### 

### **Введение**

### **Назначение**

Сайт представляет из себя площадку(маркетплейс) для покупки\продажи прав на контент (товарный знак, фото, видео, аудио). Контент для продажи на Сайте размещают владельцы прав на него в строгом соответствии с законодательством РФ.

* 1. **Базовый шаблон**

Разработка сайта выполнена на основе программного продукта Multipurpose Admin/Dashboard template, который представляет собой универсальный шаблон для создания административных панелей и дашбордов. Исходный код расположен по адресу <https://github.com/RGBvision/dashboard-boilerplate>

* 1. **Технологический стек**

**Backend**: PHP (чистый, без фреймворков).  
 **Frontend:** Vanilla JS, Chart.js, Bootstrap (стили).  
 **База данных:** MySQL.  
 **Веб-сервер:** Apache/Nginx.

1. **Общая архитектура**

**2.1. Фронтенд-архитектура**

* Bootstrap 5.1.1 как базовый фреймворк
* jQuery 3.6.0 для работы с DOM
* Адаптивный дизайн с использованием медиа-запросов

**2.2. Бэкенд-архитектура**

**Модель MVC**

* Model - работа с данными
* View - шаблоны SMARTY для визуализации
* Controller - обработка запросов и маршрутизация

**2.3. Структура**

| . ├── app/ # Основное приложение │ ├── api/ # API-эндпоинты │ │ └── API.php │ ├── classes/ # Бизнес-логика (классы TM4RENT) │ │ └── TM4RENT/ │ │ ├── Brand.php │ │ ├── Brands.php │ │ ├── Content.php │ │ └── ... (11 других классов) │ ├── i18n/ # Локализация API │ │ ├── api.en.lang.json │ │ └── api.ru.lang.json │ ├── js/ # Общие скрипты │ │ └── common.js │ ├── modules/ # Модули (MVC-структура) │ │ ├── brands/ # Пример модуля │ │ │ ├── controller/ │ │ │ ├── i18n/ │ │ │ ├── js/ │ │ │ ├── model/ │ │ │ ├── view/ │ │ │ └── Module.php │ │ └── ... (14 других модулей) │ ├── templates/ # Шаблоны Smarty │ │ └── default/ │ │ ├── header.tpl │ │ ├── footer.tpl │ │ └── ... (10 других) │ └── .htaccess │ ├── assets/ # Статические файлы │ ├── core/ # CSS/JS ядра │ ├── vendors/ # Сторонние библиотеки │ │ ├── bootstrap-datepicker/ │ │ ├── datatables.net/ │ │ └── ... (20+ других) │ └── i18n/ # Локализация фронта │ ├── en.js │ └── ru.js │ ├── configs/ # Конфигурации │ ├── db.config.php │ └── environment.php │ ├── system/ # Системное ядро │ ├── core/ # Базовые классы (Router, Auth и др.) │ │ ├── ApiRouter.php │ │ ├── Controller.php │ │ └── ... (14 других) │ ├── drivers/ # Драйверы БД, шаблонов и сессий │ └── common/ # Утилиты (Curl, Valid, Secure и др.) │ ├── uploads/ # Загружаемые файлы ├── tmp/ # Временные файлы │ ├── .envrc # Настройки окружения ├── index.php # Точка входа └── README.md # Документация |
| --- |

### **Ключевые особенности структуры:**

1. **Четкое разделение MVC**:
   * Каждый модуль в **app/modules/** содержит **controller/**, **model/**, **view/**.
   * Локализация хранится в **i18n/** для каждого модуля.
2. **Ядро системы**:
   * **system/core/** — роутинг, авторизация, DI-контейнер (Dependencies.php).
   * **system/drivers/** — абстракции для работы с БД, шаблонами (Smarty).
3. **Поддержка мультиязычности**:
   * JSON/INI-файлы локализации в **app/i18n/**, **assets/i18n/**.
4. **Бизнес-логика**:
   * Классы в **app/classes/TM4RENT/** (например, Brand.php, User.php).
5. **Внешние зависимости**:
   * Библиотеки в **assets/vendors/** (Bootstrap, TinyMCE, Dropzone и др.).

