**Поволжский Государственный Университет Телекоммуникаций и Информатики**Кафедра «Программная инженерия»

|  |  |
| --- | --- |
| Сдана на проверку  «\_\_» \_\_\_\_2024 г. | Допустить к защите  «\_\_» \_\_\_\_\_ 2024 г.  Защищена с оценкой  «\_\_\_» \_\_\_\_\_\_\_\_ 2024 г. |

**КУРСОВАЯ РАБОТА**По дисциплине: «Прикладное программирование»  
На тему: «Разработка клиент-серверного web-приложения трэкер задач»   
  
Пояснительная записка

|  |  |
| --- | --- |
| Студент группы ИВТ-13\_\_\_\_\_\_\_ Аврусевич Ю. С.  (роспись) (ФИО)  Руководитель \_\_\_\_\_\_\_\_ к.т.н., доц. Ахметшина Э. Г.  (роспись) (ФИО) | 0 210147 0  (№ зачетной книжки) |

Самара 2024

# Рецензия

# **Содержание**

[Описание предметной области. Актуальность 4](#_Toc3)

[Описание программы. Общие сведения 5](#_Toc4)

[Функциональное назначение 5](#_Toc5)

[Логическая модель базы данных 7](#_Toc6)

[Физическая модель базы данных 9](#_Toc7)

[Диаграмма классов 10](#_Toc8)

[Диаграмма компонентов 11](#_Toc9)

[Диаграмма вариантов использования 12](#_Toc10)

[Диаграмма последовательности 13](#_Toc11)

[Демонстрация работы приложения 14](#_Toc12)

[Список используемых источников 22](#_Toc13)

[Приложение А – Листинг программного кода 23](#_Toc14)

# Описание предметной области. Актуальность

Предметная область управления задачами представляет собой неотъемлемую часть процесса организации работы как в профессиональной, так и в личной сфере. С увеличением объема информации и сложности задач, становится критически важным иметь эффективные инструменты для управления и координации задач.

В современном мире бизнеса и технологий, где конкуренция неумолимо растет, эффективное управление задачами становится ключевым фактором успеха. Компании и организации нуждаются в системах, способных оптимизировать процессы управления задачами, улучшая продуктивность и качество работы.

Актуальность проекта TaskForge выражается в его способности предоставлять пользователю инструмент, который не только позволяет эффективно управлять задачами, но также обеспечивает прозрачность и удобство в работе. В сравнении с традиционными методами управления задачами, такими как использование бумажных списков или электронных таблиц, TaskForge предлагает более интуитивный и мощный инструмент для организации и отслеживания задач.

Рост числа фрилансеров, удаленных сотрудников и команд, работающих на расстоянии, также подчеркивает актуальность необходимости эффективного инструмента для управления задачами. TaskForge может быть ключевым решением для таких команд, предоставляя им возможность легко синхронизировать и координировать задачи, даже если их члены находятся в разных часовых поясах или работают удаленно.

Таким образом, разработка проекта TaskForge становится не только ответом на актуальные потребности пользователей в управлении задачами, но и важным шагом в направлении создания инновационного и эффективного инструмента, способного помочь пользователям достичь своих целей и увеличить продуктивность работы.

# Описание программы. Общие сведения

Приложение – веб-ресурс. Установка не требуется.

Язык программирования: TypeScript, PHP.

Среда разработки: PHP Storm

Объем проекта: 219 МБ (вместе с исходным кодом).

Исходный код (Java и TypeScript): 1864 строк.

# Функциональное назначение

Проект TaskForge предназначен для обеспечения пользователей интуитивно понятным и удобным инструментом для эффективного управления задачами. Основные функции включают в себя:

1. Регистрация и авторизация: Пользователи могут зарегистрироваться в системе, создав учетную запись, и авторизоваться для доступа к функционалу TaskForge.
2. Управление проектами: Возможность создания, редактирования и удаления проектов, в которых будут группироваться задачи. Пользователи могут назначать проекты для различных целей, таких как работа, учеба, личные дела и т.д.
3. Управление задачами: Возможность создания новых задач внутри проектов, определения их приоритета, описания, сроков выполнения и привязки к определенным проектам.
4. Отслеживание выполнения задач: Пользователи могут отмечать задачи как выполненные, обновлять их статусы и вносить изменения в детали задачи.

Технологический стек включает в себя:

* **Фронтенд**: Использовался фреймворк Vue.js для разработки клиентской части приложения, обеспечивая быструю и отзывчивую интерактивность.
* **Бэкенд**: Для реализации серверной логики и взаимодействия с базой данных использовался фреймворк Laravel, обеспечивающий удобное и надежное API.
* **База данных**: Для хранения данных о пользователях, проектах и задачах использовалась система управления базами данных Postgres, обеспечивающая надежное хранение и быстрый доступ к информации.

# Логическая модель базы данных

1. users:

* id: Уникальный идентификатор пользователя (целое число, автоинкремент).
* name: Имя пользователя (строка, не пустая).
* email: Адрес электронной почты пользователя (строка, уникальная).
* password: Хэш пароля пользователя (строка, не пустая).
* created\_at: Дата и время создания записи о пользователе (timestamp).
* updated\_at: Дата и время последнего обновления записи о пользователе (timestamp).

1. projects:

* id: Уникальный идентификатор проекта (целое число, автоинкремент).
* created\_at: Дата и время создания проекта (timestamp).
* updated\_at: Дата и время последнего обновления проекта (timestamp).
* name: Название проекта (строка, не пустая).
* description: Описание проекта (текст, не пустой).
* expiry\_at: Дата и время истечения срока проекта (timestamp).
* is\_public: Флаг общедоступности проекта (булево значение, не пустое).
* owner\_id: Идентификатор владельца проекта (целое число, ссылается на id пользователя).

1. tasks:

* id: Уникальный идентификатор задачи (целое число, автоинкремент).
* created\_at: Дата и время создания задачи (timestamp).
* updated\_at: Дата и время последнего обновления задачи (timestamp).
* title: Название задачи (строка, не пустая).
* done: Флаг завершенности задачи (булево значение, не пустое).
* project\_id: Идентификатор проекта, к которому относится задача (целое число, ссылается на id проекта).
* expiry\_at: Дата и время истечения срока задачи (timestamp).

Связи между сущностями:

1. Связь "Один ко многим" между пользователем и проектами:

* Тип связи: Один пользователь может быть владельцем множества проектов, но каждый проект принадлежит только одному пользователю.
* Реализация: Поле owner\_id в таблице проектов ссылается на id пользователя.

1. Связь "Один ко многим" между проектами и задачами:

* Тип связи: Один проект может содержать множество задач, но каждая задача принадлежит только одному проекту.
* Реализация: Поле project\_id в таблице задач ссылается на id проекта.

