

iWENGO Market

Бэкенд маркетплейса: монолитное PHP-приложение на Laravel с модульной структурой. Отдаёт веб-API для витрины и личного кабинета, интеграционный Merchant API, а также SSR-HTML для двух операторских панелей (админка и MAS).

Стек технологий

Компонент	Версия / продукт
PHP	<code>^8.5</code>
Laravel	<code>^13</code>
БД	PostgreSQL (рабочее окружение), SQLite в памяти в тестах
Тесты	Pest + PHPUnit конфигурация
Поиск	Elasticsearch 7.x (<code>elasticsearch/elasticsearch</code>)
Кэш / сессии	Redis (<code>CACHE_DRIVER</code> , <code>SESSION_DRIVER</code> в <code>.env.example</code>)
Очередь	По умолчанию <code>database</code> (<code>QUEUE_CONNECTION</code> в <code>.env.example</code>)

Дополнительно используются: Spatie Laravel Permission, Socialite (провайдер VK), Flysystem с адаптером S3, YooKassa SDK, ApiShip SDK, Intervention Image, PhpSpreadsheet/OpenSpout и др. — см. `composer.json` .

Структура репозитория

```
app/
├─ Console/           # Общее расписание (см. app/Console/schedule.php), команды при
ложения
├─ Http/              # Глобальные middleware и базовые контроллеры
├─ Modules/           # Доменные модули (см. ниже)
├─ Providers/         # AppServiceProvider, RouteServiceProvider, PanelServiceProvid
er, ...
└─ helpers.php        # Автоподгружаемые хелперы

bootstrap/providers.php # Регистрация Laravel и модульных сервис-провайдеров

config/               # Конфигурация; URL панелей — config/panels.php, config/urls.ph
p

database/
├─ migrations/
├─ factories/
└─ seeders/

routes/
├─ panel/             # Маршруты панелей, привязанные к доменам ADMIN_URL / MAS_URL
|   ├─ admin.php      # включает сегменты (core, orders, ...)
|   └─ mas.php
├─ web/               # Основное веб-API приложения на «API-хосте»
|   ├─ base.php
|   ├─ v1.php         # префикс /v1/... — витрина, корзина, каталог и т.д.
|   ├─ lk.php         # префикс /v1/lk/... — ЛК авторизованного пользователя
|   └─ merchant-api.php
└─ console.php

html/admin/ и html/mas/ # SPA (Vue 2 + Webpack): сборка в public/build/{admin|mas}/

public/               # Точка входа Laravel, статика после сборки панелей
```

Публичный фронт витрины находится в отдельном репозитории (**front**); текущее приложение — HTTP/API и панели, обслуживаемые тем же Laravel-процессом.

Архитектура: доменные модули

Код сгруппирован в пространствах имён `App\Modules{Module}\...`. У каждого модуля обычно есть `Providers/*ServiceProvider.php`, HTTP-слой, модели/DTO/сервисы и при необходимости `Console/Commands`.

Модуль	Назначение (кратко)
Admin	Административная панель: маршруты на домене <code>ADMIN_URL</code> , отчёты, заказы, каталог, маркетинг, настройки.
Mas	Панель MAS (merchant/операций): домен <code>MAS_URL</code> , сценарии отгрузок, возвратов, складской логики для продавца/фулфилмента.
CommonPanel	Общий код для панелей Admin/Mas (хелперы, документы, политики доступа и т.п.).
Auth	Аутентификация и регистрация, соц-логины, сброс пароля.
Blog	Блог на витрине (статьи, реакции).
CatalogImport	Импорт каталога из внешних источников, шаблоны и наблюдатели.
Cm	Customer management: клиентские бонусы, реферальные начисления, балансовые операции.
Cms	Контент: виджеты, лендинги, меню, баннеры, редиректы, кэш-теги.
Common	Перекрёстные сервисы и DTO между модулями.
File	Файлы и изображения: выдача контента, облачное хранение, ресайзы.
Logistics	Доставки, точки выдачи, тарифы, интеграции (в т.ч. ApiShip), резервы слотов.
Marketing	Скидки, промокоды, бундлы, бонусы, калькуляторы акций.
Merchant	Сущность продавца, биллинг, выплаты, консоли агентом.
MerchantApi	HTTP API для интеграций продавца (<code>/merchant-api/...</code>): Basic Auth, экспорт/импорт офферов, заказов, остатков и т.д.
Message	Сообщения, уведомления, SMS/email сценарии.
Oms	Order management: корзина, чекаут, оплата, отгрузки, возвраты (ядро операций с заказом).
Pim	Product information: товары, офферы, вариантные группы, сертификаты, публичные события, индексация в Elasticsearch.
Showcase	Логика витрины: каталог, листинги, SEO/meta, карточки, связка с Pim/Oms для корзины.
Store	Настройки торговых точек/магазинов, резервы.
Lk	Личный кабинет покупателя: заказы, адреса, избранное, сообщения (<code>/v1/lk/...</code>).