## **3. Детальное описание разделов**

### **3.1. app/**

**Назначение**: Основная бизнес-логика и модули приложения.

#### **Ключевые поддиректории::**

* **app/api/**
  + Обработка API-запросов (REST/GraphQL).
  + Ключевой файл: API.php — базовый контроллер для эндпоинтов.
* **app/classes/**
  + Ядро бизнес-логики (например, TM4RENT/Brand.php, TM4RENT/User.php).
* **app/modules/**
  + MVC-модули (каждый содержит controller/, model/, view/).
  + Пример: brands/ — управление брендами.
* **app/templates/**
  + Шаблоны Smarty (например, default/header.tpl).

### **3.1.1. app/api/** **Ключевые файлы**:

* **app/api/API.php**
  + Базовый контроллер для всех API-методов.
  + Авторизация через JWT/OAuth.
  + Пример метода:
  + php
  + Copy

public function getUsers(Request $request) {

return Response::json(User::all());

* + }

### **3.1.2. app/classes/**

**Назначение**: Ядро бизнес-логики.  
**Подпапки**:

* **app/classes/TM4RENT/** (основные сущности):
  + Brand.php, User.php, Offer.php и др.
  + Пример класса:
  + php
  + Copy

class User {

public static function getByEmail($email) {

return DB::query("SELECT \* FROM users WHERE email = ?", [$email]);

}

* + }

### **3.1.3. app/i18n/**

**Назначение**: Локализация API-ответов.  
**Файлы**:

* **app/i18n/api.en.lang.json** — английские переводы.
* **app/i18n/api.ru.lang.json** — русские переводы.  
  **Формат**:

json

Copy

{

"user.not\_found": "User not found"

}

### **3.1.4. app/js/**

**Назначение**: Клиентские JavaScript-файлы.  
**Ключевые файлы**:

* **app/js/common.js**
  + Глобальные функции (например, AJAX-запросы к API).
  + Пример:
  + javascript
  + Copy

function fetchUsers() {

return api.get('/users');

* + }

### **3.1.5. app/modules/ (главная часть приложения)**

**Назначение**: Модульная MVC-структура. Каждый модуль отвечает за отдельный функционал (например, управление пользователями).

#### **Структура модуля (на примере brands/):**

text

Copy

brands/

├── controller/ # Логика обработки запросов

│ └── Controller.php

├── i18n/ # Локализация модуля

│ ├── en.ini

│ └── ru.ini

├── js/ # Специфичные скрипты

│ └── brands.js

├── model/ # Работа с данными

│ └── Model.php

├── view/ # Шаблоны

│ ├── index.tpl

│ └── modal.tpl

└── Module.php # Конфигурация модуля

**Примеры модулей**:

* **app/modules/users/** — управление пользователями.
* **app/modules/catalog/** — каталог товаров.
* **app/modules/dashboard/** — админ-панель.

### **3.1.6. app/templates/**

**Назначение**: Шаблоны Smarty для рендеринга HTML.  
**Структура**:

* **default/** — базовая тема.
  + **header.tpl**, **footer.tpl** — общие части.
  + **index.tpl** — главная страница.  
    **Пример шаблона**:

smarty

Copy

{include file='header.tpl'}

<h1>{$title}</h1>

{include file='footer.tpl'}

### **3.1.7. app/.htaccess**

**Назначение**: Перенаправление запросов на index.php (ЧПУ).  
**Содержимое**:

apache

Copy

RewriteEngine On

RewriteRule ^(.\*)$ index.php?url=$1 [QSA,L]

## **3.1.8. Интеграция с системой**

1. **Автозагрузка классов**:
   * Классы из **app/classes/** загружаются через **system/loader/Loader.php**.
2. **Маршрутизация**:
   * Запросы в **api/** обрабатываются **app/api/API.php**.
   * Остальные — модулями из **app/modules/\*/controller/**.
3. **Локализация**:
   * Файлы из **app/i18n/** подключаются через **system/drivers/language/i18n.php**.