# Физическая модель базы данных

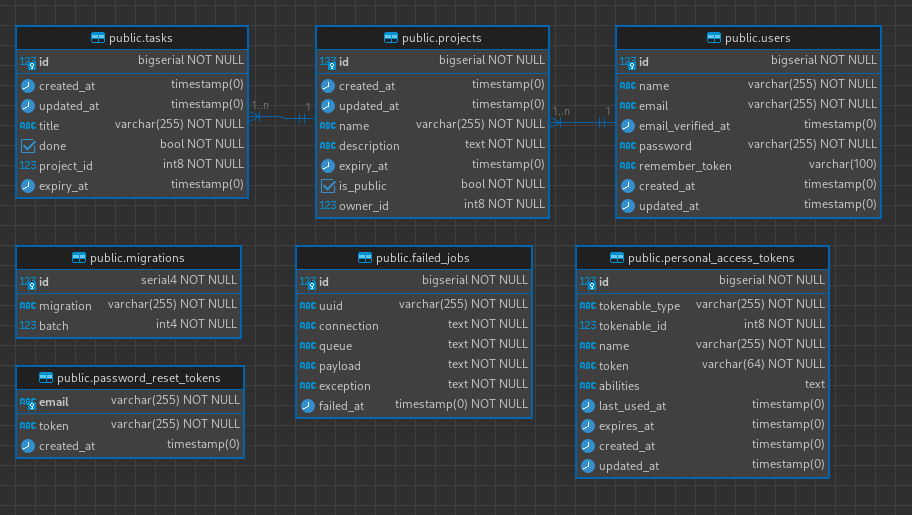


Рис. 1.1– Физическая модель базы данных

После создания базы данных в приложении, у нас добавились системные таблицы:

