
</thead>
<tbody>
@foreach ($groups as $group)
<tr>
<th scope="row">{{ $group->id }}</th>
<td>{{ $group->name }}</td>
<td>
<a href="{{ route('groups.edit', $group->id) }}" class="btn btn-sm btn-primary">Edytuj</a>
<form action="{{ route('groups.destroy', $group->id) }}" method="POST" style="display: inline;">
@csrf
@method('DELETE')
<button type="submit" class="btn btn-sm btn-danger" onclick="return confirm('Czy na pewno chcesz usunąć tę
grupę?')">Usuń</button>
```

```
</form>
</td>
</tr>
@endforeach
</tbody>
</table>

<div class="row justify-content-center mb-3">
<a href="{{ route('groups.create') }}" class="btn btn-success">Dodaj nową grupę</a>
</div>
</div>
</div>
</div>
</div>
</div>
@endsection
```

CRUD

```
$ php artisan make:view groups.create
```

resources/views/groups/create.blade.php

```
@extends('layouts.app')

@section('content')
<div class="container">
<div class="row justify-content-center">
<div class="col-md-8">
<div class="card">
<div class="card-header">Dodaj nową grupę</div>

<div class="card-body">
<form method="POST" action="{{ route('groups.store') }}">
@csrf

<div class="form-group">
<label for="name">Nazwa grupy</label>
<input type="text" name="name" id="name" class="form-control" required>
</div>

<button type="submit" class="btn btn-primary">Dodaj grupę</button>
</form>
</div>
```

```
</div>
</div>
</div>
</div>
@endsection
```

CRUD

```
$ php artisan make:view groups.edit
```

resources/views/groups/edit.blade.php

```
@extends('layouts.app')

@section('content')
<div class="container">
@if ($errors->any())
<div class="alert alert-danger">
<ul>
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div>
@endif
<div class="row justify-content-center">
<div class="col-md-8">
<div class="card">
<div class="card-header">Edytuj grupę</div>

<div class="card-body">
<form method="POST" action="{{ route('groups.update', $group->id) }}">
@csrf
@method('PATCH')

<div class="form-group">
<label for="name">Nazwa grupy</label>
<input type="text" name="name" id="name" class="form-control" value="{{ $group->name }}" required>
</div>

<button type="submit" class="btn btn-primary">Zapisz zmiany</button>
</form>
</div>
```

```
</div>
</div>
</div>
</div>
@endsection
```

Dodawanie guzika do nawigacji (tylko dla nauczyciela)

resources/views/layouts/navigation.blade.php

```
...
<!-- Navigation Links -->
<div class="hidden space-x-8 sm:-my-px sm:ms-10 sm:flex">
<x-nav-link :href="route('dashboard')" :active="request()->routeIs('dashboard')">
{{ __('Dashboard') }}
</x-nav-link>
@if (Auth::user()->role === 'teacher')
<x-nav-link :href="route('users.index')" :active="request()->routeIs('users.index')">
{{ __('Użytkownicy') }}
</x-nav-link>
<!-- Nowy link do zarządzania grupami -->
<x-nav-link :href="route('groups.index')" :active="request()->routeIs('groups.index')">
{{ __('Grupy') }}
</x-nav-link>
@endif
</div>
...
<!-- Responsive Navigation Menu -->
<div :class="{'block': open, 'hidden': ! open}" class="hidden sm:hidden">
<div class="pt-2 pb-3 space-y-1">
<x-responsive-nav-link :href="route('dashboard')" :active="request()->routeIs('dashboard')">
{{ __('Dashboard') }}
</x-responsive-nav-link>
@if (Auth::user()->role === 'teacher')
<x-responsive-nav-link :href="route('users.index')" :active="request()->routeIs('users.index')">
{{ __('Użytkownicy') }}
</x-responsive-nav-link>
<!-- Nowy link do zarządzania grupami -->
<x-responsive-nav-link :href="route('groups.index')" :active="request()->routeIs('groups.index')">
{{ __('Grupy') }}
</x-responsive-nav-link>
@endif
</div>
```

```
...
```

Testowanie

```
$ php artisan serve
```

Przypisanie studentów do grup

```
$ artisan make:migration create_group_user_table
```

💻 database/migrations/2024_04_26_071608_create_group_user_table.php

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
/**
* Run the migrations.
*/
public function up(): void
{
Schema::create('group_user', function (Blueprint $table) {
$table->unsignedBigInteger('user_id');
$table->unsignedBigInteger('group_id');
$table->foreign('user_id')->references('id')->on('users')->onDelete('cascade');
$table->foreign('group_id')->references('id')->on('groups')->onDelete('cascade');
$table->primary(['user_id', 'group_id']);
});
}

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

```
$ php artisan migrate
```

Rozbudowanie modelu User

app/Models/User.php

```
...
public function groups()
{
return $this->belongsToMany(Group::class);
}
...
```

Rozbudowanie modelu Group

app/Models/Group.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use App\Models\User;

class Group extends Model
{
use HasFactory;
protected $fillable = ['name'];
public function users()
{
return $this->belongsToMany(User::class);
}
}
```

Seeding

```
$ php artisan make:seeder GroupUserSeeder
```

database/seeders/GroupUserSeeder.php

```
<?php
```

```php
namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use App\Models\User;
use App\Models\Group;

class GroupUserSeeder extends Seeder
{
public function generateLastName($firstName)
{
$vowels = array("a", "e", "i", "o", "u");
$consonants = array("b", "c", "d", "f", "g", "h", "j", "k", "l", "m", "n", "p", "r", "s", "t", "w", "z");
$lastName = "";
$firstNameLastChar = strtolower(substr($firstName, -1));

$is_vowel = (rand(0, 1) == 1);
while ((strlen($lastName) <= 5) || $is_vowel) {
if ($is_vowel) {
$lastName .= $vowels[array_rand($vowels)];
if ((strlen($lastName) >= 3) && (rand(0, 1) == 1)) {
break;
}
} else {
$lastName .= $consonants[array_rand($consonants)];
}
$is_vowel = !$is_vowel;
}

if ($firstNameLastChar == 'a') {
$lastName .= "cka";
} else {
$lastName .= "cki";
}

return ucfirst($lastName);
}

public function run(): void
{
$firstNames = array("Jakub", "Mateusz", "Kacper", "Michał", "Adam", "Szymon", "Dawid", "Piotr", "Patryk", "Bartosz",
"Marcin", "Krzysztof", "Adrian", "Łukasz", "Maciej", "Rafał", "Damian", "Daniel", "Tomasz", "Jan", "Robert", "Wojciech",
"Paweł", "Filip", "Grzegorz", "Anna", "Julia", "Zuzanna", "Maria", "Aleksandra", "Wiktoria", "Natalia", "Karolina",
```

```
"Katarzyna", "Magdalena", "Dominika", "Monika", "Paulina", "Kinga", "Aneta", "Kamila", "Patrycja", "Ewa", "Agata",
"Joanna", "Agnieszka", "Barbara", "Dorota", "Martyna", "Justyna");
$group = Group::create([
'name' => 'Grupa A',
]);
for ($i = 0; $i < 10; $i++) {
$firstName = $firstNames[array_rand($firstNames)];
$lastName = $this->generateLastName($firstName);
$student = User::create([
'name' => $firstName . " " . $lastName,
'email' => substr(strtolower($firstName), 0, 3) . '.' . substr(strtolower($lastName), 0, 3) . '@example.com',
'password' => bcrypt('password'),
'role' => 'student',
]);
$group->users()->attach($student);
}

}
}
```

```
$ php artisan db:seed –class=GroupUserSeeder
```

Rozbudowanie kontrolera GroupController

app/Http/Controllers/GroupController.php

```
...
use App\Models\User;
...
public function assignForm()
{
$users = User::all();
$groups = Group::all();
return view('groups.assign', compact('users', 'groups'));
}

public function assign(Request $request)
{
$userGroups = $request->input('user_group');
foreach (User::all() as $user) {
if(!isset ($userGroups[$user->id])) {
$user->groups()->sync([]);
} else {
$user->groups()->sync($userGroups[$user->id]);
```

```
}
}

return redirect()->route('groups.assign.form')->with('success', 'Przypisanie użytkowników do grup zakończone.');
}
...
```

Routing

routes/web.php

```
...
Route::middleware(['auth', TeacherMiddleware::class])->group(function () {
Route::get('/groups/assign', [GroupController::class, 'assignForm'])->name('groups.assign.form');
Route::post('/groups/assign', [GroupController::class, 'assign'])->name('groups.assign');
...
```