\_\_\_\_\_\_-\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_—--------------\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### **3.2. system/**

**Назначение**: Системное ядро и утилиты.

#### **Ключевые поддиректории:**

* **system/core/**
  + Базовые классы: **Router.php, Auth.php, Controller.php**.
* **system/common/**
  + Утилиты: **Curl.php, Valid.php, Secure.php** (шифрование).
* **system/drivers/**
  + Драйверы для БД, сессий, шаблонизаторов.
  + Пример: db/DB.php — абстракция для работы с MySQL.
* **system/loader/**
  + Автозагрузка классов (Loader.php).

### **3.2.1 system/core/**

В каталоге **system/core/** находятся ключевые классы ядра системы.

Детальное описание файлов раздела:

### **3.2.1.1. system/core/ApiRouter.php**

**Назначение**: Маршрутизация API-запросов.  
**Что делает**:

* Обрабатывает HTTP-запросы к API (например, GET **api/users**).
* Определяет, какой контроллер и метод вызывать на основе URL.  
  **Особенности**:
* Поддержка RESTful-маршрутов.
* Интеграция с аутентификацией (**Auth.php**).

### **3.2.1.2. system/core/Auth.php**

**Назначение**: Аутентификация и авторизация пользователей.  
**Методы**:

* login($email, $password) — проверка учетных данных.
* checkPermission($role) — проверка прав доступа.  
  **Интеграция**:
* Работает с **system/drivers/sessions/** для хранения сессий.
* Использует User.php для управления данными пользователя.

### **3.2.1.3. system/core/Controller.php**

**Назначение**: Базовый класс для всех контроллеров.  
**Ключевые методы**:

* render($view, $data) — отрисовка шаблонов (интеграция с Smarty).
* jsonResponse($data) — отправка JSON-ответов.  
  **Зависимости**:
* Наследуется всеми модульными контроллерами (например, **app/modules/brands/controller/Controller.php**).

### **3.2.1.4. system/core/Dependencies.php**

**Назначение**: DI-контейнер (управление зависимостями).  
**Функционал**:

* Регистрирует сервисы (БД, почта, кеш).
* Автоматически внедряет зависимости в классы.

**Пример**:

| $di->register('db', new Database(/\* config \*/)); |
| --- |

### **3.2.1.5. system/core/Injections.php**

**Назначение**: Внедрение зависимостей через аннотации.  
**Как работает**:

* Анализирует doc-блоки классов (например, @inject db).
* Автоматически передает экземпляры сервисов.

### **3.2.1.6. system/core/Log.php**

**Назначение**: Логирование событий.  
**Уровни логирования**:

* error, warning, info.  
  **Куда пишет**:
* Файлы (например, logs/app.log).
* Опционально: базу данных (system/drivers/db/DB.php).

### **3.2.1.7. system/core/Model.php**

**Назначение**: Базовый класс для моделей данных.  
**Методы**:

* save() — сохранение в БД.
* find($id) — поиск по ID.  
  **Интеграция**:
* Использует system/drivers/db/DB.php для запросов.

### **3.2.1.8. system/core/Module.php**

**Назначение**: Базовый класс для модулей.  
**Что делает**:

* Регистрирует пути и настройки модуля.
* Инициализирует контроллеры и модели.

### **3.2.1.9. system/core/Navigation.php**

**Назначение**: Управление меню и навигацией.  
**Пример**:

| $nav->addItem('Dashboard', '/dashboard', 'fa-home'); |
| --- |

### **3.2.1.10. system/core/Notification.php**

**Назначение**: Система уведомлений.  
**Поддержка**:

* Email (через libraries/mailer).
* WebSocket (через libraries/WebSocket).

### **3.2.1.11. system/core/Permissions.php**

**Назначение**: Управление правами доступа (RBAC).  
**Роли**:

* admin, manager, user.  
  **Методы**:
* can($action, $resource) — проверка разрешений.

### **3.2.1.12. system/core/Router.php**

**Назначение**: Маршрутизация HTTP-запросов (не API).  
**Особенности**:

* Поддержка ЧПУ (например, /products/:id).
* Интеграция с Controller.php.