* Migrations
* Password\_reset\_tokens
* Failed\_jobs
* Personal\_access\_tokens

# Диаграмма классов

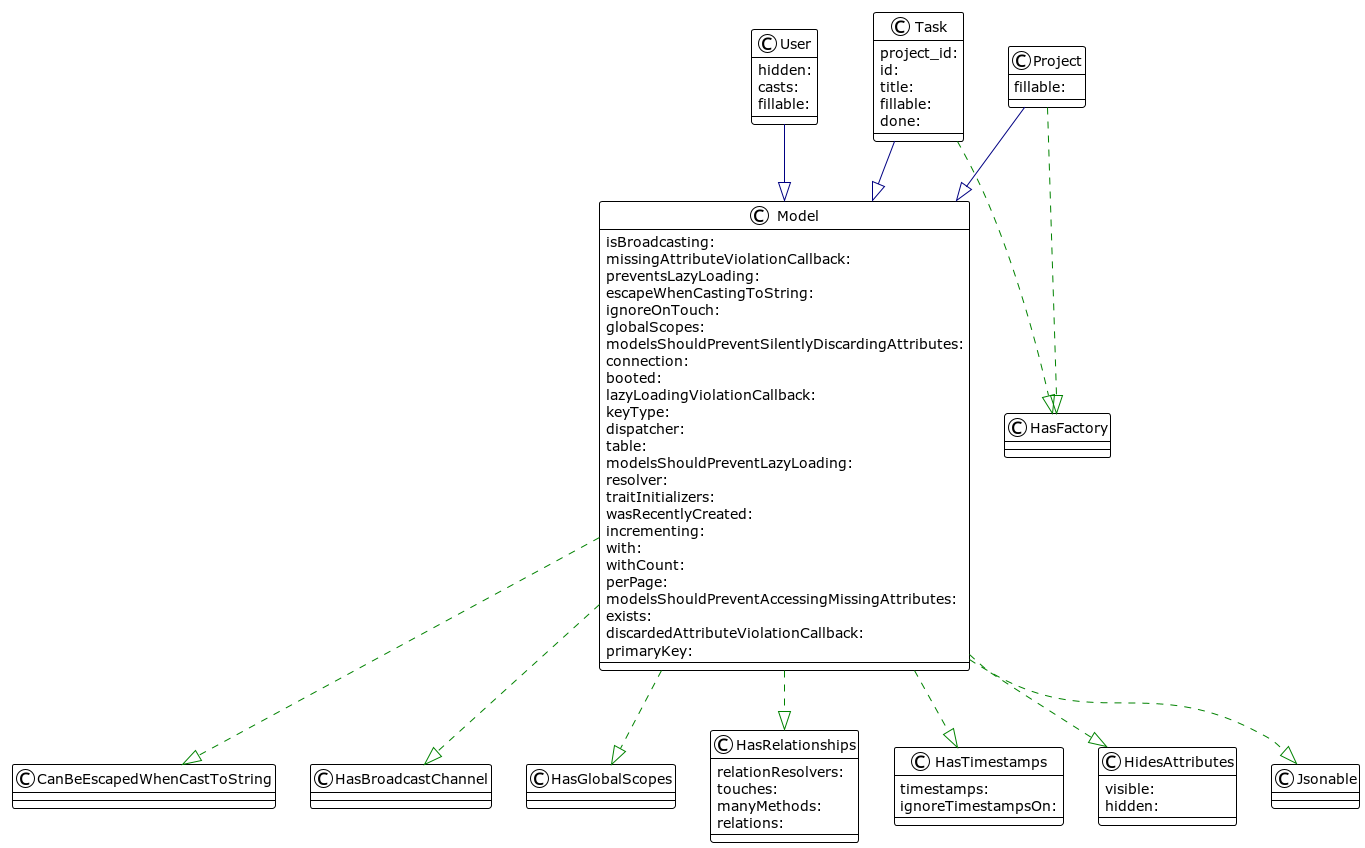


Рис. 2.1 – Диаграмма классов

# Диаграмма компонентов

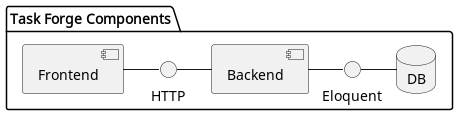


Рис. 3.1 – Диаграмма компонентов

# Диаграмма вариантов использования

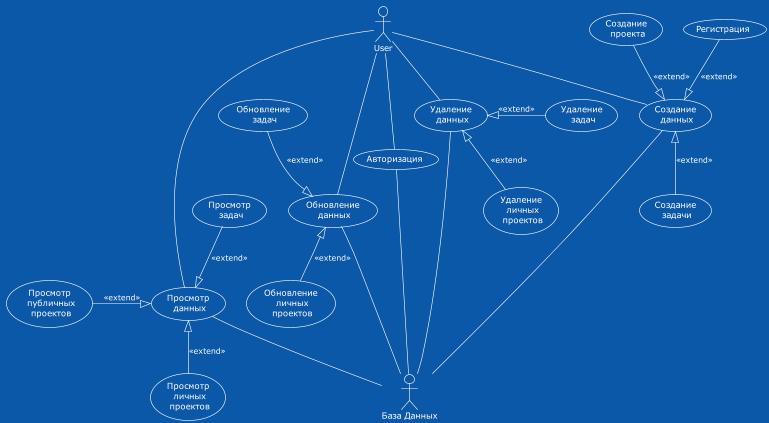


Рис. 4.1 – Диаграмма вариантов использования

# Диаграмма последовательности

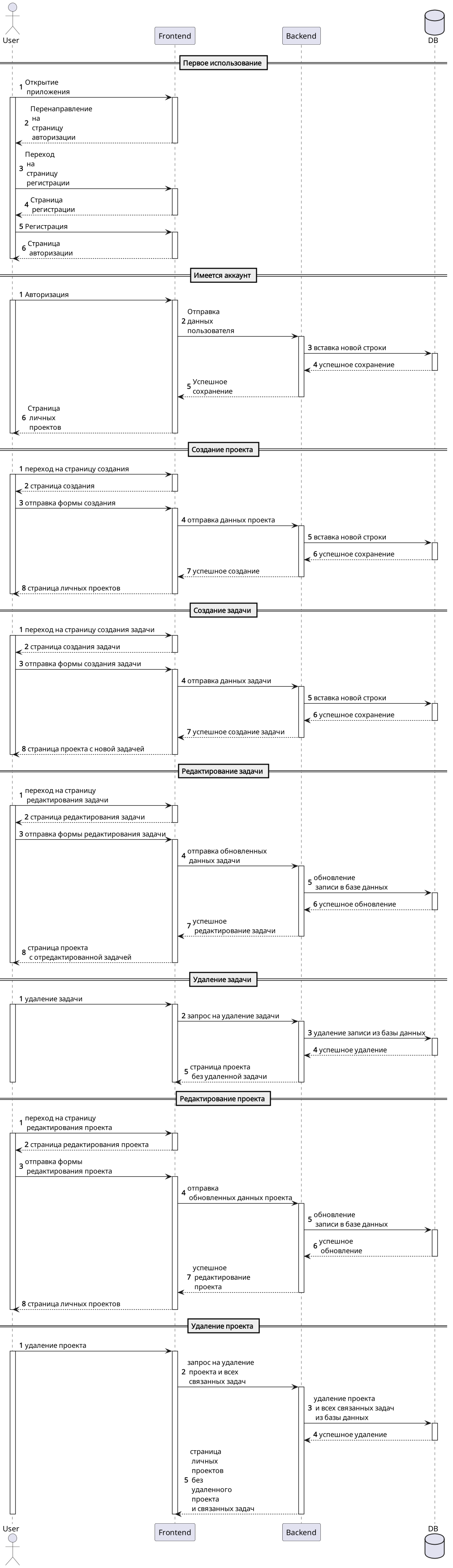


Рис. 5.1 – Диаграмма последовательности

# Демонстрация работы приложения

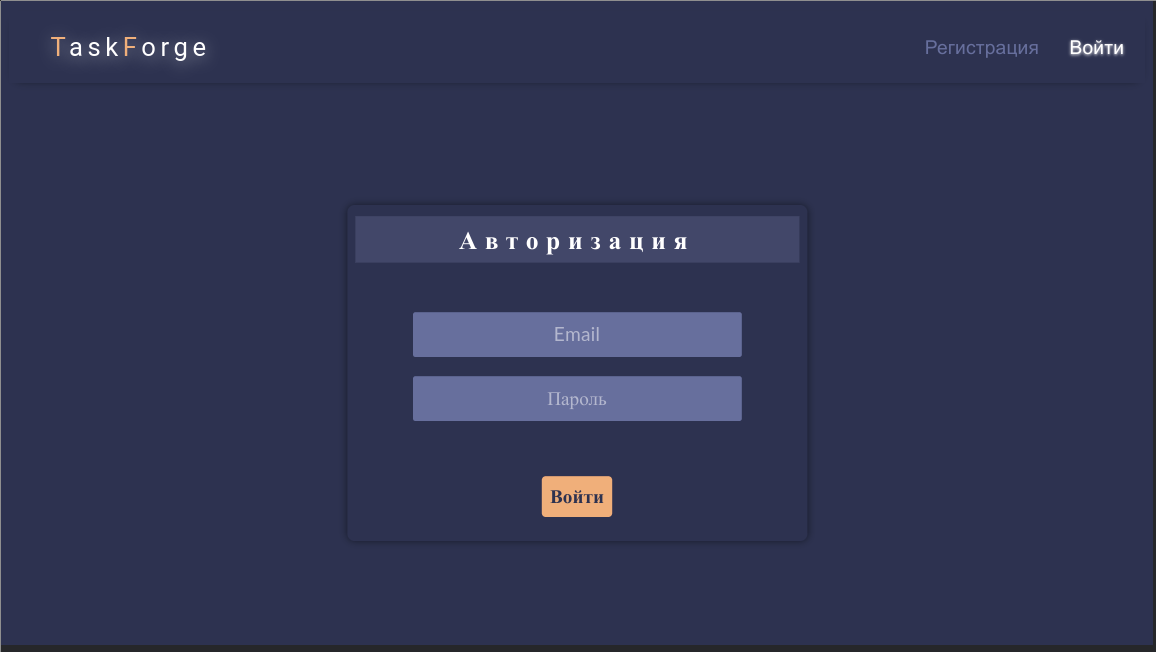


Рис. 6.1 – Первое открытие приложения

При открытии web-приложения в браузере, пользователь попадает на страницу с формой для авторизации (Рис. 6.1)

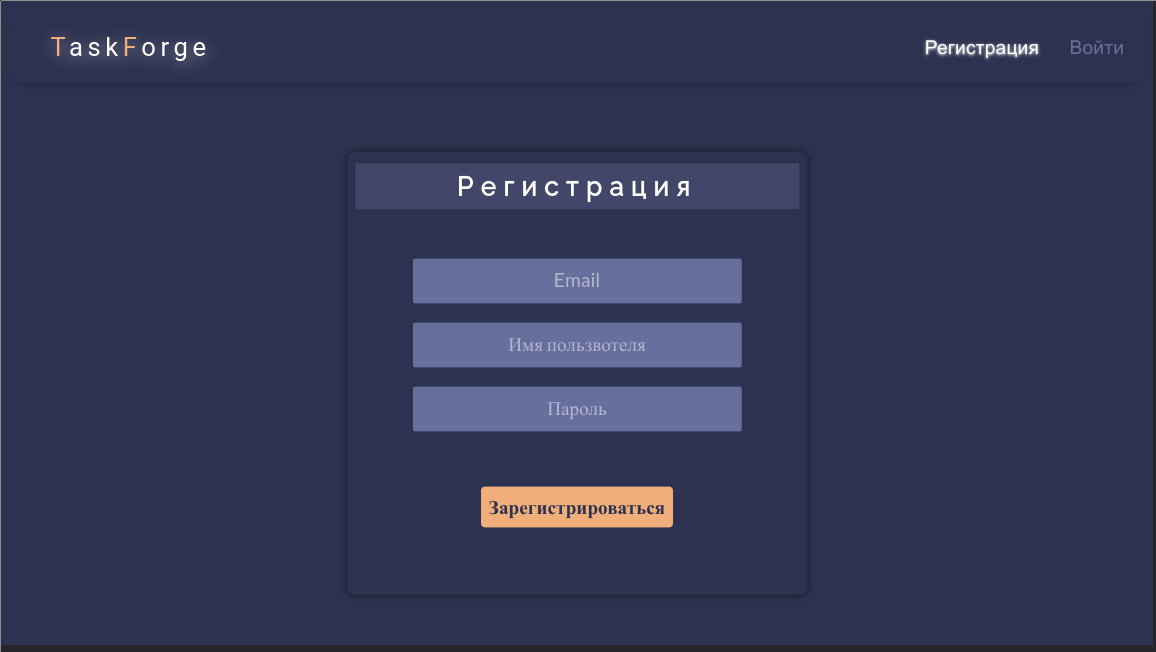


Рис. 6.2 – Регистрация

Чтобы использовать систему, нам следует зарегистрироваться. (Рис. 6.2)