Widok

```
$ php artisan make:view groups.assign
```

resources/views/groups/assign.blade.php

```
@extends('layouts.app')

@section('content')
<div class="container">
@if (session('success'))
<div class="alert alert-success">
{{ session('success') }}
</div>
@endif

<div class="row justify-content-center">
<div class="col-md-8" style="min-width:300px;">
<div class="card">
<div class="card-header">Grupowanie studentów</div>

<div class="card-body">
<form action="{{ route('groups.assign') }}" method="POST">
@csrf

<table class="table">
<thead>
```

```
<tr>
<th>Uczeń</th>
@foreach ($groups as $group)
<th style="writing-mode: vertical-rl;">{{ $group->name }}</th>
@endforeach
</tr>
</thead>
<tbody>
@foreach ($users as $user)
<tr>
<td>{{ $user->name }}</td>
@foreach ($groups as $group)
<td>
<input type="checkbox" name="user_group[{{ $user->id }}][]"
value="{{ $group->id }}"
{{ $user->groups->contains($group) ? 'checked' : '' }}>
</td>
@endforeach
</tr>
@endforeach
</tbody>
</table>
<button type="submit" class="btn btn-primary">Zapisz przypisanie do grup</button>
</form>
</div>
</div>
</div>
</div>
</div>
@endsection
```

Link w navbar

resources/views/layouts/navigation.blade.php

```
...
<!-- Navigation Links -->
...
<!-- Nowy link do grupowanie studentów -->
<x-nav-link :href="route('groups.assign.form')" :active="request()->routeIs('groups.assign.form')">
{{ __('Grupowanie Studentów') }}
</x-nav-link>
...
```

```
<!-- Responsive Navigation Menu -->
...
<!-- Nowy link do grupowania studentów -->
<x-responsive-nav-link :href="route('groups.assign.form')" :active="request()->routeIs('groups.assign.form')">
{{ __('Grupowanie Studentów') }}
</x-responsive-nav-link>
...
```

Zarządzanie pytaniami i odpowiedziami

Migracje

```
$ php artisan make:migration create_questions_table --create=questions
```

💻 database/migrations/2024_04_26_191345_create_questions_table.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
/**
* Run the migrations.
*/
public function up(): void
{Schema::create('questions', function (Blueprint $table) {
$table->id();
$table->text('content');
$table->boolean('multiselect')->default(false);
$table->timestamps();
});

}

/**
* Reverse the migrations.
*/
public function down(): void
{
```

```
Schema::dropIfExists('questions');
}
};
```

```
$ php artisan make:migration create_answers_table --create=answers
```

database/migrations/2024_04_26_191917_create_answers_table.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
/**
* Run the migrations.
*/
public function up(): void
{
Schema::create('answers', function (Blueprint $table) {
$table->id();
$table->text('content');
$table->boolean('correct')->default(false);
$table->foreignId('question_id')->constrained()->onDelete('cascade');
$table->timestamps();
});
}

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

```
$ php artisan migrate
```

Modele

```
$ php artisan make:model Question
```

app/Models/Question.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Question extends Model
{
use HasFactory;
protected $fillable = ['content', 'multiselect'];

public function answers()
{
return $this->hasMany(Answer::class);
}

public function isValid(){
if(!$this->multiselect){
return $this->answers()->where('correct', true)->exists();
} else {
return true;
}
}
}
```

```
$ php artisan make:model Answer
```

app/Models/Answer.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Answer extends Model
```

```
{
use HasFactory;
protected $fillable = ['content', 'correct'];

public function question()
{
return $this->belongsTo(Question::class);
}
}
```

Seeder

```
$ php artisan make:seeder QuestionSeeder
```

```
database/seeders/QuestionSeeder.php
```

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use App\Models\Question;
use App\Models\Answer;

class QuestionSeeder extends Seeder
{
/**
* Run the database seeds.
*/
public function run(): void
{
$question1 = Question::create([
'content' => 'Jakie jest stolice Polski?',
'multiselect' => false,
]);

$question1->answers()->createMany([
['content' => 'Warszawa', 'correct' => true],
['content' => 'Kraków', 'correct' => false],
['content' => 'Gdańsk', 'correct' => false],
['content' => 'Poznań', 'correct' => false],
```

```
]);

$question2 = Question::create([
'content' => 'Które z tych zwierząt jest ssakiem?',
'multiselect' => true,
]);

$question2->answers()->createMany([
['content' => 'Pies', 'correct' => true],
['content' => 'Kot', 'correct' => true],
['content' => 'Ptak', 'correct' => false],
['content' => 'Rekin', 'correct' => false],
]);

$question3 = Question::create([
'content' => 'Kto był pierwszym prezydentem USA?',
'multiselect' => false,
]);

$question3->answers()->createMany([
['content' => 'George Washington', 'correct' => true],
['content' => 'Thomas Jefferson', 'correct' => false],
['content' => 'John Adams', 'correct' => false],
['content' => 'Abraham Lincoln', 'correct' => false],
]);
}
}
```

```
$ php artisan db:seed --class=QuestionSeeder
```

Routing

routes/web.php

```
...
use App\Http\Controllers\QuestionController;
use App\Http\Controllers\AnswerController;
...
Route::middleware(['auth', TeacherMiddleware::class])->group(function () {
Route::post('/questions/{question}/answers', [AnswerController::class, 'store'])->name('answers.store');
Route::get('/answers/{answer}/edit', [AnswerController::class, 'edit'])->name('answers.edit');
Route::patch('/answers/{answer}', [AnswerController::class, 'update'])->name('answers.update');
Route::delete('/answers/{answer}', [AnswerController::class, 'destroy'])->name('answers.destroy');
```

```
Route::get('/questions', [QuestionController::class, 'index'])->name('questions.index');
Route::get('/questions/create', [QuestionController::class, 'create'])->name('questions.create');
Route::post('/questions', [QuestionController::class, 'store'])->name('questions.store');
Route::get('/questions/{question}/edit', [QuestionController::class, 'edit'])->name('questions.edit');
Route::patch('/questions/{question}', [QuestionController::class, 'update'])->name('questions.update');
Route::delete('/questions/{question}', [QuestionController::class, 'destroy'])->name('questions.destroy');
Route::post('/questions/{question}/duplicate', [QuestionController::class, 'duplicate'])->name('questions.duplicate');
...
```

Kontrolery

```
$ php artisan make:controller QuestionController
```

app/Http/Controllers/QuestionController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Question;

class QuestionController extends Controller
{
public function index()
{
$questions = Question::all();
return view('questions.index', compact('questions'));
}

public function create()
{
return view('questions.create');
}

public function store(Request $request)
{
$validatedData = $request->validate([
'content' => 'required|string',
]);
$validatedData['multiselect'] = $request->has('multiselect');

$question = Question::create($validatedData);
```

```
return redirect()->route('questions.edit', $question)->with('success', 'Pytanie zostało dodane.');
}

public function edit(Question $question)
{
return view('questions.edit', compact('question'));
}

public function update(Request $request, Question $question)
{
$validatedData = $request->validate([
'content' => 'required|string',
]);
$validatedData['multiselect'] = $request->has('multiselect');

$question->update($validatedData);

return redirect()->route('questions.edit', $question)->with('success', 'Pytanie zostało zaktualizowane.');
}

public function destroy(Question $question)
{
$question->delete();
return redirect()->route('questions.index')->with('success', 'Pytanie zostało usunięte.');
}
public function duplicate(Question $question)
{
$new_question = $question->replicate();
$new_question->save();

foreach ($question->answers as $answer) {
$new_answer = $answer->replicate();
$new_answer->question_id = $new_question->id;
$new_answer->save();
}

return redirect()->route('questions.index')->with('success', 'Pytanie zostało duplikowane.');
}
}
```

```
$ php artisan make:controller AnswerController
```

app/Http/Controllers/AnswerController.php

```php
<?php

namespace App\Http\Controllers;
use App\Models\Answer;
use App\Models\Question;

use Illuminate\Http\Request;

class AnswerController extends Controller
{
public function store(Request $request, Question $question)
{
$validatedData = $request->validate([
'content' => 'required|string',
]);
$validatedData['correct'] = $request->has('correct');

$question->answers()->create($validatedData);

return redirect()->route('questions.edit', $question)->with('success', 'Odpowiedź została dodana.');
}

public function edit(Answer $answer)
{
return view('answers.edit', compact('answer'));
}

public function update(Request $request, Answer $answer)
{
$validatedData = $request->validate([
'content' => 'required|string',
]);
$validatedData['correct'] = $request->has('correct');

$answer->update($validatedData);

return redirect()->route('questions.edit', $answer->question)->with('success', 'Odpowiedź została zaktualizowana.');
}

public function destroy(Answer $answer)
{
```

```
$answer->delete();
return redirect()->route('questions.edit', $answer->question)->with('success', 'Odpowiedź została usunięta.');
}
}
```

Widoki

CRUD

