**Техническое задание (ТЗ) на разработку веб-приложения**

**1. Общая информация**

**Название проекта**: Система управления проектами и задачами  
**Цель**: Разработать веб-приложение для управления проектами и задачами, включающее REST API и клиентскую часть.  
**Технологический стек**:

* Бэкенд: Laravel
* Фронтенд: Чистый JavaScript или фреймворк (например, Vue.js или React) с возможностью отправки AJAX-запросов через jQuery.

**2. Группы пользователей**

1. **Администратор**
   * Управление пользователями системы (создание, редактирование, удаление)
   * Управление проектами и задачами
   * Просмотр и экспорт отчётов
   * Управление ролями и правами доступа пользователей
2. **Руководитель проекта**
   * Создание и управление проектами
   * Назначение задач пользователям
   * Просмотр состояния проектов и задач
   * Просмотр и экспорт отчётов по проекту
3. **Исполнитель**
   * Просмотр и выполнение назначенных задач
   * Отметка о выполнении задач
   * Возможность добавления комментариев к задачам

**3. Основные сущности системы**

1. **Пользователь**
   * **Атрибуты**:
     + ID (int) - Уникальный идентификатор пользователя
     + Имя (string) - Имя пользователя
     + Email (string) - Email для связи и входа в систему
     + Роль (enum) - Администратор, Руководитель проекта, Исполнитель
     + Пароль (hashed string) - Хешированный пароль для входа
   * **Ограничения**:
     + Email должен быть уникальным и валидным
     + Пароль минимум 8 символов, должен содержать цифры и буквы
2. **Проект**
   * **Атрибуты**:
     + ID (int) - Уникальный идентификатор проекта
     + Название (string) - Название проекта
     + Описание (text) - Детальное описание проекта
     + Дата начала (date) - Дата начала проекта
     + Дата окончания (date) - Дата планируемого окончания проекта
     + Статус (enum) - Статусы проекта: "Создан", "В процессе", "Завершён"
     + Остаток дней (int) - Количество дней до завершения проекта
   * **Ограничения**:
     + Название проекта должно быть уникальным
     + Дата окончания не может быть раньше даты начала
3. **Задача**
   * **Атрибуты**:
     + ID (int) - Уникальный идентификатор задачи
     + Название (string) - Краткое описание задачи
     + Описание (text) - Детальное описание задачи
     + Проект (foreign key) - ID проекта, к которому относится задача
     + Исполнитель (foreign key) - ID пользователя, ответственного за выполнение задачи
     + Приоритет (enum) - "Низкий", "Средний", "Высокий" (цветовое отображение приоритета)
     + Дата начала (date) - Дата начала выполнения задачи
     + Дата окончания (date) - Планируемая дата завершения задачи
     + Статус (enum) - Статусы задачи: "Назначена", "Выполняется", "Завершена"
     + Остаток дней (int) - Количество дней до завершения задачи
   * **Ограничения**:
     + Дата окончания задачи не может быть раньше даты начала
     + Приоритет задачи указывается при создании
     + Статус задачи может изменять только руководитель проекта или администратор
4. **Отчёт**
   * **Атрибуты**:
     + ID (int) - Уникальный идентификатор отчёта
     + Проект (foreign key) - ID проекта, по которому создается отчёт
     + Дата создания (date) - Дата генерации отчёта
     + Ответственный (foreign key) - ID пользователя, создавшего отчёт
     + Статистика (JSON) - Структурированные данные по проекту (количество задач, статус задач и т.д.)
   * **Ограничения**:
     + Отчёты могут формироваться только по завершенным или активным проектам
     + Данные в отчёте должны быть актуальными на момент создания

**4. Основные функции и сценарии**

1. **Администратор**:
   * Управление пользователями: создание, редактирование и удаление пользователей.
   * Просмотр всех проектов и задач.
   * Назначение руководителей проектов и исполнителей.
   * Генерация отчетов по всей системе, экспорт данных в формате PDF или Excel.
2. **Руководитель проекта**:
   * Создание нового проекта: задание названия, описания, сроков и команды.
   * Управление задачами: создание, назначение исполнителей, изменение статусов.
   * Просмотр текущего статуса задач и проектов.
   * Фильтрация задач по приоритету.
   * Просмотр обратного отсчёта дней до завершения задач и проектов.
   * Пагинация задач при большом количестве данных.
   * Генерация отчёта по проекту (например, список задач с их статусом) и экспорт в формате PDF или Excel.
3. **Исполнитель**:
   * Просмотр списка задач, назначенных на него.
   * Выполнение и закрытие задач.
   * Комментирование задач.
   * Просмотр своей рабочей статистики.
   * Фильтрация задач по приоритету.
   * Просмотр обратного отсчёта дней до завершения задачи.
   * Пагинация для удобного просмотра списка задач.

**5. Ограничения на информацию**

1. **Пользователь**:
   * Email должен быть уникальным и проверяться на корректность.
   * Пароль должен быть хеширован (не хранить в открытом виде).
   * Входные данные должны валидироваться с помощью регулярных выражений и встроенных средств Laravel.
2. **Проект**:
   * Название проекта должно быть уникальным, длиной не менее 3 символов.
   * Даты должны проверяться на корректность и последовательность.
3. **Задача**:
   * Поля названия и описания задачи обязательны к заполнению.
   * Приоритет и статус задачи могут принимать только разрешенные значения.