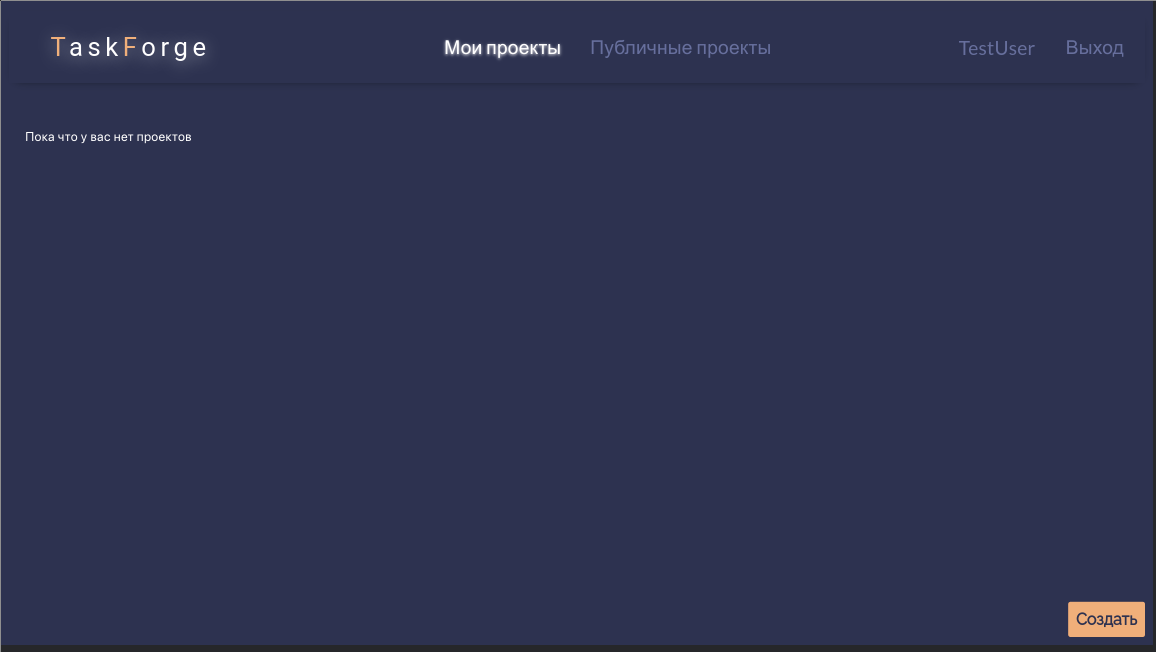


Рис. 6.3 – Страница мои проекты

После создания аккаунта, мы авторизуемся и попадаем на страницу «Мои проекты». На данный момент здесь ничего нет. (Рис. 6.3)

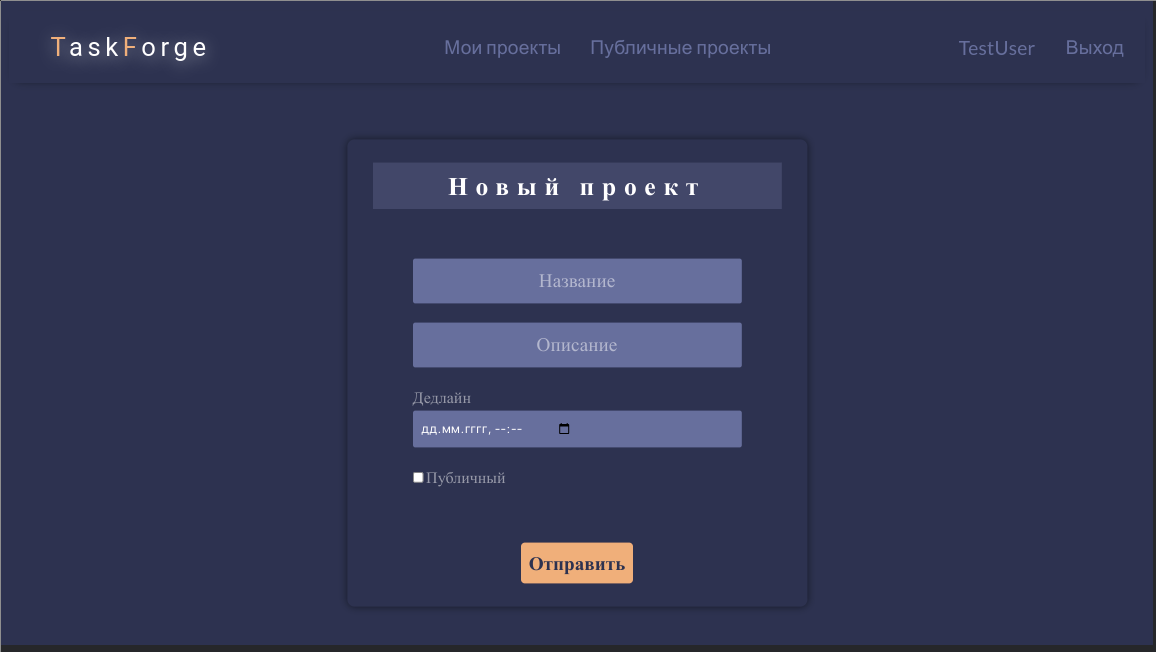


Рис. 6.4 – Страница создания проекта

При нажатии на кнопку «Создать» на странице «Мои проекты» пользователь попадает на страницу с формой создания проекта. (Рис. 6.4)

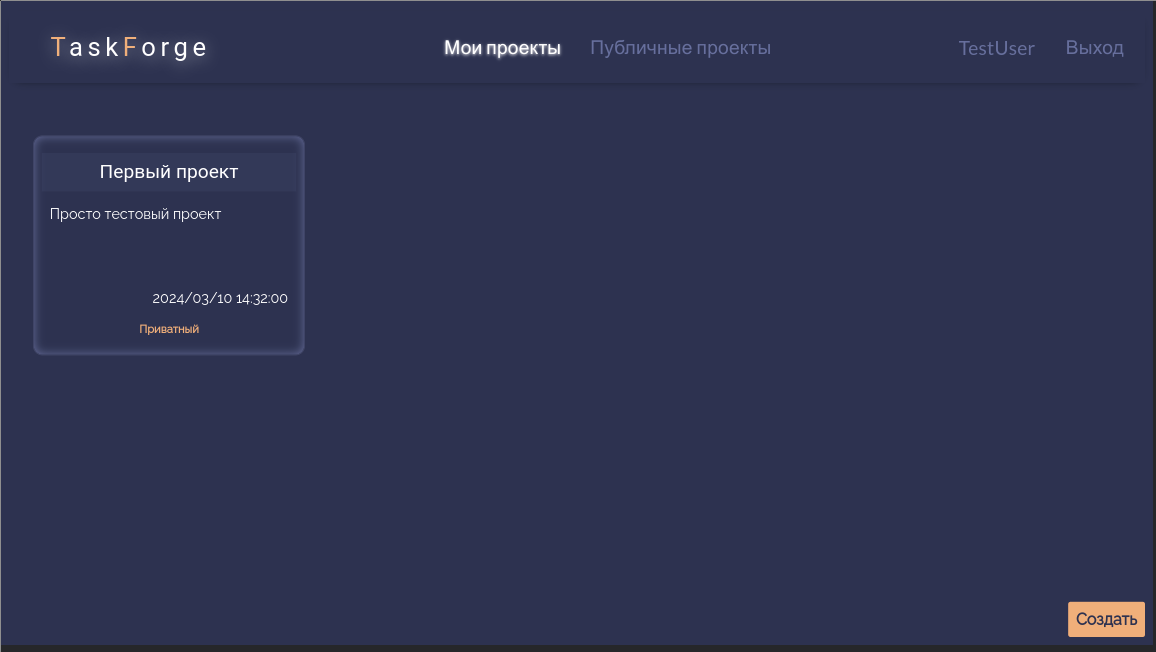


Рис. 6.5 – Созданный проект

После ввода и отправка формы на сервер, пользователь попадет на страницу «Мои проекты», где он видит карточку созданного проекта. (Рис. 6.5)

