Техническое задание на разработку модульного веб-сайта

1. Введение

Цель проекта: разработать гибкое ядро CMS-платформы с модульной архитектурой, обеспечивающей возможность подключения/отключения и быстрого создания новых модулей без изменения ядра системы.

Язык программирования: PHP 8.x
Фреймворк: собственное ядро или на базе облегченного фреймворка (Laravel, Symfony)

Базовые модули для первой версии:

- Users (`/modules/users`) - управление пользователями и авторизацией
- News (`/modules/news`) - новостной функционал
- Comments (`/modules/comments`) - система комментариев

Предполагаемые модули для будущей разработки:

- Forum (`/modules/forum`) - дискуссионная площадка
- Gallery (`/modules/gallery`) - медиа-галерея
- Shop (`/modules/shop`) - интернет-магазин

2. Структура проекта

```
/                  # корневая папка
├── index.php      # точка входа приложения
├── core/          # ядро системы
│   ├── Router.php  # маршрутизатор
│   ├── Module.php  # базовый класс для модулей
│   ├── Database.php # подключение и работа с БД
│   ├── Auth.php    # аутентификация и авторизация
│   ├── Config.php  # менеджер конфигураций
│   ├── Event.php   # система событий
│   ├── Cache.php   # кэширование
│   └── Helper.php  # вспомогательные функции
├── bootstrap/     # загрузчик приложения
│   ├── app.php     # инициализация приложения
│   └── modules.php # сканирование и загрузка модулей
├── config/        # конфигурационные файлы системы
│   ├── app.php     # основная конфигурация
│   ├── database.php # настройки БД
│   └── modules.php # конфигурация модулей
├── admin/         # полный код и шаблоны админ-панели
│   ├── dashboard.php # главная панель управления
│   ├── modules/    # управление модулями
│   └── settings/   # общие настройки
├── modules/       # папка с модулями
│   ├── users/      # модуль пользователей
│   │   ├── controllers/
│   │   ├── models/
│   │   ├── views/
│   │   ├── routes.php
│   │   ├── module.json # мета-информация о модуле
│   │   ├── migrations/ # миграции модуля
│   │   └── assets/
│   ├── news/
│   │   ├── controllers/
│   │   ├── models/
│   │   ├── views/
│   │   ├── routes.php
│   │   ├── module.json
│   │   ├── migrations/
│   │   └── assets/
│   └── comments/
│       ├── controllers/
│       ├── models/
│       ├── views/
│       ├── routes.php
│       ├── module.json
│       ├── migrations/
```

```
│       └── assets/
└── themes/
    ├── public/     # темы для публичной части
    │   └── default/
    │       ├── header.php
    │       ├── footer.php
    │       ├── sidebar.php
    │       ├── menu.php
    │       ├── body.php
    │       └── assets/
    │           ├── css/
    │           ├── js/
    │           └── images/
    └── admin/      # темы для админской части
        └── default-admin/
            ├── header.php
            ├── footer.php
            ├── sidebar.php
            ├── menu.php
            ├── body.php
            └── assets/
                ├── css/
                ├── js/
                └── images/
```

3. Архитектура ядра системы

3.1 Компоненты ядра

- **Router** - маршрутизатор запросов, поддержка RESTful-паттернов
- **Module** - базовый класс для всех модулей, определяющий интерфейс взаимодействия
- **Database** - слой абстракции для работы с БД, поддержка миграций
- **Config** - управление конфигурациями (системными и модульными)
- **Event** - система событий и обработчиков для межмодульного взаимодействия
- **Auth** - единая система аутентификации и авторизации
- **Cache** - кэширование данных на разных уровнях
- **Template** - система шаблонизации с поддержкой тем

3.2 Интерфейс модулей

Каждый модуль должен реализовывать следующие методы:

- `install()` - установка модуля (создание таблиц, базовых данных)
- `uninstall()` - удаление модуля
- `enable()` - включение модуля
- `disable()` - отключение модуля
- `update()` - обновление модуля
- `getRoutes()` - предоставление маршрутов модуля
- `getAdminMenu()` - пункты меню для админ-панели
- `getWidgets()` - доступные виджеты модуля

3.3 Система событий

Для обеспечения взаимодействия между модулями без жесткой связности реализовать систему событий:

- Регистрация событий: `Event::register('event_name')`
- Подписка на события: `Event::subscribe('event_name', function($data) {...})`
- Вызов событий: `Event::trigger('event_name', $data)`