### **3.2.1.13. system/core/Search.php**

**Назначение**: Полнотекстовый поиск.  
**Индексация**:

* Использует Elasticsearch или MySQL FULLTEXT.

### **3.2.1.14. system/core/Settings.php**

**Назначение**: Управление настройками системы.  
**Хранение**:

* База данных (settings table).
* Кеширование в Redis/Memcached.

### **3.2.1.15. system/core/User.php**

**Назначение**: Работа с данными пользователя.  
**Методы**:

* getByEmail($email) — поиск по email.
* updateProfile($data) — обновление профиля.

### **3.2.1.16. system/core/UserRoles.php**

**Назначение**: Связь пользователей с ролями.  
**Таблицы**:

* users\_roles (many-to-many).

### **Общие наблюдения:**

1. **Паттерны**:
   * MVC (модули в app/ наследуют Controller.php/Model.php).
   * DI (внедрение через Dependencies.php).

### **3.2.2 system/common/**

В каталоге system/common/ находятся вспомогательные классы-утилиты, которые обеспечивают базовую функциональность для работы системы.

Детальное описание файлов раздела:

### **3.2.2.1. system/common/Arrays.php**

**Назначение**: Работа с массивами.  
**Ключевые методы**:

* get($array, $key, $default) — безопасное получение значения из массива.
* flatten($array) — преобразование многомерного массива в одномерный.  
  **Пример**:

php

Copy

$data = ['user' => ['name' => 'John']];

Arrays::get($data, 'user.name'); // 'John'

### **3.2.2.2. system/common/Cast.php**

**Назначение**: Приведение типов данных.  
**Методы**:

* toInt($value), toFloat($value), toBool($value) — конвертация в строгие типы.
* toString($value) — безопасное преобразование в строку.

### **3.2.2.3. system/common/Cookie.php**

**Назначение**: Управление HTTP-куками.  
**Функционал**:

* set($name, $value, $expiry) — установка куки.
* delete($name) — удаление.  
  **Безопасность**:
* Автоматическая установка флага HttpOnly.

### **3.2.2.4. system/common/Countries.php**

**Назначение**: Работа со странами и регионами.  
**Данные**:

* Список стран в формате ['code' => 'Name'].  
  **Методы**:
* getAll() — возвращает все страны.
* getByCode($code) — поиск по коду (например, RU).

### **3.2.2.5. system/common/Curl.php**

**Назначение**: Отправка HTTP-запросов через cURL.  
**Пример**:

php

Copy

$curl = new Curl();

$response = $curl->get('https://api.example.com/data');

**Поддержка**:

* GET/POST/PUT/DELETE-запросов.
* Автоматическая обработка JSON-ответов.

### **3.2.2.6. system/common/Debug.php**

**Назначение**: Отладка кода.  
**Методы**:

* log($message) — запись в лог.
* dump($var) — вывод переменной с подсветкой (аналог var\_dump).

### **3.2.2.7. system/common/Dir.php**

**Назначение**: Работа с файловой системой (папками).  
**Методы**:

* create($path) — создание папки.
* isEmpty($path) — проверка на пустоту.

### **3.2.2.8. system/common/File.php**

**Назначение**: Управление файлами.  
**Функционал**:

* read($path), write($path, $content) — чтение/запись.
* safeName($filename) — генерация безопасного имени файла.

### **3.2.2.9. system/common/Hooks.php**

**Назначение**: Система хуков (событий).  
**Пример**:

php

Copy

Hooks::add('user.created', function($user) {

Log::info("User created: {$user->email}");

});

### **3.2.2.10. system/common/Html.php**

**Назначение**: Генерация HTML-кода.  
**Методы**:

* link($url, $text) — создание ссылки.
* escape($string) — экранирование спецсимволов.

### **3.2.2.11. system/common/IP.php**

**Назначение**: Работа с IP-адресами.  
**Методы**:

* getClientIp() — определение IP пользователя.
* isLocal($ip) — проверка, является ли IP локальным.

### **3.2.2.12. system/common/Json.php**

**Назначение**: Работа с JSON.  
**Методы**:

* encode($data), decode($json) — кодирование/декодирование.
* validate($json) — проверка валидности JSON.