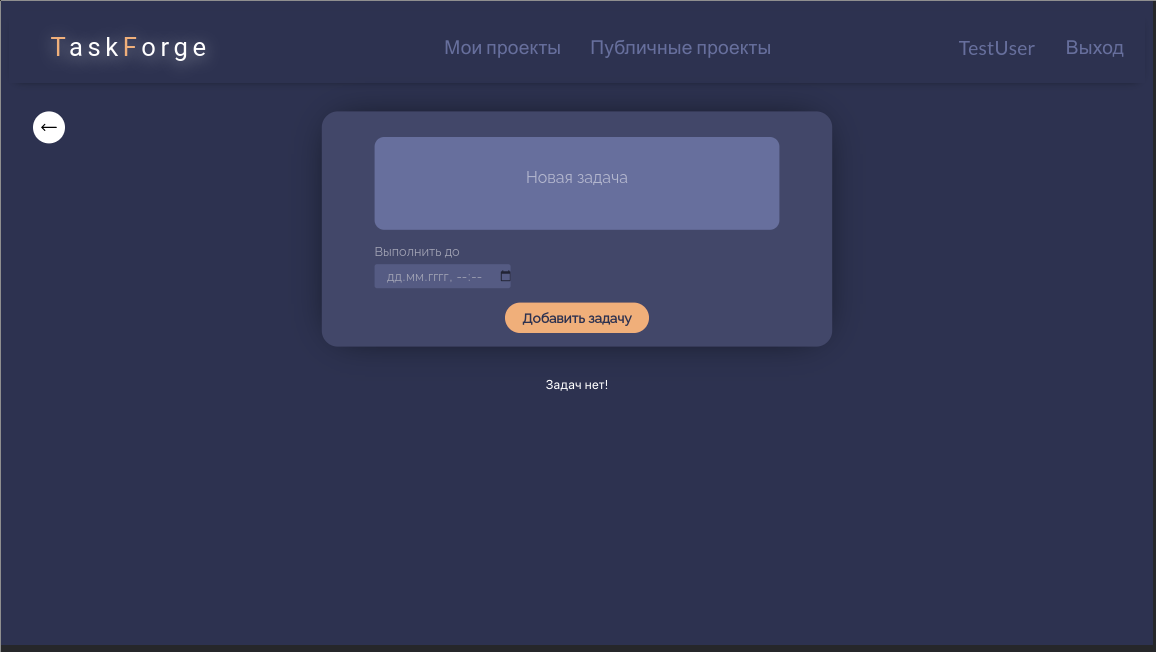


Рис. 6.6 – Открытие проекта

После нажатия на карточку проекта мы попадаем на страницу, где можно увидеть задачи и их статусы, но пока что здесь нет задач. (Рис. 6.6)

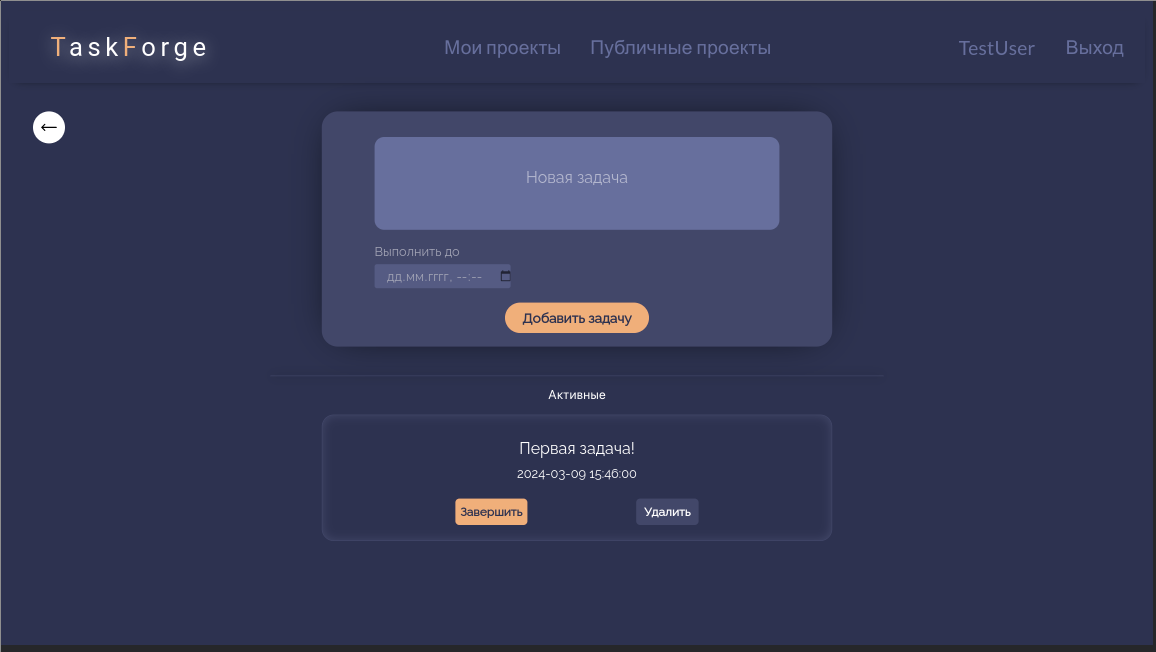


Рис. 6.7 – Создание задачи

После заполнения формы для создания задачи и отправки ее на сервер, мы можем видеть новую задачу в списке. Сейчас она активная. (Рис. 6.7)

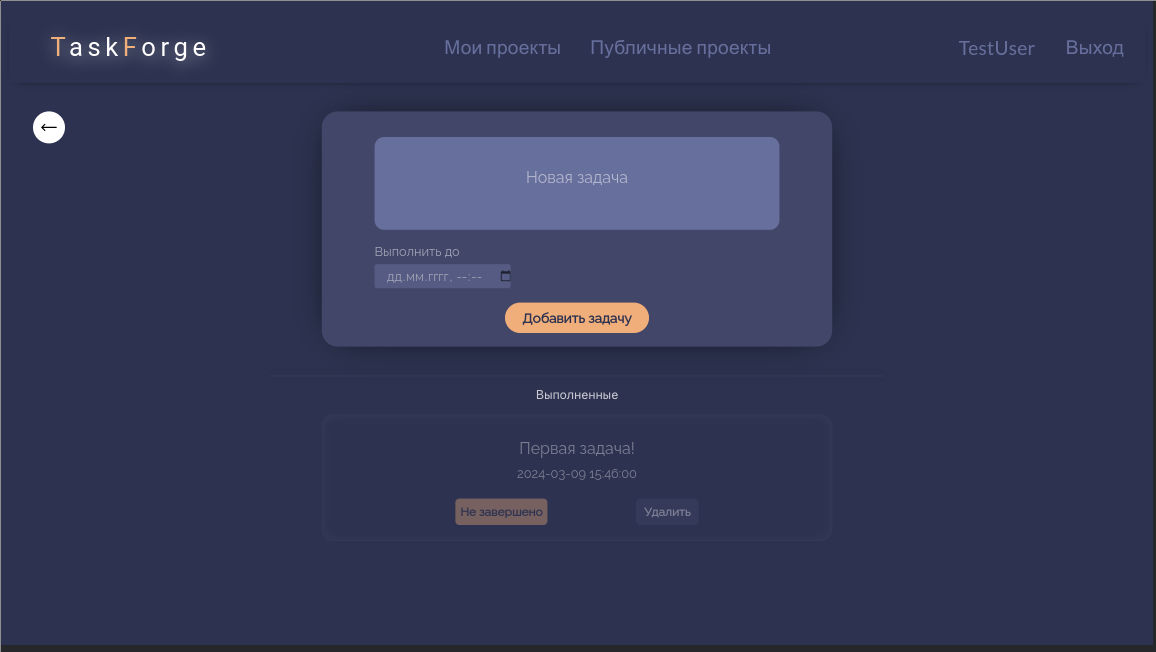


Рис. 6.8 – Выполненнная задача

После нажатия «Завершить», задача принимает статус «завершено» и становится затемненной. (Рис. 6.8)