```
$ php artisan make:view questions.index
```

resources/views/questions/index.blade.php

```
@extends('layouts.app')

@section('content')
<div class="container">
@if (session('success'))
<div class="alert alert-success">
{{ session('success') }}
</div>
@elseif ($errors->any())
<div class="alert alert-danger">
<ul>
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div>
@endif

<div class="row justify-content-center">
<div class="col-md-8" style="min-width:600px;">
<div class="card">
<div class="card-header">Lista pytań</div>

<div class="card-body">
<table class="table">
<thead>
<tr>
<th scope="col">#</th>
<th scope="col">Treść pytania</th>
<th scope="col">Odpowiedzi</th>
```

```
<th scope="col">Wielokrotny wybór</th>
<th scope="col">Akcje</th>
</tr>
</thead>
<tbody>
@foreach ($questions as $question)
<tr>
<th scope="row">{{ $question->id }}</th>
<td>{{ $question->content }}</td>
<td>
<ul>
@foreach( $question->answers as $answer)
<li style="color:{{ $answer->correct?"green":"red" }}">
[{{$answer->content}}]
</li>
@endforeach
</ul>
</td>
<td>{{ $question->multiselect ? 'Tak' : 'Nie' }}</td>
<td>
<a href="{{ route('questions.edit', $question->id) }}" class="btn btn-sm btn-primary">Edytuj</a>
<form action="{{ route('questions.duplicate', $question->id) }}" method="POST" style="display: inline;">
@csrf
@method('POST')
<button type="submit" class="btn btn-sm btn-primary">Zduplikuj</button>
</form>
<form action="{{ route('questions.destroy', $question->id) }}" method="POST" style="display: inline;">
@csrf
@method('DELETE')
<button type="submit" class="btn btn-sm btn-danger" onclick="return confirm('Czy na pewno chcesz usunąć to
pytanie?')">Usuń</button>
</form>
</td>
</tr>
@endforeach
</tbody>
</table>

<div class="row justify-content-center mb-3">
<a href="{{ route('questions.create') }}" class="btn btn-success">Dodaj nowe pytanie</a>
</div>
</div>
</div>
```

```
</div>
</div>
</div>
@endsection
```

CRUD

```
$ php artisan make:view questions.create
```

resources/views/questions/create.blade.php

```
@extends('layouts.app')

@section('content')
<div class="container">
@if ($errors->any())
<div class="alert alert-danger">
<ul>
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div>
@endif
<div class="row justify-content-center">
<div class="col-md-8">
<div class="card">
<div class="card-header">Dodaj nowe pytanie</div>

<div class="card-body">
<form method="POST" action="{{ route('questions.store') }}">
@csrf

<div class="form-group">
<label for="content">Treść pytania</label>
<textarea name="content" id="content" class="form-control" rows="4" required>{{ old('content') }}</textarea>
</div>

<div class="form-group">
<div class="form-check">
<input class="form-check-input" type="checkbox" name="multiselect" id="multiselect">
<label class="form-check-label" for="multiselect">
Wielokrotny wybór
```

```
</label>
</div>
</div>

<button type="submit" class="btn btn-primary">Dodaj pytanie</button>
</form>
</div>
</div>
</div>
</div>
</div>
@endsection
```

CRUD

```
$ php artisan make:view questions.edit
```

resources/views/questions/edit.blade.php

```
@extends('layouts.app')

@section('content')
<div class="container">
@if (session('success'))
<div class="alert alert-success">
{{ session('success') }}
</div>
@elseif ($errors->any())
<div class="alert alert-danger">
<ul>
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div>
@endif
<div class="row justify-content-center">
<div class="col-md-8">
<div class="card">
<div class="card-header">Edytuj pytanie</div>

<div class="card-body">
<form method="POST" action="{{ route('questions.update', $question->id) }}">
```

```
@csrf
@method('PATCH')

<div class="form-group">
<label for="content">Treść pytania</label>
<textarea name="content" id="content" class="form-control" rows="1" required>{{ old('content', $question->content)
}}</textarea>
</div>

<div class="form-group">
<div class="form-check">
<input class="form-check-input" type="checkbox" name="multiselect" id="multiselect" @if($question->multiselect) checked
@endif>
<label class="form-check-label" for="multiselect">
Wielokrotny wybór
</label>
</div>
</div>
<button type="submit" class="btn btn-primary">Zapisz zmiany</button>
</form>
</div>
</div>
<div class="card">
<div class="card-header">Edytuj odpowiedzi</div>
<div class="card-body">
<table class="table">
<thead>
<tr>
<th scope="col">#</th>
<th scope="col">Treść odpowiedzi</th>
<th scope="col">Poprawna</th>
<th scope="col">Akcje</th>
<th scope="col"></th>
</tr>
</thead>
<tbody>
@foreach ($question->answers as $answer)
<tr>
<form method="POST" action="{{ route('answers.update', $answer->id) }}">
@csrf
@method('PATCH')
<td>{{ $answer->id }}</td>
<td>
```

```
<input type="text" name="content" id="content_{{ $answer->id }}" class="form-control" required value="{{ old('content', $answer->content) }}" />
</td>
<td>
<input class="form-check-input" type="checkbox" name="correct" id="correct_{{ $answer->id }}" @if($answer->correct) checked @endif>
</td>
<td>
<button type="submit" class="btn btn-sm btn-primary">Zapisz</button>
</td>
</form>
<td>
<form action="{{ route('answers.destroy', $answer->id) }}" method="POST" style="display: inline;">
@csrf
@method('DELETE')
<button type="submit" class="btn btn-sm btn-danger" onclick="return confirm('Czy na pewno chcesz usunąć tę odpowiedź?')">Usuń</button>
</form>
</td>
</tr>
@endforeach
</tbody>
<tfoot>
<form method="POST" action="{{ route('answers.store', $question) }}">
@csrf
@method('POST')
<tr>
<td>+</td>
<td>
<input type="text" name="content" id="content_new" class="form-control" required />
</td>
<td>
<input class="form-check-input" type="checkbox" name="correct" id="correct_new">
</td>
<td colspan=2>
<button type="submit" class="btn btn-sm btn-primary">Dodaj</button>
</td>
</tr>
</form>
</tfoot>
</table>
</div>
</div>
```

```
</div>
</div>
</div>
@endsection
```

Zarządzanie testami

```
$ php artisan make:migration create_tests_table --create=tests
```

database/migrations/2024_04_27_090924_create_tests_table.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
public function up(): void
{
Schema::create('tests', function (Blueprint $table) {
$table->id();
$table->string('name');
$table->timestamps();
});
}

public function down(): void
{
Schema::dropIfExists('tests');
}
};
```

```
$ php artisan make:migration create_question_test_table --create=question_test
```

database/migrations/2024_04_27_102526_create_question_test_table.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
```

```
return new class extends Migration
{
public function up(): void
{
Schema::create('question_test', function (Blueprint $table) {
$table->unsignedBigInteger('question_id');
$table->unsignedBigInteger('test_id');
$table->foreign('question_id')->references('id')->on('questions')->onDelete('cascade');
$table->foreign('test_id')->references('id')->on('tests')->onDelete('cascade');
$table->primary(['question_id', 'test_id']);
});
}
public function down(): void
{
Schema::dropIfExists('question_test');
}
};
```

```
$ php artisan migrate
```

Model

```
$ php artisan make:model Test
```

app/Models/Test.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use App\Models\Question;

class Test extends Model
{
use HasFactory;
protected $fillable = ['name', 'start_date', 'end_date'];

public function questions()
{
return $this->belongsToMany(Question::class);
```

```
}
}
```

app/Models/Question.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use App\Models\Test;

class Question extends Model
{
use HasFactory;
protected $fillable = ['content', 'multiselect'];

public function answers()
{
return $this->hasMany(Answer::class);
}
public function isValid(){
if(!$this->multiselect){
return $this->answers()->where('correct', true)->exists();
} else {
return true;
}
}
public function tests()
{
return $this->belongsToMany(Test::class);
}
}
```

Routing

routes/web.php

```
...
Route::middleware(['auth', TeacherMiddleware::class])->group(function () {
Route::get('/tests', [TestController::class, 'index'])->name('tests.index');
Route::get('/tests/create', [TestController::class, 'create'])->name('tests.create');
```

```
Route::post('/tests', [TestController::class, 'store'])->name('tests.store');
Route::get('/tests/{test}/edit', [TestController::class, 'edit'])->name('tests.edit');
Route::patch('/tests/{test}', [TestController::class, 'update'])->name('tests.update');
Route::delete('/tests/{test}', [TestController::class, 'destroy'])->name('tests.destroy');
Route::get('/tests/assign', [TestController::class, 'assignForm'])->name('tests.assign.form');
Route::post('/tests/assign', [TestController::class, 'assign'])->name('tests.assign');
Route::post('/tests/{test}/duplicate', [TestController::class, 'duplicate'])->name('tests.duplicate');
...
```

Kontroler