### **3.2.2.13. system/common/Mailer.php**

**Назначение**: Отправка email.  
**Интеграция**:

* Использует libraries/SwiftMailer.  
  **Пример**:

php

Copy

Mailer::send('user@example.com', 'Subject', 'Body');

### **3.2.2.14. system/common/Netcom.php**

**Назначение**: Сетевая коммуникация (TCP/UDP).  
**Использование**:

* Для интеграции с внешними API или сокетами.

### **3.2.2.15. system/common/Number.php**

**Назначение**: Форматирование чисел.  
**Методы**:

* format($number, $decimals) — округление.
* toWords($number) — прописью (например, «42» → «сорок два»).

### **3.2.2.16. system/common/Request.php**

**Назначение**: Обработка HTTP-запросов.  
**Данные**:

* get(), post(), headers() — доступ к параметрам.  
  **Безопасность**:
* Фильтрация XSS-атак.

### **3.2.2.17. system/common/Response.php**

**Назначение**: Управление HTTP-ответами.  
**Методы**:

* json($data) — отправка JSON.
* redirect($url) — перенаправление.

### **3.2.2.18. system/common/Secure.php**

**Назначение**: Защита данных.  
**Функционал**:

* encrypt($data), decrypt($data) — AES-256.
* hashPassword($password) — bcrypt-хэширование.

### **3.2.2.19. system/common/SystemLocale.php**

**Назначение**: Локализация системы.  
**Поддержка**:

* Мультиязычность (переключение между en/ru).

### **3.2.2.20. system/common/UUID.php**

**Назначение**: Генерация UUID.  
**Пример**:

php

Copy

$uuid = UUID::v4(); // 'f47ac10b-58cc-4372-a567-0e02b2c3d479'

### **3.2.2.21. system/common/Valid.php**

**Назначение**: Валидация данных.  
**Методы**:

* email($string), url($string) — проверка формата.
* length($string, $min, $max) — проверка длины.

### **Ключевые особенности:**

1. **Универсальность**: Классы используются во всех модулях системы (например, Valid — в формах, Curl — для API).
2. **Безопасность**:
   * Secure.php и Request.php содержат критически важные механизмы защиты.
3. **Интеграция**:
   * Mailer.php и Netcom.php взаимодействуют с внешними сервисами.

### **3.2.3. system/functions/**

В папке system/functions/ находятся глобальные функции-хелперы, которые используются во всей системе для упрощения типовых операций. Вот детальное описание файлов:

### **3.2.3.1. common.php**

**Назначение**: Основные вспомогательные функции.  
**Ключевые функции**:

#### **3.2.3.1.1. e($string)**

* **Описание**: Экранирование HTML-сущностей (аналог htmlspecialchars).
* **Пример**:
* php
* Copy
* echo e('<script>alert(1)</script>'); // Выведет: &lt;script&gt;alert(1)&lt;/script&gt;

#### **3.2.3.1.2. dd($var)**

* **Описание**: Дамп переменной с завершением скрипта (аналог Laravel-функции).
* **Пример**:
* php
* Copy
* dd($user); // Выведет содержимое $user и остановит выполнение.

#### **3.2.3.1.3. config($key, $default = null)**

* **Описание**: Получение значения из конфигурации (например, config('app.name')).
* **Интеграция**: Читает данные из файлов в папке configs/.

#### **3.2.3.1.4. asset($path)**

* **Описание**: Генерация пути к статическому файлу (CSS/JS).
* **Пример**:
* php
* Copy
* echo asset('css/app.css'); // Выведет: /assets/css/app.css

#### **3.2.3.1.5. route($name, $params = [])**

* **Описание**: Генерация URL по имени маршрута.
* **Пример**:
* php
* Copy
* echo route('user.profile', ['id' => 42]); // /user/42/profile

#### **3.2.3.1.6. old($key, $default = null)**

* **Описание**: Получение старого ввода формы после валидации (для повторного заполнения полей).
* **Использование**:
* html
* Copy
* <input value="<?= old('username') ?>">
* Run HTML

#### **3.2.3.1.7. csrf\_token()**

* **Описание**: Генерация CSRF-токена для защиты форм.
* **Интеграция**: Использует system/common/Secure.php.