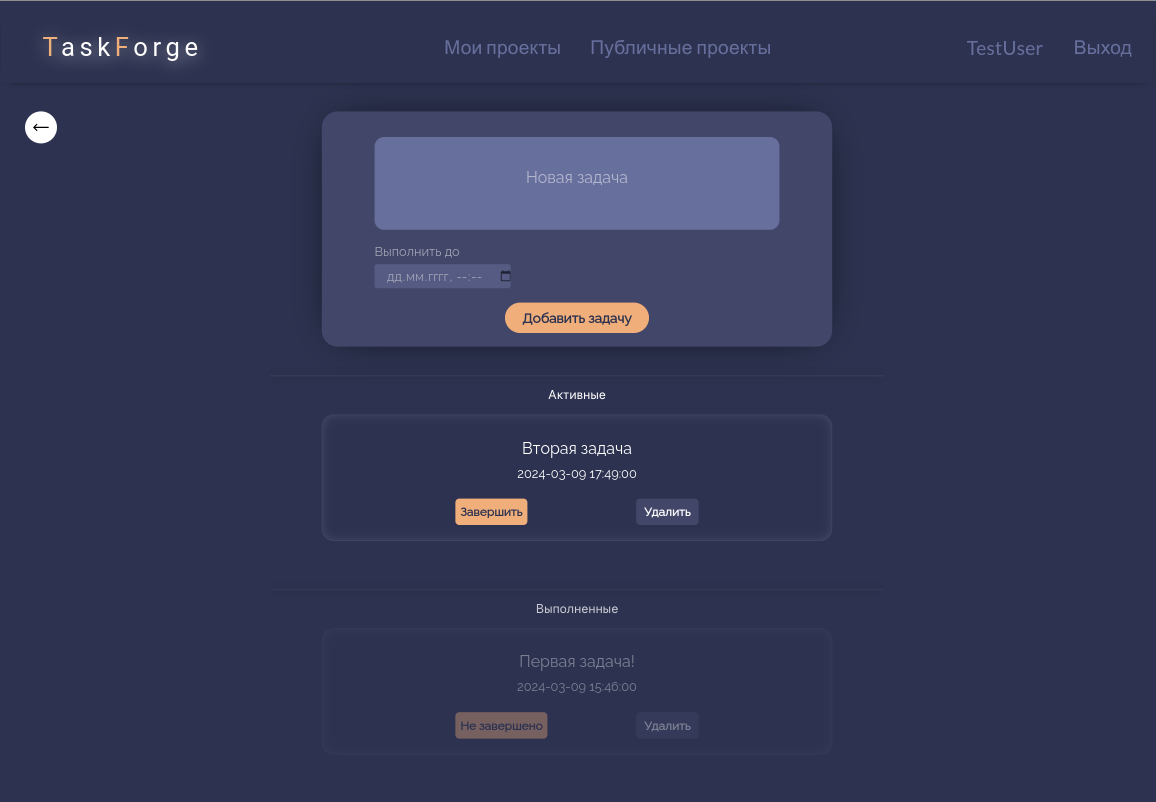


Рис. 6.9 – Создание второй задачи

Следующим пунктом мы создаем вторую задачу и видим, что первая задача располагается в отдельной секции, помеченной как «Выполненные». (Рис. 6.9)

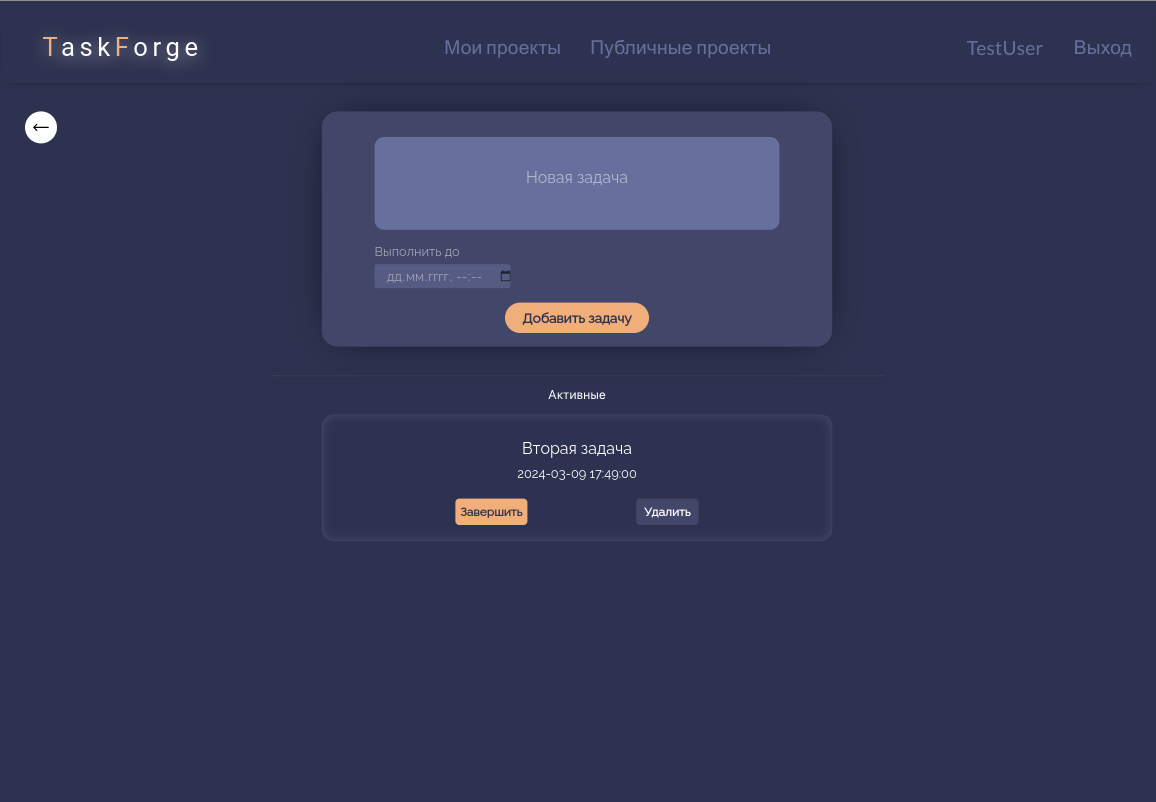


Рис. 6.10 – Удаление задачи

Нажатием кнопки «Удалить» на карточке задачи мы удалили первую задачу. (Рис. 6.10)