```
$ php artisan make:controller TestController
```

app/Http/Controllers/TestController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Test;
use App\Models\Question;

class TestController extends Controller
{
public function index()
{
$tests = Test::all();
return view('tests.index', compact('tests'));
}

public function create()
{
return view('tests.create');
}

public function store(Request $request)
{
$validatedData = $request->validate([
'name' => 'required|string|max:255',
]);

Test::create($validatedData);
```

```
return redirect()->route('tests.index')->with('success', 'Test została dodana pomyślnie.');
}

public function edit(Test $test)
{
return view('tests.edit', compact('test'));
}

public function update(Request $request, Test $test)
{
$validatedData = $request->validate([
'name' => 'required|string|max:255',
]);

$test->update($validatedData);

return redirect()->route('tests.index')->with('success', 'Dane testu zostały zaktualizowane.');
}

public function destroy(Test $test)
{
$test->delete();
return redirect()->route('tests.index')->with('success', 'Test został usunięty.');
}

public function assignForm()
{
$questions = Question::all();
$tests = Test::all();
return view('tests.assign', compact('questions', 'tests'));
}
public function assign(Request $request)
{
$questionTests = $request->input('question_test');
foreach (Question::all() as $question) {
if(!isset ($questionTests[$question->id])) {
$question->tests()->sync([]);
} else {
$question->tests()->sync($questionTests[$question->id]);
}
}
```

```
return redirect()->route('tests.assign.form')->with('success', 'Przypisanie pytań do testów zakończone.');
}

public function duplicate(Test $test)
{
$new_test = $test->replicate();
$new_test->save();

foreach ($test->questions as $question) {
$new_test->questions()->attach($question);
}
return redirect()->route('tests.index')->with('success', 'Test został zduplikowane.');
}
}
```

Nawigacja

resources/views/layouts/navigation.blade.php

```
...
<!-- Navigation Links -->
...
<x-nav-link :href="route('tests.index')" :active="request()->routeIs('tests.index')">
{{ __('Testy') }}
</x-nav-link>
<x-nav-link :href="route('tests.assign.form')" :active="request()->routeIs('tests.assign.form')">
{{ __('Grupowanie Pytań') }}
</x-nav-link>

...
<!-- Responsive Navigation Menu -->
...
<x-responsive-nav-link :href="route('tests.index')" :active="request()->routeIs('tests.index')">
{{ __('Testy') }}
</x-responsive-nav-link>
<x-responsive-nav-link :href="route('tests.assign.form')" :active="request()->routeIs('tests.assign.form')">
{{ __('Grupowanie Pytań') }}
</x-responsive-nav-link>
...
```

Widoki

CRUD

```
$ php artisan make:view tests.index
```

resources/views/tests/index.blade.php

```
@extends('layouts.app')

@section('content')
<div class="container">
@if (session('success'))
<div class="alert alert-success">
{{ session('success') }}
</div>
@endif
<div class="row justify-content-center">
<div class="col-md-8" style="min-width:300px;">
<div class="card">
<div class="card-header">Lista testów</div>

<div class="card-body">
<table class="table">
<thead>
<tr>
<th scope="col">#</th>
<th scope="col">Nazwa</th>
<th scope="col">Pytania</th>
<th scope="col">Akcje</th>
</tr>
</thead>
<tbody>
@foreach ($tests as $test)
<tr>
<th scope="row">{{ $test->id }}</th>
<td>{{ $test->name }}</td>
<td>
<ul>
@foreach( $test->questions as $question)
<li style="color:{{ $question->isValid()?"black":"red" }}">
[<a href="{{ route('questions.edit', $question->id) }}">{{$question->content}}</a>]
</li>
@endforeach
```

```
</ul>
</td>
<td>
<a href="{{ route('tests.edit', $test->id) }}" class="btn btn-sm btn-primary">Edytuj</a>
<form action="{{ route('tests.duplicate', $test->id) }}" method="POST" style="display: inline;">
@csrf
@method('POST')
<button type="submit" class="btn btn-sm btn-primary")">Zduplikuj</button>
</form>
<form action="{{ route('tests.destroy', $test->id) }}" method="POST" style="display: inline;">
@csrf
@method('DELETE')
<button type="submit" class="btn btn-sm btn-danger" onclick="return confirm('Czy na pewno chcesz usunąć tę
grupę?')">Usuń</button>
</form>
</td>
</tr>
@endforeach
</tbody>
</table>

<div class="row justify-content-center mb-3">
<a href="{{ route('tests.create') }}" class="btn btn-success">Dodaj nowy test</a>
</div>
</div>
</div>
</div>
</div>
</div>
@endsection
```

CRUD

```
$ php artisan make:view tests.create
```

💻 resources/views/tests/create.blade.php

```
@extends('layouts.app')

@section('content')
<div class="container">
@if ($errors->any())
<div class="alert alert-danger">
<ul>
```

```
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div>
@endif
<div class="row justify-content-center">
<div class="col-md-8">
<div class="card">
<div class="card-header">Dodaj nowy test</div>

<div class="card-body">
<form method="POST" action="{{ route('tests.store') }}">
@csrf

<div class="form-group">
<label for="name">Nazwa testu</label>
<input type="text" name="name" id="name" class="form-control" required>
</div>

<button type="submit" class="btn btn-primary">Dodaj test</button>
</form>
</div>
</div>
</div>
</div>
</div>
@endsection
```

CRUD

```
$ php artisan make:view tests.edit
```

resources/views/tests/edit.blade.php

```
@extends('layouts.app')

@section('content')
<div class="container">
@if ($errors->any())
<div class="alert alert-danger">
<ul>
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
```

```
@endforeach
</ul>
</div>
@endif
<div class="row justify-content-center">
<div class="col-md-8">
<div class="card">
<div class="card-header">Edytuj test</div>

<div class="card-body">
<form method="POST" action="{{ route('tests.update', $test) }}">
@csrf
@method('PATCH')

<div class="form-group">
<label for="name">Nazwa testu</label>
<input type="text" name="name" id="name" class="form-control" value="{{ $test->name }}" required>
</div>

<button type="submit" class="btn btn-primary">Zapisz zmiany</button>
</form>
</div>
</div>
</div>
</div>
</div>
@endsection
```

Grupowanie pytań do testów

```
$ php artisan make:view tests.assign
```

```
resources/views/tests/assign.blade.php
```

```
@extends('layouts.app')

@section('content')
<div class="container">
@if (session('success'))
<div class="alert alert-success">
{{ session('success') }}
</div>
@endif
```

```
<div class="row justify-content-center">
<div class="col-md-8" style="min-width:300px;">
<div class="card">
<div class="card-header">Grupowanie pytań</div>

<div class="card-body">
<form action="{{ route('tests.assign') }}" method="POST">
@csrf

<table class="table">
<thead>
<tr>
<th>Pytanie</th>
@foreach ($tests as $test)
<th style="writing-mode: vertical-rl;">{{ $test->name }}</th>
@endforeach
</tr>
</thead>
<tbody>
@foreach ($questions as $question)
<tr style="background:{{$question->isValid()?"none":"pink"}}">
<td>
<a href="{{ route('questions.edit', $question->id) }}">
{{ substr($question->content, 0, 40) }}{{ strlen($question->content)>40?"...":"" }}
</a>
</td>
@foreach ($tests as $test)
<td>
<input type="checkbox" name="question_test[{{ $question->id }}][]"
value="{{ $test->id }}"
{{ $question->tests->contains($test) ? 'checked' : '' }}>
</td>
@endforeach
</tr>
@endforeach
</tbody>
</table>
<button type="submit" class="btn btn-primary">Zapisz przypisanie do testu</button>
</form>
</div>
</div>
</div>
</div>
```

```
</div>
@endsection
```

Seeder

```
$ php artisan make:seeder TestSeeder
```

database/seeders/TestSeeder.php

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use App\Models\Question;
use App\Models\Test;
use App\Models\Answer;

class TestSeeder extends Seeder
{
/**
* Run the database seeds.
*/
public function run(): void
{
$test = Test::create([
"name"=> "Algorytmy i struktury danych 1",
]);

$questions = array();

$questions[] = Question::create([
'content' => 'Co to jest algorytm?',
'multiselect' => false,
]);
end($questions)->answers()->createMany([
['content' => 'Matematyczna procedura rozwiązująca problem', 'correct' => true],
['content' => 'Dane przechowywane w pamięci komputera', 'correct' => false],
['content' => 'Funkcja w języku programowania', 'correct' => false],
['content' => 'Sposób komunikacji między programami', 'correct' => false],
]);
```