3.4 API для модулей

Ядро должно предоставлять унифицированный API для модулей:

- Доступ к функциям ядра через сервис-контейнер
- Доступ к конфигурации: `Config::get('module.setting')`
- Доступ к БД через unified interface: `DB::table('table')->select(...)`
- Доступ к моделям других модулей: `Module::get('users')->getModel('User')`

4. Модульность и управление

1. **Каждый модуль** располагается в `/modules/{название}` и включает:
   - `module.json` – метаданные модуля (название, версия, автор, зависимости)
   - `routes.php` – маршруты модуля
   - `controllers/` – контроллеры
   - `models/` – модели данных
   - `views/` – шаблоны вывода
   - `migrations/` – миграции для БД модуля
   - `assets/` – собственные CSS, JS, изображения
2. **Установка и управление**:
   - Админ-панель → «Управление модулями»
     - Список доступных (установленных и новых) модулей
     - Действия: Установить / Удалить / Включить / Отключить
     - Ссылки на настройки модулей (если нужны)
     - Проверка зависимостей между модулями
     - Автоматическое определение новых модулей в директории
3. **Загрузка модулей**:
   - Автоматическое сканирование директории `/modules`
   - Загрузка только активных модулей
   - Проверка целостности и совместимости модулей
   - Приоритетная загрузка зависимостей

5. Базовые модули

5.1 Модуль "Users"

- Регистрация и авторизация пользователей
- Управление профилями
- Ролевая модель доступа (RBAC)
- Подтверждение email, восстановление пароля
- Интеграция с социальными сетями (опционально)

5.2 Модуль "News"

- CRUD для новостей (создание, редактирование, удаление, просмотр)
- Категории и теги
- Публичная лента с пагинацией
- Виджеты «Последние» и «Популярные»

5.3 Модуль "Comments"

- Привязка к любым сущностям (новости, статьи, товары)
- Модерация: одобрение, удаление
- Древовидная структура
- CAPTCHA или авторизация для защиты от спама

6. Административная панель

- **Доступ**: роль «администратор» (RBAC)
- **Меню**: динамическое формирование по зарегистрированным модулям
- **Страница «Модули»**:

Название	Версия	Статус	Зависимости	Действия
Users	1.0.0	Включен	-	Отключить / Настройки
News	1.0.0	Включен	Users	Отключить / Настройки
Comments	0.9.5	Выключен	Users	Включить / Удалить

- **Настройки ядра**: конфигурация системы, логирование, кэширование
- **Настройки тем**: выбор публичной и админ-тем

7. Безопасность и производительность

- **Безопасность**:
  - CSRF-защита для всех форм
  - XSS-фильтрация ввода
  - SQL-инъекции: подготовленные запросы
  - Хеширование паролей (bcrypt/Argon2)
  - Защита от брутфорса (rate limiting)
  - Валидация данных на уровне ядра
- **Производительность**:
  - Кэширование на нескольких уровнях
  - Минимизация SQL-запросов
  - Ленивая загрузка модулей
  - Оптимизация статических ресурсов (CSS/JS)
  - Время ответа API ≤ 300 мс
- **Логирование**:
  - Ошибки выполнения
  - Действия пользователей
  - Действия администраторов
  - Журнал изменений модулей

8. Технологический стек

- **Backend**: PHP 8.x
- **База данных**: MySQL 8 / PostgreSQL
- **Шаблонизатор**: Blade / Twig / собственный
- **Frontend**: HTML5, CSS3, JavaScript (Vanilla/jQuery)
- **VCS**: Git (GitHub/GitLab)
- **CI/CD**: GitHub Actions / GitLab CI

9. План работ и сроки

Этап	Описание
1. Подготовка среды	Репозиторий, CI, скелет приложения
2. Разработка ядра	Основные компоненты, загрузчик, API
3. Модуль «Users»	Аутентификация, профили, права доступа
4. Система управления модулями	Установка, удаление, включение модулей
5. Модуль «News»	CRUD, публичный вывод, категоризация
6. Модуль «Comments»	Интеграция с другими модулями, дерево
7. Админ-панель	Управление системой и модулями
8. Тестирование и документация	Тесты, руководства, документация API
Итого	

10. Итоговые материалы

1. Исходный код на Git-репозитории
2. Документация по ядру системы
3. Руководство по разработке модулей
4. Руководство администратора
5. API-документация для интеграции с внешними системами
6. Набор тестов (unit + интеграционные)