### **Как используются эти функции?**

1. **Во всех слоях приложения**:
   * В контроллерах (app/modules/\*/controller/).
   * В шаблонах Smarty (через {php} или модификаторы).
2. **Для безопасности**:
   * e() и csrf\_token() защищают от XSS и CSRF.
3. **Для удобства разработки**:
   * dd() упрощает отладку, asset() — управление статикой.

### **Пример подключения**

Функции автоматически подключаются в system/init.php или через:

php

Copy

require\_once \_\_DIR\_\_ . '/functions/common.php';

#### **3.2.4 system/drivers/**

В каталоге system/drivers/ находятся Драйверы

Детальное описание файлов раздела:

## **3.2.4.1. system/drivers/db/**

Драйверы для работы с базами данных.

### **3.2.4.1.1. system/drivers/db/DB.php**

**Назначение**: Абстракция для работы с СУБД.  
**Поддерживаемые СУБД**: MySQL, PostgreSQL.  
**Ключевые методы**:

* query($sql, $params) — выполнение запроса с подготовкой параметров.
* fetchAll() — получение всех строк результата.
* lastInsertId() — ID последней вставленной записи.  
  **Пример**:

php

Copy

$db->query("SELECT \* FROM users WHERE id = ?", [42]);

## **3.2.4.2. system/drivers/language/**

Драйверы для локализации.

### **3.2.4.2.1. i18n.php**

**Назначение**: Управление мультиязычностью.  
**Функционал**:

* Загрузка языковых файлов из system/i18n/ и app/i18n/.
* Параллельное хранение переводов в JSON и INI.  
  **Методы**:
* t($key, $lang) — получение перевода (например, t('welcome', 'ru')).
* setLanguage($lang) — смена языка.

## **3.2.4.3. system/drivers/sessions/**

Драйверы для работы с сессиями.

### **system/drivers/sessions/Session.php (Абстрактный класс)**

**Назначение**: Базовый интерфейс для всех драйверов сессий.  
**Методы**:

* start() — инициализация сессии.
* get($key) — чтение данных.

### **Конкретные реализации:**

#### **system/drivers/sessions/db.php**

**Хранение**: Сессии в базе данных (таблица sessions).  
**Использование**:

php

Copy

new SessionDB($db\_connection);

#### **system/drivers/sessions/files.php**

**Хранение**: Сессии в файлах (tmp/sessions/).

#### **system/drivers/sessions/native.php**

**Хранение**: Стандартный механизм PHP ($\_SESSION).

## **3.2.4.4. system/drivers/template/**

Драйверы шаблонизаторов.

### **system/drivers/template/Template.php (Абстрактный класс)**

**Назначение**: Интерфейс для рендеринга шаблонов.

**Методы**:

* render($view, $data) — отрисовка шаблона.

## **3.2.4.5 Ключевые особенности:**

1. **Гибкость**:
   * Легко заменить СУБД или шаблонизатор, не меняя код ядра.
2. **Безопасность**:
   * Все драйверы БД используют подготовленные запросы.
   * Сессии защищены от hijacking (реализация в SessionDB).
3. **Производительность**:
   * Кеширование языковых файлов в i18n.php.
   * Поддержка OPcache для шаблонов Smarty.

## **3.2.4. system/loader/**

Папка **system/loader/** отвечает за **автозагрузку классов (autoloading) и динамическую загрузку компонентов системы**. Это критически важная часть архитектуры, которая обеспечивает:

* **Автозагрузка классов** (PSR-4 или кастомная реализация) — чтобы не подключать файлы вручную через require.
* **Инициализация зависимостей** — загрузка сервисов, драйверов, библиотек при старте приложения.
* **Управление пространствами имен** — маппинг классов на файловую систему.