```php
$questions[] = Question::create([
'content' => 'Jaką złożoność czasową ma algorytm o złożoności O(log n)?',
'multiselect' => false,
]);
end($questions)->answers()->createMany([
['content' => 'Stałą', 'correct' => false],
['content' => 'Liniową', 'correct' => false],
['content' => 'Logarytmiczną', 'correct' => true],
['content' => 'Wykładniczą', 'correct' => false],
]);

$questions[] = Question::create([
'content' => 'Która struktura danych najlepiej nadaje się do szybkiego wyszukiwania elementów w posortowanym zbiorze?',
'multiselect' => false,
]);

end($questions)->answers()->createMany([
['content' => 'Tablica', 'correct' => false],
['content' => 'Lista jednokierunkowa', 'correct' => false],
['content' => 'Kopiec binarny', 'correct' => true],
['content' => 'Drzewo binarne', 'correct' => false],
]);

$questions[] = Question::create([
'content' => 'Jakie jest główne zastosowanie stosu w strukturach danych?',
'multiselect' => false,
]);
end($questions)->answers()->createMany([
['content' => 'Przechowywanie danych w losowej kolejności', 'correct' => false],
['content' => 'Wykonywanie operacji w kolejności LIFO (Last In, First Out)', 'correct' => true],
['content' => 'Przechowywanie danych w posortowanej kolejności', 'correct' => false],
['content' => 'Wykonywanie operacji w kolejności FIFO (First In, First Out)', 'correct' => false],
]);

$questions[] = Question::create([
'content' => 'Który z algorytmów służy do sortowania przez scalanie?',
'multiselect' => false,
]);
end($questions)->answers()->createMany([
['content' => 'Quick Sort', 'correct' => false],
['content' => 'Bubble Sort', 'correct' => false],
['content' => 'Merge Sort', 'correct' => true],
['content' => 'Insertion Sort', 'correct' => false],
```

```
]);

$questions[] = Question::create([
'content' => 'Które z poniższych są przykładami struktur danych?',
'multiselect' => true,
]);
end($questions)->answers()->createMany([
['content' => 'Tablica', 'correct' => true],
['content' => 'Drzewo', 'correct' => true],
['content' => 'Stos', 'correct' => true],
['content' => 'RAM', 'correct' => false],
['content' => 'CPU', 'correct' => false],
]);

$questions[] = Question::create([
'content' => 'Które z poniższych algorytmów są wykorzystywane do przeszukiwania grafów?',
'multiselect' => true,
]);
end($questions)->answers()->createMany([
['content' => 'BFS', 'correct' => true],
['content' => 'DFS', 'correct' => true],
['content' => 'DFT', 'correct' => false],
['content' => 'Algorytm Dijkstry', 'correct' => true],
['content' => 'Algorytm Gaussa-Jordana', 'correct' => false],
]);

foreach($questions as $question){
$test->questions()->attach($question);
}
}
}
```

```
$ php artisan db:seed --class=TestSeeder
```

Egzaminy

Migracja

```
$ php artisan make:migration create_exams_table
```

```
database/migrations/2024_04_27_121518_create_exams_table.php
```

```
<?php
```

```php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
public function up(): void
{
Schema::create('exams', function (Blueprint $table) {
$table->id();
$table->unsignedBigInteger('test_id');
$table->unsignedBigInteger('group_id');
$table->dateTime('start_date');
$table->dateTime('end_date');
$table->timestamps();
$table->foreign('test_id')->references('id')->on('tests')->onDelete('cascade');
$table->foreign('group_id')->references('id')->on('groups')->onDelete('cascade');
});
}
public function down(): void
{
Schema::dropIfExists('exams');
}
};
```

```
$ php artisan migrate
```

Model

```
$ php artisan make:model Exam
```

```
app/Models/Exam.php
```

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Exam extends Model
{
```

```
use HasFactory;
protected $fillable = [
'test_id',
'group_id',
'start_date',
'end_date',
];

public function test()
{
return $this->belongsTo(Test::class);
}

public function group()
{
return $this->belongsTo(Group::class);
}
}
```

app/Models/Test.php

```
...
public function exams()
{
return $this->hasMany(Exam::class);
}
...
```

app/Models/Group.php

```
...
public function exams()
{
return $this->hasMany(Exam::class);
}
...
```

Routing

routes/web.php

```
...
use App\Http\Controllers\ExamController;
...
```

```
Route::middleware(['auth', TeacherMiddleware::class])->group(function () {
Route::get('/exams', [ExamController::class, 'index'])->name('exams.index');
Route::get('/exams/create', [ExamController::class, 'create'])->name('exams.create');
Route::post('/exams', [ExamController::class, 'store'])->name('exams.store');
Route::get('/exams/{exam}/edit', [ExamController::class, 'edit'])->name('exams.edit');
Route::patch('/exams/{exam}', [ExamController::class, 'update'])->name('exams.update');
Route::delete('/exams/{exam}', [ExamController::class, 'destroy'])->name('exams.destroy');
...
```

Kontroler

```
$ php artisan make:controller ExamController
```

💻 app/Http/Controllers/ExamController.php

```
...
use App\Http\Controllers\ExamController;
...
Route::middleware(['auth', TeacherMiddleware::class])->group(function () {
Route::get('/exams', [ExamController::class, 'index'])->name('exams.index');
Route::get('/exams/create', [ExamController::class, 'create'])->name('exams.create');
Route::post('/exams', [ExamController::class, 'store'])->name('exams.store');
Route::get('/exams/{exam}/edit', [ExamController::class, 'edit'])->name('exams.edit');
Route::patch('/exams/{exam}', [ExamController::class, 'update'])->name('exams.update');
Route::delete('/exams/{exam}', [ExamController::class, 'destroy'])->name('exams.destroy');
...
```

Nawigacja

💻 resources/views/layouts/navigation.blade.php

```
...
<!-- Navigation Links -->
...
<x-nav-link :href="route('exams.index')" :active="request()->routeIs('exams.index')">
{{ __('Egzaminy') }}
</x-nav-link>

...
<!-- Responsive Navigation Menu -->
...
<x-responsive-nav-link :href="route('exams.index')" :active="request()->routeIs('exams.index')">
{{ __('Egzaminy') }}
```

```
</x-responsive-nav-link>
...
```

Widoki

CRUD

```
$ php artisan make:view exams.index
```

resources/views/exams/index.blade.php

```
@extends('layouts.app')

@section('content')
<div class="container">
@if (session('success'))
<div class="alert alert-success">
{{ session('success') }}
</div>
@endif
<div class="row justify-content-center">
<div class="col-md-8" style="min-width:300px;">
<div class="card">
<div class="card-header">Lista egzaminów</div>

<div class="card-body">
<table class="table">
<thead>
<tr>
<th scope="col">#</th>
<th scope="col">Test</th>
<th scope="col">Grupa</th>
<th scope="col">Start</th>
<th scope="col">Koniec</th>
<th scope="col">Akcje</th>
</tr>
</thead>
<tbody>
@foreach ($exams as $exam)
<tr>
<th scope="row">{{ $exam->id }}</th>
<td>{{ $exam->test->name }}</td>
<td>{{ $exam->group->name }}</td>
```

```
<td>{{ $exam->start_date }}</td>
<td>{{ $exam->end_date }}</td>
<td>
<a href="{{ route('exams.edit', $exam->id) }}" class="btn btn-sm btn-primary">Edytuj</a>
<form action="{{ route('exams.destroy', $exam->id) }}" method="POST" style="display: inline;">
@csrf
@method('DELETE')
<button type="submit" class="btn btn-sm btn-danger" onclick="return confirm('Czy na pewno chcesz usunąć ten
egzamin?')">Usuń</button>
</form>
</td>
</tr>
@endforeach
</tbody>
</table>

<div class="row justify-content-center mb-3">
<a href="{{ route('exams.create') }}" class="btn btn-success">Dodaj nowy egzamin</a>
</div>
</div>
</div>
</div>
</div>
</div>
@endsection
```