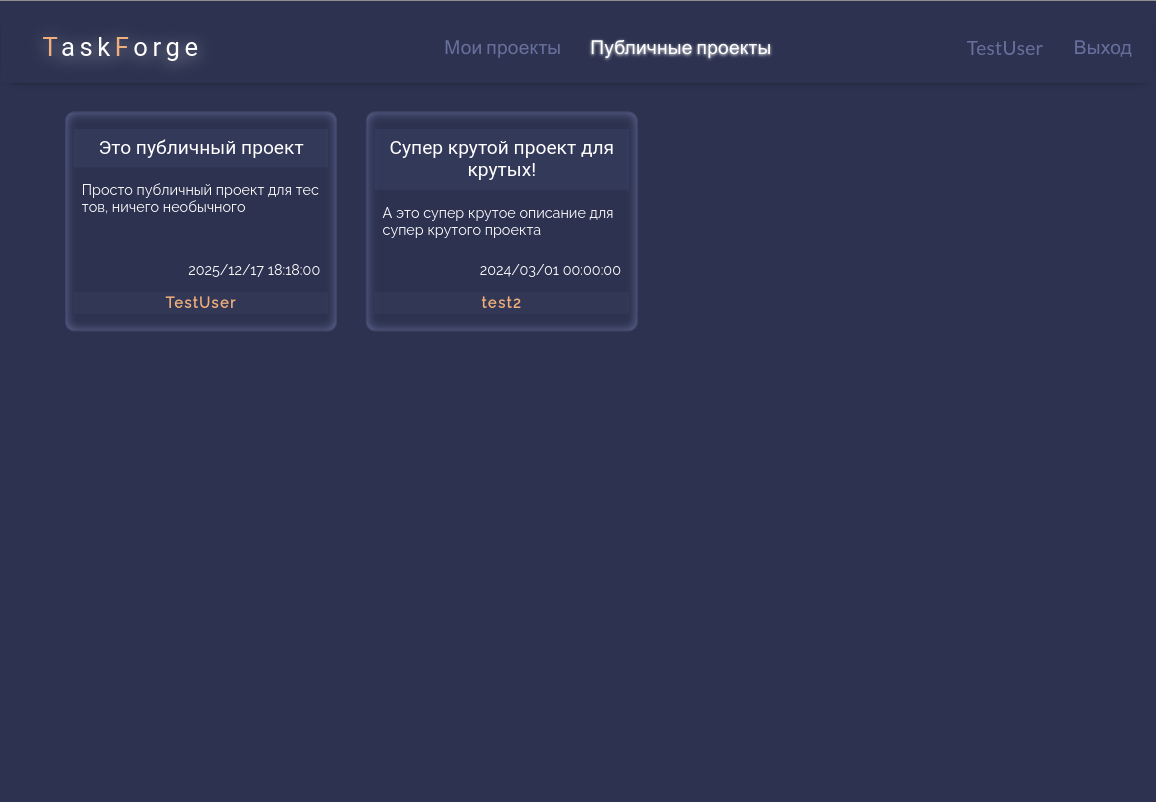


Рис. 6.11 – Публичные проекты

Мы создали второй аккаунт в системе и добавили туда проект с пометкой «Публичный». (Рис. 6.11)

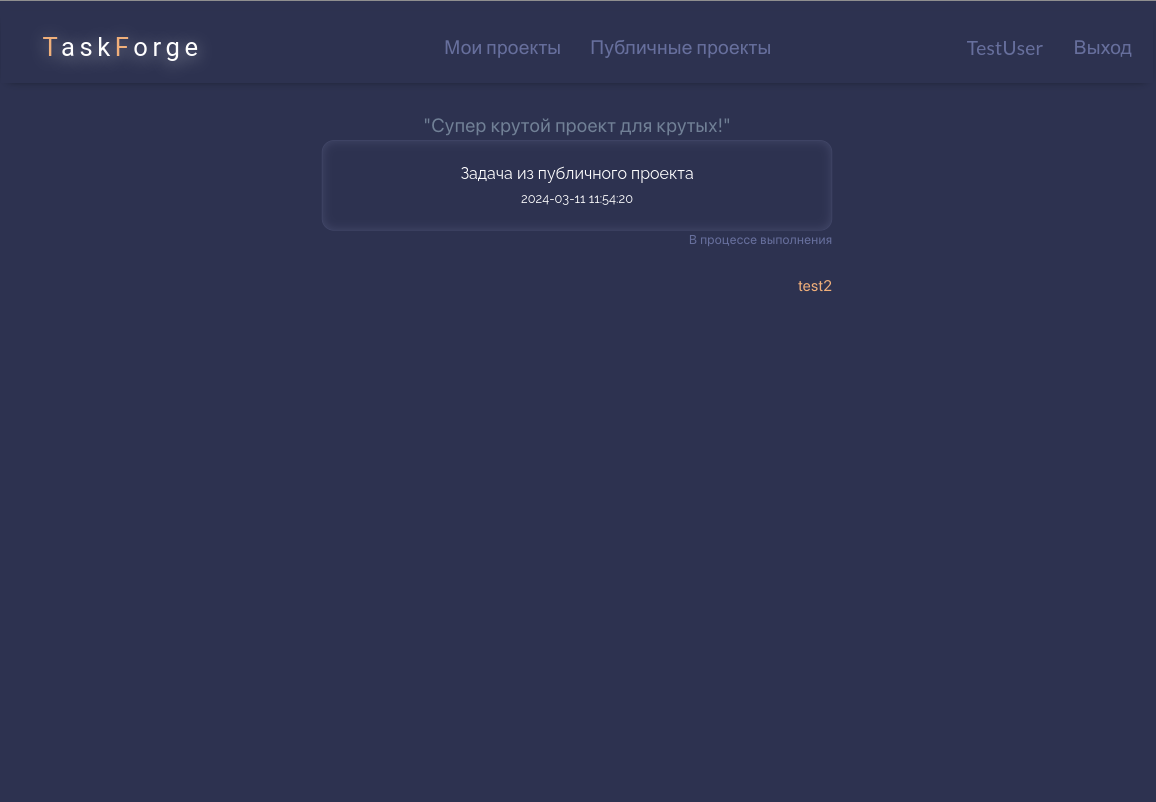


Рис. 6.12 – Задачи публичного проекта

После перехода в публичный проект пользователя test2 мы можем увидеть список его задач, а также их статус (Рис. 6.12)

# Список используемых источников

1. Джон Смит "PHP и его применение в веб-разработке", Феникс, 2023, 320 стр. (ориг. "PHP: The Complete Reference", McGraw-Hill)
2. Элизабет Джонсон "Разработка с использованием Laravel: Современные практики", Вильямс, 2022, 800 стр. (ориг. "Laravel: Up and Running", O'Reilly Media)
3. Майкл Томпсон "TypeScript: Полное руководство", Питер, 2021, 944 стр. (ориг. "TypeScript for Enterprise Development", Manning Publications)
4. Джеймс Браун "PostgreSQL: Руководство для разработчиков", Питер, 2023, 560 стр. (ориг. "PostgreSQL: Developer's Guide", Apress)
5. Карл Уоллс "Продвинутое использование TypeScript в веб-разработке", Вильямс, 2022, 720 стр. (ориг. "Advanced TypeScript Techniques", O'Reilly Media)
6. Стивен Миллер "Основы PHP и HTML: Создание динамических веб-сайтов", Питер, 2022, 400 стр. (ориг. "PHP and HTML Basics", Wiley)
7. Мария Гарсия "Разработка с использованием PostgreSQL и Laravel: Современные практики", Вильямс, 2021, 880 стр. (ориг. "PostgreSQL and Laravel Development: Best Practices", O'Reilly Media)
8. Анна Мартин "JavaScript: Современные методы и подходы", Питер, 2023, 1056 стр. (ориг. "JavaScript: Modern Methods and Approaches", Manning Publications)
9. Максим Ковальский "Разработка веб-приложений с использованием Laravel: Практическое руководство", Вильямс, 2021, 640 стр. (ориг. "Web Application Development with Laravel: Practical Guide", Apress)
10. Эмилия Джонс "Реактивное программирование с использованием TypeScript и Laravel", Питер, 2022, 480 стр. (ориг. "Reactive Programming with TypeScript and Laravel", Wiley)