#### **system/loader/Loader.php (главный класс)**

**3.2.4.1. Что делает**:

1. **Регистрирует автозагрузчик** через spl\_autoload\_register().
2. **Преобразует имена классов в пути к файлам**:
3. php
4. Copy
5. // Пример: класс `System\Core\Router` -> `/system/core/Router.php`
6. **Поддерживает псевдонимы (aliases)** для удобного доступа к классам:
7. php
8. Copy

$loader->setAlias('Router', 'System\Core\Router');

1. // Теперь можно использовать `new Router()` вместо `new System\Core\Router()`.
2. **Загружает конфиги и зависимости** из system/config.php или configs/.

**Пример использования**:

php

Copy

$loader = new Loader();

$loader->register(); // Активирует автозагрузку

$loader->addNamespace('App', \_\_DIR\_\_ . '/app'); // Добавляет пространство имен

### **3.2.4.2. Интеграция с другими компонентами**

* **С system/init.php**:
  + Loader вызывается при старте приложения для подключения всех зависимостей.
* **С system/core/Dependencies.php**:
  + DI-контейнер использует Loader для резолва имен классов.

### **3.2.4.3. Особенности реализации**

* **Гибкость**: Поддержка кастомных путей (например, для модулей в /app/modules/\*).
* **Оптимизация**: Кеширование путей к классам для скорости (если реализовано).
* **Безопасность**: Проверка существования файлов перед подключением.

### **3.2.4.4. Как это работает на практике?**

1. При обращении к несуществующему классу:
2. php
3. Copy
4. new App\Modules\Users\Controller();
5. Loader:
   * Преобразует имя в путь: app/modules/users/Controller.php.
   * Подключает файл, если он существует.
   * Выбрасывает исключение, если класс не найден.

### **3.3. configs/**

**Назначение**: Конфигурация приложения.

* **configs/db.config.php** — настройки подключения к БД.
* **configs/environment.php** — переменные окружения (dev/prod).

### **3.4. assets/**

**Назначение**: Статические ресурсы.

* **assets/vendors/** — сторонние библиотеки (Bootstrap, TinyMCE).
* **assets/i18n/** — локализация фронтенда (en.js, ru.js).

Краткое описание каждой сторонней зависимости из папки **assets/vendors/** с указанием их назначения в проекте:

### **3.4.1. assets/vendors/ace-builds/**

**Назначение**: Редактор кода с подсветкой синтаксиса.  
**Использование**: Встроенный редактор для админ-панели (например, для редактирования шаблонов или CSS/JS).

### **3.4.2. assets/vendors/apexcharts/**

**Назначение**: Библиотека для построения интерактивных графиков.  
**Использование**: Дашборды, статистика (продажи, активность пользователей).

### **3.4.3. assets/vendors/bootstrap-colorpicker/**

**Назначение**: Плагин выбора цвета.  
**Использование**: Формы настройки дизайна (например, выбор цвета темы).

### **3.4.4. assets/vendors/bootstrap-datepicker/**

**Назначение**: Календарь для выбора дат.  
**Использование**: Формы бронирования, фильтры по дате.

### **3.4.5. assets/vendors/bootstrap-maxlength/**

**Назначение**: Ограничение длины ввода в текстовых полях.  
**Использование**: Формы с ограничениями (например, комментарии до 500 символов).

### **3.4.6. assets/vendors/bootstrap-slider/**

**Назначение**: Слайдер для выбора значений.  
**Использование**: Фильтры цен, настройки диапазонов.

### **3.4.7. assets/vendors/cades/**

**Назначение**: Криптографические функции (ЭЦП, шифрование).  
**Использование**: Работа с электронной подписью (документооборот).

### **3.4.8. assets/vendors/clipboard/**

**Назначение**: Копирование текста в буфер обмена.  
**Использование**: Кнопки "Скопировать ссылку" или код.

### **3.4.9. assets/vendors/cropperjs/**

**Назначение**: Обрезка и редактирование изображений.  
**Использование**: Загрузка аватаров, редактирование фото.

### **3.4.10. assets/vendors/datatables.net/**

**Назначение**: Интерактивные таблицы с сортировкой и пагинацией.  
**Использование**: Таблицы пользователей, заказов в админке.