CRUD

```
$ php artisan make:view exams.create
```

💻 resources/views/exams/create.blade.php

```
@extends('layouts.app')

@section('content')
<div class="container">
<div class="row justify-content-center">
<div class="col-md-8">
<div class="card">
<div class="card-header">Dodaj nowy egzamin</div>

<div class="card-body">
@if ($errors->any())
<div class="alert alert-danger">
```

```
<ul>
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div>
@endif

<form method="POST" action="{{ route('exams.store') }}">
@csrf

<div class="form-group">
<label for="test_id">Test</label>
<select name="test_id" id="test_id" class="form-control">
@foreach($tests as $test)
<option value="{{ $test->id }}">{{ $test->name }}</option>
@endforeach
</select>
</div>

<div class="form-group">
<label for="group_id">Grupa</label>
<select name="group_id" id="group_id" class="form-control">
@foreach($groups as $group)
<option value="{{ $group->id }}">{{ $group->name }}</option>
@endforeach
</select>
</div>

<div class="form-group">
<label for="start_date">Data rozpoczęcia</label>
<input type="date" name="start_date" id="start_date" class="form-control" required>
</div>

<div class="form-group">
<label for="end_date">Data zakończenia</label>
<input type="date" name="end_date" id="end_date" class="form-control" required>
</div>

<button type="submit" class="btn btn-primary">Dodaj egzamin</button>
</form>
</div>
</div>
```

```
</div>
</div>
</div>
@endsection
```

CRUD

```
$ php artisan make:view exams.edit
```

resources/views/exams/edit.blade.php

```
@extends('layouts.app')

@section('content')
<div class="container">
<div class="row justify-content-center">
<div class="col-md-8">
<div class="card">
<div class="card-header">Edytuj egzamin</div>

<div class="card-body">
@if ($errors->any())
<div class="alert alert-danger">
<ul>
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div>
@endif

<form method="POST" action="{{ route('exams.update', $exam->id) }}">
@csrf
@method('PATCH')

<div class="form-group">
<label for="test_id">Test</label>
<select name="test_id" id="test_id" class="form-control">
@foreach($tests as $test)
<option value="{{ $test->id }}" {{ $test->id == $exam->test_id ? 'selected' : '' }}>{{ $test->name }}</option>
@endforeach
</select>
</div>
```

```
<div class="form-group">
<label for="group_id">Grupa</label>
<select name="group_id" id="group_id" class="form-control">
@foreach($groups as $group)
<option value="{{ $group->id }}" {{ $group->id == $exam->group_id ? 'selected' : '' }}>{{ $group->name }}</option>
@endforeach
</select>
</div>

<div class="form-group">
<label for="start_date">Data rozpoczęcia</label>
<input type="date" name="start_date" id="start_date" class="form-control" value="{{ $exam->start_date }}" required>
</div>

<div class="form-group">
<label for="end_date">Data zakończenia</label>
<input type="date" name="end_date" id="end_date" class="form-control" value="{{ $exam->end_date }}" required>
</div>

<button type="submit" class="btn btn-primary">Zapisz zmiany</button>
</form>
</div>
</div>
</div>
</div>
</div>
@endsection
```

Rozwiązanie egzaminu

Migracje

```
$ php artisan make:migration create_assigned_exams_table --create=assigned_exams
```

💻 database/migrations/2024_04_28_101318_create_assigned_exams_table.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
```

```php
return new class extends Migration
{
public function up(): void
{
Schema::create('assigned_exams', function (Blueprint $table) {
$table->id();
$table->foreignId('user_id')->constrained()->onDelete('cascade');
$table->foreignId('exam_id')->nullable()->constrained()->onDelete('set null');
$table->string('name');
$table->boolean('finished')->default(false);

$table->timestamps();

});
}
public function down(): void
{
Schema::dropIfExists('assigned_exams');
}
};
```

```
$ php artisan make:migration create_assigned_questions_table --create=assigned_questions
```

database/migrations/2024_04_28_101907_create_assigned_questions_table.php

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
public function up(): void
{
Schema::create('assigned_questions', function (Blueprint $table) {
$table->id();
$table->foreignId('assigned_exam_id')->constrained()->onDelete('cascade');
$table->text('content');
$table->boolean('multiselect');
$table->boolean('anwsered')->default(false);
$table->timestamps();
});
}
```

```
public function down(): void
{
Schema::dropIfExists('assigned_questions');
}
};
```

```
$ php artisan make:migration create_assigned_answers_table --create=assigned_answers
```

database/migrations/2024_04_28_102120_create_assigned_answers_table.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
public function up(): void
{
Schema::create('assigned_answers', function (Blueprint $table) {
$table->id();
$table->text('content');
$table->boolean('correct');
$table->boolean('user_answer')->nullable()->default(null);
$table->foreignId('assigned_question_id')->constrained()->onDelete('cascade');
$table->timestamps();
});
}

public function down(): void
{
Schema::dropIfExists('assigned_answers');
}
};
```

Modele

```
$ php artisan make:model AssignedExam
```

app/Models/AssignedExam.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class AssignedExam extends Model
{
use HasFactory;
protected $fillable = [
'user_id',
'exam_id',
'name',
'finished',
];

public function user()
{
return $this->belongsTo(User::class);
}

public function exam()
{
return $this->belongsTo(Exam::class);
}

public function assignedQuestions()
{
return $this->hasMany(AssignedQuestion::class);
}

public function result()
{
$available = $this->assignedQuestions->count();
$earned = 0;
foreach ($this->assignedQuestions()->where('answered', true)->get() as $question) {
$earned += $question->result();
}
return $earned / $available;
}
}
```

app/Models/User.php

```
...
public function assignedExams()
{
return $this->hasMany(AssignedExam::class);
}
...
```

app/Models/Exam.php

```
...
public function assignedExams()
{
return $this->hasMany(AssignedExam::class);
}
...
```

```
$ php artisan make:model AssignedQuestion
```

app/Models/AssignedQuestion.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class AssignedQuestion extends Model
{
use HasFactory;
protected $fillable = [
'content',
'multiselect',
'answered',
'assigned_exam_id'
];

public function assignedExam()
{
return $this->belongsTo(AssignedExam::class);
}
```

```
public function assignedAnswers()
{
return $this->hasMany(AssignedAnswer::class);
}
public function result()
{
if($this->multiselect){
$available = $this->assignedAnswers()->count();
$earned = 0;
foreach($this->assignedAnswers()->whereNotNull('user_anwser')->get() as $answer){
$earned += $answer->result();
}
return $earned / $available;
} else {
return $this->assignedAnswers()->where('correct', true)->where('user_answer', true)->count();
}
}
}
```

```
$ php artisan make:model AssignedAnswer
```

app/Models/AssignedAnswer.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class AssignedAnswer extends Model
{
use HasFactory;
protected $fillable = [
'content',
'correct',
'user_answer',
'assigned_question_id'
];

public function assignedQuestion()
{
return $this->belongsTo(AssignedQuestion::class);
}
```

```
public function result(){
return $this->user_answer == $this->correct ? 1 : 0;
}
}
```

