САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа 2

Выполнил:

Сахно Ярослав

Группа К3341

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

- Реализовать все модели данных, спроектированные в рамках ДЗ1;
- Реализовать набор из CRUD-методов для работы с моделями данных;
- Реализовать API-эндпоинт для получения пользователя по id/email.

Ход работы

1. Реализация моделей

Пример реализованной модели - User (см. Листинг 1).

Листинг 1 - Модель User:

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\BelongsToMany;
use Illuminate\Database\Eloquent\SoftDeletes;
use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable
{
    use HasFactory, SoftDeletes;

    /**
     * The attributes that are mass assignable.
     *
```

```php
     * @var array<int, string>
     */
    protected $fillable = [
        'id',
        'first_name',
        'last_name',
        'middle_name',
        'password',
        'image',
        'user_type_id',
        'ip',
        'is_confirmed',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    protected $casts = [
        'password' => 'hashed',
    ];

    public $incrementing = false;

    public function getFullNameAttribute(): string
    {
```

```
        return $this->last_name.' '.$this->first_name.'
'.$this->middle_name;

    }



    public function user_type(): belongsTo

    {

        return $this->belongsTo(UserType::class);

    }



    public function courses(): belongsToMany

    {

        return $this->belongsToMany(Course::class);

    }

}
```

Реализованы поля модели, а также выделены связи модели User с другими моделями.

Аналогичным образом были реализованы оставшиеся модели Attempt, Course, Homework, Task, Attempt, UserType.

2. Реализация CRUD-методов

Далее был реализован набор из CRUD-методов для работы с каждой моделью данных.

Для каждой модели были реализованы методы для создания, получения, редактирования и удаления. Логика каждого метода реализована в соответствующем контроллере (пример контроллера для пользователя см. на Листинге 2).

Листинг 2 - Контроллер методов для работы с пользователем:

```
<?php
```

```php
namespace App\Http\Controllers;


use App\Actions\AttachCoursesAction;

use App\Actions\DetachCoursesAction;

use App\Http\Requests\EditUserRequest;

use App\Http\Requests\RegistrationRequest;

use App\Models\Course;

use App\Models\User;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Cache;

use Illuminate\Support\Facades\Storage;


class UserController extends Controller

{

    /**

     * Display a listing of the resource.

     */

    public function index(Request $request)

    {

        $page = $request->input('page', 1);


        $cacheKey =
"cached_users_{$request->input('type')}_page_$page";


        if (Cache::has($cacheKey)) {

            $users = Cache::get($cacheKey);

        } else {
```

```
            $users = User::when($request->has('type'), function
($query) use ($request) {

                $query->where('user_type_id',
$request->input('type'));

            })

                ->where('user_type_id', '!=', 1)

                ->with('user_type')

                ->orderBy('created_at')


->paginate(config('constants.options.paginate_number'));



            Cache::put($cacheKey, $users);

        }



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

    }



    /**

     * Show the form for creating a new resource.

     */

    public function create()

    {

        return view('admin.create_user', ['courses' =>
Course::all()]);

    }



    /**

     * Store a newly created resource in storage.
```

```
    */
    public function store(
        RegistrationRequest $request,
        AttachCoursesAction $attachCoursesAction
    ) {
        $filename_to_store = '';

        if ($request->hasFile('image')) {
            $filename_with_extension =
$request->file('image')->getClientOriginalName();

            $filename = pathinfo($filename_with_extension,
PATHINFO_FILENAME);

            $extension =
$request->file('image')->getClientOriginalExtension();

            $filename_to_store =
'avatars/'.$filename.'_'.uniqid().'.'.$extension;

            Storage::put($filename_to_store,
fopen($request->file('image'), 'r+'));
        }

        $user = User::create($request->except('image') + ['ip' =>
$request->ip(), 'image' => $filename_to_store]);

        $attachCoursesAction($request, $user, true);
```

```
        if (! $request->get('is_confirmed')) {

            session()->flash('success', 'Заявка на регистрацию
отправлена');



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

        }

        session()->flash('success', 'Пользователь создан');



        return redirect()->route('users.index');

    }



    /**

     * Show the form for editing the specified resource.

     */

    public function edit(string $id)

    {

        $unattachedCourses = Course::whereDoesntHave('users',
function ($query) use ($id) {

            $query->where('users.id', $id);

        })->get();



        return view('admin.edit_user', ['user' => User::find($id),
'unattachedCourses' => $unattachedCourses]);

    }



    /**

     * Update the specified resource in storage.

     */
```

```php
    public function update(
        EditUserRequest $request,
        AttachCoursesAction $attachCoursesAction,
        DetachCoursesAction $detachCoursesAction,
        string $id
    ) {
        $user = User::find($id);
        $user->update($request->input());

        $attachCoursesAction($request, $user, true);
        $detachCoursesAction($request, $user);

        return redirect()->route('users.index')->with('success',
'Изменения сохранены');
    }

    /**
     * Remove the specified resource from storage.
     */
    public function destroy(string $id)
    {
        User::destroy($id);

        return redirect()->route('users.index')->with('success',
'Пользователь удален');
    }
}
```

3. Реализация API-эндпоинта для получения пользователя по id/email

Реализовано в контроллере

Вывод

В рамках работы были реализованы модели данных в соответствие со схемой из ДЗ1, реализованы CRUD-методы для всевозможной работы со всеми данными.