# Приложение А – Листинг программного кода

package ru.michael.backend.entity;  
<?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;  
use Laravel\Sanctum\HasApiTokens;  
use Tymon\JWTAuth\Contracts\JWTSubject;  
  
class User extends Authenticatable implements JWTSubject  
{  
 use Notifiable;  
 /\*\*  
 \* The attributes that are mass assignable.  
 \*  
 \* @var array<int, string>  
 \*/  
 protected $fillable = [  
 'name',  
 'email',  
 'password',  
 ];  
  
 /\*\*  
 \* The attributes that should be hidden for serialization.  
 \*  
 \* @var array<int, string>  
 \*/  
 protected $hidden = [  
 'password',  
 'remember\_token',  
 ];  
  
 /\*\*  
 \* The attributes that should be cast.  
 \*  
 \* @var array<string, string>  
 \*/  
 protected $casts = [  
 'email\_verified\_at' => 'datetime',  
 'password' => 'hashed',  
 ];  
  
 public function getJWTIdentifier()  
 {  
 return $this->getKey();  
 // TODO: Implement getJWTIdentifier() method.  
 }  
  
 public function getJWTCustomClaims()  
 {  
 return [];  
 // TODO: Implement getJWTCustomClaims() method.  
 }  
}

<?php  
namespace App\Models;  
  
use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Model;  
  
class Task extends Model  
{  
 use HasFactory;  
  
 private int $project\_id;  
 private int $id;  
 private string $title;  
 private bool $done;  
  
 protected $fillable = ["title", "done", "project\_id", "expiry\_at"];  
}

<?php  
namespace App\Models;  
  
use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Model;  
  
class Project extends Model  
{  
 use HasFactory;  
// private string $name;  
// private string $description;  
// private \DateTime $expiryAt;  
// private int $ownerId;  
 protected $fillable = ["name", "description", "expiry\_at", "owner\_id", "is\_public"];  
}

<?php  
namespace App\Http\Controllers;  
  
use App\Http\Requests\StoreTaskRequest;  
use App\Http\Requests\UpdateTaskRequest;  
use App\Http\Resources\TaskResource;  
use App\Models\Project;  
use App\Models\Task;  
use App\Service\ArrayKeysTransformator;  
use App\Service\Query\ProjectTasksQuery;  
use App\Service\Responser;  
use Illuminate\Http\Request;  
use function Symfony\Component\Translation\t;  
  
class TaskController extends Controller  
{  
 /\*\*  
 \* Display a listing of the resource.  
 \*/  
 public function index(Project $project, Request $request)  
 {  
 $queryTransformer = new ProjectTasksQuery($project->id);  
 $queryParams = $queryTransformer->filter($request);  
// dd($queryParams);  
 return TaskResource::collection(Task::query()->where($queryParams)->paginate());  
// return TaskResource::collection(Task::query()->paginate());  
 }  
  
 /\*\*  
 \* Show the form for creating a new resource.  
 \*/  
 public function create()  
 {  
  
 }  
  
 /\*\*  
 \* Store a newly created resource in storage.  
 \*/  
 public function store(StoreTaskRequest $request): TaskResource  
 {  
  
 $taskJson = $request->json()->all();  
 $task = new Task(ArrayKeysTransformator::toWithUnderscores($taskJson));  
 $task->save();  
 return new TaskResource($task);  
 }  
  
 /\*\*  
 \* Display the specified resource.  
 \*/  
 public function show(Task $task)  
 {  
 //  
 }  
  
 /\*\*  
 \* Update the specified resource in storage.  
 \*/  
 public function update(UpdateTaskRequest $request, Task $task)  
 {  
 //  
 }  
  
 /\*\*  
 \* Remove the specified resource from storage.  
 \*/  
 public function destroy(Project $project, Task $task)  
 {  
// dd($project, $task);  
// $id = Task::query()->delete($task->id);  
// dd($id);  
 Task::destroy([$task->getAttribute("id")]);  
 return Responser::createSuccessResponse(data: [], message: "task deleted");  
 }  
  
 public function done(Project $project, Task $task)  
 {  
 $task->setAttribute("done", !$task->getAttribute("done"));  
 $task->save();  
 return new TaskResource($task);  
 }  
}

<?php  
namespace App\Http\Controllers;  
  
use App\Models\User;  
use App\Service\UserAuthenticator;  
use Illuminate\Contracts\Foundation\Application;  
use Illuminate\Contracts\Routing\ResponseFactory;  
use Illuminate\Http\JsonResponse;  
use Illuminate\Http\Request;  
use Illuminate\Http\Response;  
use Illuminate\Validation\ValidationException;  
use Illuminate\Validation\Validator;  
use function Laravel\Prompts\password;  
  
class RegistrationController extends Controller  
{  
 public function \_\_construct()  
 {  
  
 }  
  
 public function register(Request $request)  
 {  
 try {  
 $this->validate($request, [  
 "email" => ["required", "unique:users", "min:8"],  
 "password"=>["required", "min:8"]  
 ]  
 );  
 } catch (ValidationException $e) {  
// dd($e);  
 return \response($e->errors(), $e->status);  
 }  
  
 $user = new User(array\_merge(request()->json()->all(), ["name" => \request()->json()->get("username")]));  
 $user->save();  
 return response()->json([], "200");  
 }  
}

<?php  
namespace App\Http\Controllers;  
  
use App\Http\Requests\StoreProjectRequest;  
use App\Http\Requests\UpdateProjectRequest;  
use App\Http\Resources\ProjectResource;  
use App\Models\Project;  
use App\Service\ArrayKeysTransformator;  
use App\Service\Query\ProjectQuery;  
use Illuminate\Http\Request;  
use Illuminate\Support\Facades\Auth;  
use Illuminate\Support\Facades\Gate;  
  
class ProjectController extends Controller  
{  
 public function \_\_construct()  
 {  
 $this->middleware("auth:api")->only(["my"]);  
 $this->authorizeResource(Project::class, "project");  
 }  
  
 public function index(Request $request)  
 {  
 if (!in\_array($request->get("filter"), ["my", "public"])) {  
 return response(["message" => "You can't view all projects"], 400, ["Content-Type" => "application/json"]);  
 }  
  
 $projectsQuery = new ProjectQuery();  
 $projects = Project::query()->where($projectsQuery->filter($request))->paginate();  
 return ProjectResource::collection($projects);  
 }  
  
 public function store(StoreProjectRequest $request): ProjectResource  
 {  
 $data = ArrayKeysTransformator::toWithUnderscores($request->json()->all());  
 $project = new Project($data);  
 $project->setAttribute("owner\_id", Auth::getUser()->getAuthIdentifier());  
 $project->save();  
 return new ProjectResource($project);  
 }  
  
 public function show(Project $project): ProjectResource  
 {  
// $this->authorizeForUser(Auth::user(), "view", $project);  
 return new ProjectResource($project);  
 }  
  
 public function edit(Project $project)  
 {  
 dd($project);  
 }  
  
 public function update(UpdateProjectRequest $request, Project $project)  
 {  
  
// dump(array\_keys($project->getAttributes()));  
 Project::query()->where("id", "=", $project->id)->updateOrCreate(  
 array\_keys($project->getAttributes()),  
 array\_map(function ($attr) use ($project) {  
 return $project->getAttributes()[$attr];  
 }, $project->getAttributes()));  
 //  
 }  
// public function destroy(\Illuminate\Http\Request $request): void  
// {  
// dd($request);  
// $urlParts = explode("/", $request->url());  
// $id = $urlParts[count($urlParts) - 1];  
// Project::destroy($id);  
// Project::destroy($project);  
// }  
 public function destroy(Project $project)  
 {  
 $project->delete();  
 return new ProjectResource($project);  
 }  
}