Routing

routes/web.php

```
<?php

use App\Http\Controllers\ProfileController;
use Illuminate\Support\Facades\Route;
use App\Http\Middleware\TeacherMiddleware;
use App\Http\Controllers\UserController;
use App\Http\Controllers\GroupController;
use App\Http\Controllers\QuestionController;
use App\Http\Controllers\AnswerController;
use App\Http\Controllers\TestController;
use App\Http\Controllers\ExamController;
use App\Http\Controllers\AssignedExamController;

Route::get('/', function () {
return view('welcome');
});

Route::get('/dashboard', function () {
return view('dashboard');
})->middleware(['auth', 'verified'])->name('dashboard');

Route::middleware('auth')->group(function () {
Route::post('/exams/{exam}/start', [AssignedExamController::class, 'start'])->name('exams.start');
Route::get('/assigned-exams/{assignedExam}', [AssignedExamController::class, 'show'])->name('assigned_exams.show');
Route::patch('/assigned-exams/{assignedExam}/{assignedQuestion}/update', [AssignedExamController::class, 'update'])-
>name('assigned_questions.update');
Route::post('/assigned-exams/{assignedExam}/finish', [AssignedExamController::class, 'finish'])-
>name('assigned_exams.finish');
Route::get('/profile', [ProfileController::class, 'edit'])->name('profile.edit');
Route::patch('/profile', [ProfileController::class, 'update'])->name('profile.update');
Route::delete('/profile', [ProfileController::class, 'destroy'])->name('profile.destroy');
});

Route::middleware(['auth', TeacherMiddleware::class])->group(function () {
```

```php
// UŻYTKOWNICY
Route::get('/users', [UserController::class, 'index'])->name('users.index');
Route::get('/users/create', [UserController::class, 'create'])->name('users.create');
Route::post('/users', [UserController::class, 'store'])->name('users.store');
Route::get('/users/{user}/edit', [UserController::class, 'edit'])->name('users.edit');
Route::patch('/users/{user}', [UserController::class, 'update'])->name('users.update');
Route::delete('/users/{user}', [UserController::class, 'destroy'])->name('users.destroy');

// GRUPY
Route::get('/groups', [GroupController::class, 'index'])->name('groups.index');
Route::get('/groups/create', [GroupController::class, 'create'])->name('groups.create');
Route::post('/groups', [GroupController::class, 'store'])->name('groups.store');
Route::get('/groups/{group}/edit', [GroupController::class, 'edit'])->name('groups.edit');
Route::patch('/groups/{group}', [GroupController::class, 'update'])->name('groups.update');
Route::delete('/groups/{group}', [GroupController::class, 'destroy'])->name('groups.destroy');
Route::get('/groups/assign', [GroupController::class, 'assignForm'])->name('groups.assign.form');
Route::post('/groups/assign', [GroupController::class, 'assign'])->name('groups.assign');

// ODPOWIEDZI
Route::post('/questions/{question}/answers', [AnswerController::class, 'store'])->name('answers.store');
Route::get('/answers/{answer}/edit', [AnswerController::class, 'edit'])->name('answers.edit');
Route::patch('/answers/{answer}', [AnswerController::class, 'update'])->name('answers.update');
Route::delete('/answers/{answer}', [AnswerController::class, 'destroy'])->name('answers.destroy');

// PYTANIA
Route::get('/questions', [QuestionController::class, 'index'])->name('questions.index');
Route::get('/questions/create', [QuestionController::class, 'create'])->name('questions.create');
Route::post('/questions', [QuestionController::class, 'store'])->name('questions.store');
Route::get('/questions/{question}/edit', [QuestionController::class, 'edit'])->name('questions.edit');
Route::patch('/questions/{question}', [QuestionController::class, 'update'])->name('questions.update');
Route::delete('/questions/{question}', [QuestionController::class, 'destroy'])->name('questions.destroy');
Route::post('/questions/{question}/duplicate', [QuestionController::class, 'duplicate'])->name('questions.duplicate');

// TESTY
Route::get('/tests', [TestController::class, 'index'])->name('tests.index');
Route::get('/tests/create', [TestController::class, 'create'])->name('tests.create');
Route::post('/tests', [TestController::class, 'store'])->name('tests.store');
Route::get('/tests/{test}/edit', [TestController::class, 'edit'])->name('tests.edit');
Route::patch('/tests/{test}', [TestController::class, 'update'])->name('tests.update');
Route::delete('/tests/{test}', [TestController::class, 'destroy'])->name('tests.destroy');
Route::get('/tests/assign', [TestController::class, 'assignForm'])->name('tests.assign.form');
Route::post('/tests/assign', [TestController::class, 'assign'])->name('tests.assign');
Route::post('/tests/{test}/duplicate', [TestController::class, 'duplicate'])->name('tests.duplicate');
```

```php
// EGZAMINY
Route::get('/exams', [ExamController::class, 'index'])->name('exams.index');
Route::get('/exams/create', [ExamController::class, 'create'])->name('exams.create');
Route::post('/exams', [ExamController::class, 'store'])->name('exams.store');
Route::get('/exams/{exam}/edit', [ExamController::class, 'edit'])->name('exams.edit');
Route::patch('/exams/{exam}', [ExamController::class, 'update'])->name('exams.update');
Route::delete('/exams/{exam}', [ExamController::class, 'destroy'])->name('exams.destroy');

// WYNIKI
Route::get('/assigned-exams', [AssignedExamController::class, 'index'])->name('assigned_exams.index');
});

require __DIR__.'/auth.php';
```

Kontroler

```
$ php artisan make:controller AssignedExamController
```

app/Http/Controllers/AssignedExamController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\AssignedExam;
use App\Models\AssignedQuestion;
use App\Models\AssignedAnswer;
use App\Models\Exam;

class AssignedExamController extends Controller
{
public function index(){
$finishedExams = AssignedExam::all()->where('finished', true);
$unfinishedExams = AssignedExam::all()->where('finished', false);
return view('assigned_exams.index', compact('finishedExams', 'unfinishedExams'));
}
public function start(Request $request, Exam $exam)
{
$is_exam_active = $exam->start_date <= now() && $exam->end_date >= now();
if(! $is_exam_active) {
return redirect()->route('dashboard')->withErrors(['Ten egzamin nie jest aktywny.']);
```

```
}
$user_belongs_to_group_of_exam = $request->user()->groups()->where('id', $exam->group_id)->exists();
if (!$user_belongs_to_group_of_exam) {
return redirect()->route('dashboard')->withErrors(['Ten egzamin nie jest dla Twojej grupy.']);
}
$assignedExam = AssignedExam::where('exam_id', '=', $exam->id)->where('user_id', '=', $request->user()->id)->first();
if ($assignedExam != null) {
if($assignedExam->finished){
return redirect()->route('assigned_exams.show', $assignedExam)->with('success', 'Przegląd wyników.');
}
return redirect()->route('assigned_exams.show', $assignedExam)->with('success', 'Kontynuacja egzaminu.');
}

$assignedExam = AssignedExam::create([
'user_id' => $request->user()->id,
'exam_id' => $exam->id,
'name' => $exam->test->name,
]);

$questions = $exam->test->questions->shuffle();

foreach ($questions as $question) {
$assignedQuestion = AssignedQuestion::create([
'assigned_exam_id' => $assignedExam->id,
'content' => $question->content,
'multiselect' => $question->multiselect,
'answered' => false,
]);
$answers = $question->answers->shuffle();
foreach ($answers as $answer) {
AssignedAnswer::create([
'assigned_question_id' => $assignedQuestion->id,
'content' => $answer->content,
'correct' => $answer->correct,
'user_answer' => null,
]);
}
}
return redirect()->route('assigned_exams.show', $assignedExam)->with('success', 'Exam started.');
}

public function show(Request $request, AssignedExam $assignedExam){
if($assignedExam->user_id != $request->user()->id && $request->user()->role != 'teacher'){
```

```
return redirect()->route('dashboard')->withErrors(['To nie Twój egzamin.']);
}
$answeredQuestions = $assignedExam->assignedQuestions->where('answered', '=', true);
$unansweredQuestions = $assignedExam->assignedQuestions->where('answered', '=', false);
return view('assigned_exams.show', compact('assignedExam', 'answeredQuestions', 'unansweredQuestions'));
}
public function update(Request $request, AssignedExam $assignedExam, AssignedQuestion $assignedQuestion){
if($assignedExam->user_id != $request->user()->id){
return redirect()->route('dashboard')->withErrors(['To nie Twój egzamin.']);
}
if($assignedExam->finished){
return redirect()->route('dashboard')->withErrors(['Ten egzamin już został zakończony.']);
}
if($assignedQuestion->assigned_exam_id != $assignedExam->id){
return redirect()->route('dashboard')->withErrors(['To pytanie nie należy do egzaminu.']);
}
$assignedQuestion->answered = true;
if ($assignedQuestion->multiselect) {
$selectedAnswers = $request->input('answers', []);
foreach($assignedQuestion->assignedAnswers as $answer){
$answer->user_answer = in_array($answer->id, $selectedAnswers);
$answer->save();
}
} else {
$selectedAnswer = $request->input('answer', null);
$assignedQuestion->assignedAnswers()->update(['user_answer' => false]);
foreach($assignedQuestion->assignedAnswers as $answer){
$answer->user_answer = $answer->id == $selectedAnswer;
$answer->save();
}
}
$assignedQuestion->save();
return redirect()->route('assigned_exams.show', $assignedExam)->with('success', 'Zmiany zostały zapisane.');
}
public function finish(Request $request, AssignedExam $assignedExam){
if($assignedExam->user_id != $request->user()->id){
return redirect()->route('dashboard')->withErrors(['To nie Twój egzamin.']);
}
$assignedExam->finished = true;
$assignedExam->save();
return redirect()->route('assigned_exams.show', $assignedExam)->with('success', 'Egzamin zakończony.');
}
}
```

Widoki

```
$ php artisan make:view assigned_exams.show
```

resources/views/assigned_exams/show.blade.php

```
@extends('layouts.app')

@section('content')
<div class="container">
@if ($assignedExam->finished)
<div class="alert alert-success">
Przegląd wyników
</div>
@elseif (session('success'))
<div class="alert alert-success">
{{ session('success') }}
</div>
@endif
<div class="row justify-content-center">
<div class="col-md-8">
@if($unansweredQuestions->count() > 0)
<div class="card" style="background-color:#fa7066">
<div class="card-header">Pytania bez odpowiedzi</div>
<div class="card-body">
@foreach ($unansweredQuestions as $question)
<div class="card">
<div class="card-header">{{ $question->content }}{{$assignedExam->finished ? " (". str($question->result()) . ")" :
""}}</div>
<div class="card-body">
<form action="{{ route('assigned_questions.update', ['assignedExam' => $assignedExam, 'assignedQuestion' => $question])
}}" method="POST" style="display: inline;">
@csrf
@method('patch')
@foreach ($question->assignedAnswers as $answer)
<label style='background-color: {{ $assignedExam->finished ? '#fa7066' : 'white' }}'>
<input type='{{ $question->multiselect ? "checkbox" : "radio" }}'
name='{{ $question->multiselect ? "answers[]" : "answer" }}'
value='{{ $answer->id }}'
{{ $assignedExam->finished ? 'disabled': '' }} />
{{ $answer->content }}
</label>
<br />
```