Регистрация модулей: `bootstrap/providers.php`.

Маршрутизация и хосты

Загрузка маршрутов настраивается в `bootstrap/app.php` через `RouteServiceProvider::registerRoutes()` (после `health: /up`).

1. **Панели по отдельным доменам** (`config/panels.php` получает хосты из `ADMIN_URL` и `MAS_URL`):

- админка: `routes/panel/admin.php` + подключаемые файлы;
- MAS: `routes/panel/mas.php` с префиксом имён маршрутов `mas.*`.

Для каждого HTTP-запроса определяется строка `'panel'` в контейнере (`''`, `'admin'` или `'mas'`) в `PanelServiceProvider` по совпадению хоста — это используется в шаблонах и макросе `View::component()` для проброса маршрутов во фронт панелей.

2. **Основное веб-приложение на API-хосте** — группы из `routes/web/` с middleware `web`:

- `base.php` — корень приложения (`/balance`, статика файлов через `/content/...`, отладочные маршруты при `ENABLE_DEBUG_ROUTES`).
- `v1.php` — публичное и авторизованное API витрины под префиксом `/v1/`.
- `lk.php` — ЛК покупателя под `/v1/lk/` (частично продублировано включением через отдельный файл для ленивой подгрузки).
- `merchant-api.php` — `/merchant-api/...` с Basic Auth (`merchant_api`), лимит запросов и логгер.

Ленивая подгрузка маршрутов: в HTTP без `route:cache` файлы `v1.php`, `lk.php`, `merchant-api.php`, `base.php` подключаются по префиксу пути текущего запроса — чтобы не грузить весь граф на каждый запрос. В консоли, тестах и при кэше маршрутов (`php artisan route:cache`) подключаются все нужные группы полностью (см. комментарии в `RouteServiceProvider`).

Ограничения параметров маршрутов: в `RouteServiceProvider` заданы паттерны для числовых `id`, `order_id`, `shipment_id` и т.д., а также `slug`.

Middleware

Глобальные настройки в `bootstrap/app.php`: кастомные `TrustProxies`, `TrimCookies` / `EncryptCookies`, глобальный `JsonMiddleware`, алиасы `auth`, `guest`, `active` (`IsActiveProfile`), редирект гостей на `/`.

Конфигурация окружения

Ориентир по переменным — `.env.example`: `APP_*`, URL панелей и витрины, PostgreSQL, Redis, почта, SMSC, Elasticsearch, ЮKassa, ApiShip, DaData, лимиты Merchant API и флаги отладки.

`config/urls.php` при пустых `ADMIN_URL` / `MAS_URL` / `SHOWCASE_HOST` может вывести значения из `APP_URL`, если hostname содержит подстроку `api.` (замена на `admin.`, `mas.` или отрезание префикса для витрины).

База данных и фоновые задачи

- Миграции: `database/migrations/`.
- Планировщик Laravel: `app/Console/schedule.php` — sitemap, кэш страниц, импорт логистики, платежи мерчантам, синхронизация Elasticsearch (`ElasticRun`), смена статусов заказов/возвратов/промокодов и др. На продакшене требуется cron: `php artisan schedule:run`.
- Консольные команды дополнительно регистрируются из модулей (см. `bootstrap/app.php`, `withCommands`).

Тестирование

```
composer test                # все сьюты из phpunit.xml
composer test:admin          # только Admin-модуль
composer test:mas            # только Mas-модуль
composer test:coverage      # покрытие по модулям (отдельный конфиг)
```

В `phpunit.xml` для тестов задаются `ADMIN_URL`, `MAS_URL`, in-memory SQLite, драйверы `array/sync` и т.д. Для симуляции панели в части загрузки маршрутов используется `APP_TEST_PANEL` (см. комментарий в `phpunit.xml`).

Сборка фронтенда панелей

Панели — Vue 2 и Webpack (см. `html/admin/package.json`, `html/mas/package.json`). Скрипты сборки пишут артефакты в `public/build/admin` и `public/build/mas`. Требуется Node.js/Yarn по версиям проекта; для сборки может понадобиться `NODE_OPTIONS=--openssl-legacy-provider` (указано в скрипте `admin-build`).

Полезные команды Artisan

- `php artisan route:list` — список маршрутов (учитывать `route:cache` и домены панелей).
- `php artisan migrate` — применить миграции.
- `php artisan queue:work` — обработчик очереди при `QUEUE_CONNECTION` $\neq$ `sync`.

Автозагрузка и пространство имён

- Код приложения: PSR-4 префикс `App` → каталог` app/``.
- Тестовый код: `Tests` → tests/`` (если используется).
- Глобальные функции из `app/helpers.php` подключаются через `composer.json` (`autoload.files`).

Этот документ описывает устройство репозитория `iwengo/api` на уровне разработки и эксплуатации. Детали доменной логики смотрите в соответствующих модулях `App\Modules*` и привязанных к ним маршрутах.