**6. Выходные данные**

Приложение должно предоставлять возможность генерации отчётов:

* **Типы отчётов**:
  + Отчет по проекту: состояние задач (назначены, выполнены, в процессе).
  + Отчет по исполнителю: список задач, выполненных или находящихся в процессе выполнения.
  + Сводный отчёт по всей системе для администратора.
* **Форматы**:
  + PDF
  + Excel (XLSX)

**7. API и клиент**

 **REST API**:

* Реализация CRUD операций для всех основных сущностей: пользователей, проектов, задач, отчетов.
* Аутентификация через JWT токены.
* Эндпоинты:
  + /api/projects - работа с проектами
  + /api/tasks - работа с задачами
  + /api/users - работа с пользователями
  + /api/reports - генерация и получение отчетов

 **Клиент**:

* Взаимодействие с API для реализации всех функций приложения.
* Для администраторов: интерфейс управления пользователями, проектами и задачами.
* Для руководителей: создание и управление проектами, генерация отчётов.
* Для исполнителей: просмотр и выполнение задач, комментарии.
* Интуитивно понятный интерфейс с реакцией на действия пользователя (например, подтверждение выполнения задачи).

**7.1 Пример обработки запроса API (бэкенд)**

На сервере Laravel, REST API будет реализовано через роутинг и контроллеры. Пример получения списка всех задач по проекту с фильтрацией по приоритету и пагинацией:

**Запрос:**

bash

Копировать код

GET /api/projects/{project\_id}/tasks?priority=high&page=1

**Контроллер:**

php

Копировать код

public function getTasksByProject(Request $request, $project\_id) {

$priority = $request->get('priority');

$tasksQuery = Task::where('project\_id', $project\_id);

if ($priority) {

$tasksQuery->where('priority', $priority);

}

$tasks = $tasksQuery->paginate(10); // Пагинация по 10 задач на страницу

return response()->json($tasks);

}

**Ожидаемый ответ (JSON):**

json

Копировать код

{

"data": [

{

"id": 1,

"title": "Разработка макета",

"description": "Создать макет проекта",

"status": "В процессе",

"priority": "Высокий",

"assigned\_user": "Иван Иванов",

"days\_remaining": 5

},

{

"id": 2,

"title": "Написание API",

"description": "Создать REST API для системы",

"status": "Назначена",

"priority": "Высокий",

"assigned\_user": "Петр Петров",

"days\_remaining": 3

}

],

"links": {

"first": "http://example.com/api/projects/1/tasks?page=1",

"last": "http://example.com/api/projects/1/tasks?page=5",

"prev": null,

"next": "http://example.com/api/projects/1/tasks?page=2"

},

"meta": {

"current\_page": 1,

"from": 1,

"last\_page": 5,

"path": "http://example.com/api/projects/1/tasks",

"per\_page": 10,

"to": 10,

"total": 50

}

}

**7.2 Пример отправки запроса к API (клиентская часть)**

Используя jQuery, можно отправить AJAX-запрос к серверу для создания новой задачи с указанием приоритета.

**Пример кода:**

javascript

Копировать код

$(document).ready(function() {

$('#createTaskForm').on('submit', function(e) {

e.preventDefault();

const formData = {

title: $('#taskTitle').val(),

description: $('#taskDescription').val(),

project\_id: $('#projectId').val(),

assigned\_user\_id: $('#assignedUser').val(),

priority: $('#taskPriority').val(),

\_token: $('meta[name="csrf-token"]').attr('content') // Laravel CSRF токен

};

$.ajax({

url: '/api/tasks',

type: 'POST',

data: formData,

success: function(response) {

alert('Задача успешно создана!');

// Можно обновить список задач или перенаправить пользователя

},

error: function(error) {

alert('Ошибка при создании задачи');

}

});

});

});

**8. Тестирование и безопасность**

1. **Тестирование**:
   * Модульные и интеграционные тесты для API.
   * Тестирование клиентской части на совместимость с различными браузерами.
2. **Безопасность**:
   * Защита от SQL-инъекций и XSS-атак.
   * Использование CSRF-токенов для защиты от подделки запросов.
   * Шифрование паролей и безопасное хранение данных.

**9. Дизайн и UI/UX требования**

1. **Цветовое отображение приоритета задач**:
   * **Низкий приоритет**: Зелёный
   * **Средний приоритет**: Жёлтый
   * **Высокий приоритет**: Красный
2. **Общие требования**:
   * **Интуитивно понятный интерфейс**: Пользователи должны легко ориентироваться в системе. Важно минимизировать количество кликов для выполнения основной задачи.
   * **Универсальный дизайн**: Приложение должно корректно отображаться на различных устройствах (адаптивная вёрстка).
   * **Использование модальных окон**: Для создания/редактирования задач и проектов можно использовать модальные окна, чтобы не перезагружать страницу.
   * **Цветовая схема**: Нейтральные цвета для фона, акцентные — для важных элементов (кнопки, уведомления).
   * **Иконки**: Использование иконок для обозначения действий (например, значки для редактирования, удаления, завершения задач).
3. **Навигация**:
   * Удобное боковое меню для доступа к основным разделам: проекты, задачи, отчёты.
   * Быстрые ссылки для доступа к текущим задачам исполнителя и проектам руководителя.

**10. Сроки выполнения**

* Разработка приложения — 1 месяц.
* Тестирование — в течении выполнения проекта.