### **3.4.11. assets/vendors/dropzone/**

**Назначение**: Загрузка файлов через drag-and-drop.  
**Использование**: Формы загрузки изображений или документов.

### **3.4.12. assets/vendors/flag-icon-css/**

**Назначение**: Флаги стран в виде CSS-иконок.  
**Использование**: Выбор языка/региона.

### **3.4.13. assets/vendors/flatpickr/**

**Назначение**: Альтернативный календарь для выбора дат и времени.  
**Использование**: Формы с выбором времени (например, бронирование).

### **3.4.14. assets/vendors/fullcalendar/**

**Назначение**: Календарь событий.  
**Использование**: Система бронирования, планировщик задач.

### **3.4.15. assets/vendors/hover-slider/**

**Назначение**: Слайдер, активируемый при наведении.  
**Использование**: Галереи товаров.

### **3.4.16. assets/vendors/inputmask/**

**Назначение**: Маски для ввода данных (телефон, дата).  
**Использование**: Формы с полями для телефона, кредитных карт.

### **3.4.17. assets/vendors/jquery-\***

* **jquery-json-form-binding/** — привязка JSON-данных к формам.
* **jquery-tags-input/** — ввод тегов (например, для статей).
* **jquery-validation/** — валидация форм на клиенте.

### **3.4.18. assets/vendors/libphonenumber/**

**Назначение**: Парсинг и валидация телефонных номеров.  
**Использование**: Формы регистрации с телефоном.

### **3.4.19. assets/vendors/mdi/**

**Назначение**: Иконки Material Design.  
**Использование**: Иконки в интерфейсе (кнопки, меню).

### **3.4.20. assets/vendors/moment/**

**Назначение**: Работа с датами и временем.  
**Использование**: Форматирование дат в таблицах и уведомлениях.

### **3.4.21. assets/vendors/owl.carousel/**

**Назначение**: Карусель изображений.  
**Использование**: Слайдеры на главной странице.

### **3.4.22. assets/vendors/pdfobject/**

**Назначение**: Встраивание PDF в веб-страницы.  
**Использование**: Просмотр договоров или инструкций.

### **3.4.23. assets/vendors/tinymce/**

**Назначение**: WYSIWYG-редактор для текста.  
**Использование**: Формы создания статей или описаний товаров.

### **3.4.24. assets/vendors/typeahead.js/**

**Назначение**: Автодополнение ввода.  
**Использование**: Поиск по товарам или пользователям.

## **Вывод**

Большинство зависимостей используются для:

* Улучшения UX (формы, слайдеры, редакторы).
* Визуализации данных (графики, таблицы).
* Работы с медиа (изображения, PDF).

### **3.5. libraries/**

**Назначение**: Интеграция сторонних решений.

* **libraries/Smarty/** — шаблонизатор.
* **libraries/SwiftMailer/** — отправка email.

## **3.6. Важные файлы**

### **3.6.1. /index.php**

**Роль**: Точка входа в приложение.  
**Функционал**:

* Инициализация автозагрузчика (system/loader/Loader.php).
* Запуск маршрутизатора (system/core/Router.php).

### **3.6.2. /.htaccess**

**Роль**: Настройки веб-сервера.

* Перенаправляет запросы на index.php (ЧПУ).
* Запрещает доступ к敏感 папкам (например, system/).

## **3.6.3. Интеграция компонентов**

1. **Запрос → Роутинг**:
   * index.php → Router.php → Модуль (app/modules/\*/controller/).
2. **Автозагрузка классов**:
   * Loader.php маппит имена классов на файлы (например, System\Core\Auth → system/core/Auth.php).
3. **Локализация**:
   * Файлы из app/i18n/ и system/i18n/ загружаются через system/drivers/language/i18n.php.

**4. Установка и настройка**

### Системные требования

* PHP ≥ 7.4, MySQL ≥ 5.7, Apache2/Nginx.

### Инструкция по развёртыванию

1. Загрузить исходные файлы в корневую папку сайта.
2. Настроить БД:  
   * Импортировать дамп sql/database.sql.
   * Обновить данные в config/db.php:

php

Copy

define('DB\_HOST', 'localhost');

define('DB\_USER', 'root');

define('DB\_PASS', '');

define('DB\_NAME', 'dashboard\_db');

1. Запустить веб-сервер.