```
@endforeach
@if(!$assignedExam->finished)
<button class="btn btn-sm btn-primary" type="submit">Zapisz odpowiedź</button>
@endif
</form>
</div>
</div>
@endforeach
</div>
</div>
@endif

@if(!$assignedExam->finished)
<div class="card">
<div class="card-header">Zakończ</div>
<div class="card-body">
<form action="{{ route('assigned_exams.finish', $assignedExam->id) }}" method="POST" style="display: inline;">
@csrf
@method('POST')
@if($unansweredQuestions->count() == 0)
<button type="submit" class="btn btn-sm btn-primary")" onclick="return confirm('Czy na pewno chcesz zakończyć ten
egzamin?')">Zakończ</button>
@else
<button type="submit" class="btn btn-sm btn-danger")" onclick="return confirm('Masz jeszcze nierozwiązane zadania. Czy
na pewno chcesz zakończyć ten egzamin?')">Zakończ</button>
@endif
</form>
</div>
</div>
@else
<div class="card">
<div class="card-header">Wynik</div>
<div class="card-body">
{{$assignedExam->result() * 100}}%
</div>
</div>
@endif

@if($answeredQuestions->count() > 0)
<div class="card">
<div class="card-header">Udzielone odpowiedzi</div>

<div class="card-body">
```

```
@foreach ($answeredQuestions as $question)
<div class="card">
<div class="card-header">{{ $question->content }}{{$assignedExam->finished ? " (". str($question->result()) . ")" :
""}}</div>
<div class="card-body">
<form action="{{ route('assigned_questions.update', ['assignedExam' => $assignedExam, 'assignedQuestion' => $question])
}}" method="POST" style="display: inline;">
@csrf
@method('patch')
@foreach ($question->assignedAnswers as $answer)
<label style='background-color: {{ $assignedExam->finished ? ($answer->result() == 1 ? '#7dfa57' : '#fa7066'): 'white'
}}'>
<input type='{{ $question->multiselect ? "checkbox" : "radio" }}'
name='{{ $question->multiselect ? "answers[]" : "answer" }}'
value='{{ $answer->id }}'
{{ $answer->user_answer==true ? 'checked' : '' }}
{{ $assignedExam->finished ? 'disabled': '' }} />
{{ $answer->content }}
</label>
<br />
@endforeach
@if(!$assignedExam->finished)
<button class="btn btn-sm btn-primary" type="submit">Zmień odpowiedź</button>
@endif
</form>
</div>
</div>
@endforeach
</div>
</div>
@endif
</div>
</div>
</div>
@endsection
```

```
$ php artisan make:view assigned_exams.index.php
```

```
@extends('layouts.app')
@php
use Illuminate\Support\Carbon;
@endphp
@section('content')
```

```
<div class="container">
<div class="card">
<div class="card-header">Zakończone egzaminy</div>
<table class="table card-body">
<thead>
<tr>
<th scope="col">Użytkownik</th>
<th scope="col">Nazwa</th>
<th scope="col">Data rozpoczęcia</th>
<th scope="col">Data zakończenia</th>
<th scope="col">Wynik</th>
<th scope="col">Akcje</th>
</tr>
</thead>
@foreach($finishedExams as $assignedExam)
<tr>
<td>{{$assignedExam->user->name}}</td>
<td>{{$assignedExam->name}}</td>
<td>{{ Carbon::parse($assignedExam->created_at)->format('Y-m-d H:i') }}</td>
<td>{{ Carbon::parse($assignedExam->updated_at)->format('Y-m-d H:i') }}</td>
<td>{{ round($assignedExam->result() * 100,2) }}%</td>
<td><a href="{{ route('assigned_exams.show', $assignedExam) }}" class="btn btn-sm btn-primary">Zobacz</a></td>
<tr>
@endforeach
</table>
</div>
<div class="card">
<div class="card-header">Niezakończone egzaminy</div>
<table class="table card-body">
<thead>
<tr>
<th scope="col">Użytkownik</th>
<th scope="col">Nazwa</th>
<th scope="col">Data rozpoczęcia</th>
<th scope="col">Data zakończenia</th>
<th scope="col">Wynik</th>
<th scope="col">Akcje</th>
</tr>
</thead>
@foreach($unfinishedExams as $assignedExam)
<tr>
<td>{{$assignedExam->user->name}}</td>
<td>{{$assignedExam->name}}</td>
```

```
<td>{{ Carbon::parse($assignedExam->created_at)->format('Y-m-d H:i') }}</td>
<td>-</td>
<td>{{ round($assignedExam->result() * 100,2) }}%</td>
<td><a href="{{ route('assigned_exams.show', $assignedExam) }}" class="btn btn-sm btn-primary">Zobacz</a></td>
<tr>
@endforeach
</table>
</div>
</div>
@endsection
```

resources/views/dashboard.blade.php

```
@extends('layouts.app')
@php
use Illuminate\Support\Carbon;
@endphp
@section('content')
<div class="container">
<div class="card">
<div class="card-header">Zakończone egzaminy</div>
<table class="table card-body">
<thead>
<tr>
<th scope="col">Użytkownik</th>
<th scope="col">Nazwa</th>
<th scope="col">Data rozpoczęcia</th>
<th scope="col">Data zakończenia</th>
<th scope="col">Wynik</th>
<th scope="col">Akcje</th>
</tr>
</thead>
@foreach($finishedExams as $assignedExam)
<tr>
<td>{{$assignedExam->user->name}}</td>
<td>{{$assignedExam->name}}</td>
<td>{{ Carbon::parse($assignedExam->created_at)->format('Y-m-d H:i') }}</td>
<td>{{ Carbon::parse($assignedExam->updated_at)->format('Y-m-d H:i') }}</td>
<td>{{ round($assignedExam->result() * 100,2) }}%</td>
<td><a href="{{ route('assigned_exams.show', $assignedExam) }}" class="btn btn-sm btn-primary">Zobacz</a></td>
<tr>
@endforeach
</table>
```

```
</div>
<div class="card">
<div class="card-header">Niezakończone egzaminy</div>
<table class="table card-body">
<thead>
<tr>
<th scope="col">Użytkownik</th>
<th scope="col">Nazwa</th>
<th scope="col">Data rozpoczęcia</th>
<th scope="col">Data zakończenia</th>
<th scope="col">Wynik</th>
<th scope="col">Akcje</th>
</tr>
</thead>
@foreach($unfinishedExams as $assignedExam)
<tr>
<td>{{$assignedExam->user->name}}</td>
<td>{{$assignedExam->name}}</td>
<td>{{ Carbon::parse($assignedExam->created_at)->format('Y-m-d H:i') }}</td>
<td>-</td>
<td>{{ round($assignedExam->result() * 100,2) }}%</td>
<td><a href="{{ route('assigned_exams.show', $assignedExam) }}" class="btn btn-sm btn-primary">Zobacz</a></td>
<tr>
@endforeach
</table>
</div>
</div>
@endsection
```

Trochę zmiany stylu

resources/views/layouts/app.blade.php

```
...
<head>
...
<style>
.card{
margin-bottom: 20px;
}
.blink_me {
animation: blinker 1s linear infinite;
}
```

```
@keyframes blinker {
50% {
opacity: 0;
}
}
</style>
</head>
...
```

Nawigacja

💻 resources/views/layouts/navigation.blade.php

```
...
<!-- Navigation Links -->
...
@if (Auth::user()->role === 'teacher')
...
<x-nav-link :href="route('assigned_exams.index')" :active="request()->routeIs('assigned_exams.index')">
{{ __('Wyniki') }}
</x-nav-link>
@endif
...
<!-- Responsive Navigation Menu -->
...
@if (Auth::user()->role === 'teacher')
...
<x-responsive-nav-link :href="route('assigned_exams.index')" :active="request()->routeIs('assigned_exams.index')">
{{ __('Wyniki') }}
</x-responsive-nav-link>
@endif
...
```

Testowanie

```
$ php artisan serve
```

ERD

```
$ schemacrawler.sh --server=sqlite --database=/var/www/l3z1/database/database.sqlite --info-level=maximum \
  --command=schema --output-format=scdot --output-file=erd.scdot
```

```
$ dot -Tpng erd.scdot -o erd.png
```