Инструкция по развертыванию IK Backend

Структура проекта

После распаковки архива вы увидите следующую структуру:

- **app/** – основной код приложения: бизнес-логика, модули, модели, контроллеры, команды, интеграции
- **bootstrap/** – файлы запуска и кеш конфигурации
- **config/** – конфигурационные файлы (БД, кеш, почта, очереди, Nova и т.д.)
- **database/** – миграции, фабрики и сидеры
- **deployment/** – скрипты и конфиги для деплоя (cron, supervisord, unit, php.ini)
- **docker/** – Dockerfile и конфиги для разных версий PHP
- **public/** – точка входа (index.php), статика (favicon, robots.txt)
- **resources/** – frontend-ресурсы: CSS, JS, Blade-шаблоны, локализация
- **routes/** – файлы маршрутов (web.php, console.php)
- **storage/** – логи, кеш, загружаемые файлы
- **tests/** – unit- и feature-тесты
- **vendor/** – сторонние пакеты (Composer)
- **backups/** – дампы БД (Yandex Managed PostgreSQL)
- **nova-components/** – кастомные компоненты для Laravel Nova

Выбор способа развертывания

Приложение можно развернуть двумя способами:

1. **На локальном компьютере** - для тестирования и ознакомления
2. **На удаленном сервере (VPS)** - для постоянной работы с доступом через интернет

Вариант 1: Развертывание на локальном компьютере

Минимальные требования

- **ОС**: Windows 10/11, macOS 10.15+, или Linux (Ubuntu 20.04+)
- **Процессор**: 2 ядра
- **Оперативная память**: 4 ГБ
- **Свободное место**: 10 ГБ
- **Права**: права администратора для установки Docker

Шаг 1: Установка Docker

Windows

1. Скачать Docker Desktop для Windows
2. Запустить установщик и следовать инструкциям

3. Перезагрузить компьютер
4. Запустить Docker Desktop
5. Дождаться запуска (иконка Docker в трее должна быть зеленой)

macOS

1. Скачать Docker Desktop для Mac
2. Открыть .dmg файл и перетащить Docker в Applications
3. Запустить Docker из Applications
4. Разрешить все запросы системы
5. Дождаться запуска

Linux (Ubuntu/Debian)

Открыть терминал и выполнить:

```
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
sudo usermod -aG docker $USER
```

Важно: После выполнения команд полностью перелогиньтесь (выйдите и войдите в систему).

Шаг 2: Проверка установки Docker

Откройте терминал (командную строку) и выполните:

```
docker --version
docker compose version
```

Вы должны увидеть версии Docker (например, `Docker version 24.0.0`) и Docker Compose (например, `Docker Compose version 2.20.0`).

Шаг 3: Получение проекта

```
# Перейдите в папку, где хотите разместить проект
cd /путь/к/папке

# Скачайте архив с Яндекс.Диска
# Ссылка: https://disk.yandex.ru/d/HIjo61wZfgxwtw
# Сохраните файл как ik-backend.zip

# Распакуйте архив
unzip ik-backend.zip
cd ik-backend
```

Если команда `unzip` не найдена:

```
# Установите unzip
sudo apt install unzip
# Затем повторите распаковку
unzip ik-backend.zip
```

Шаг 4: Настройка окружения

```
# Скопируйте файл с настройками
cp .env.example .env
```

Если порт 80 на вашем компьютере занят, откройте файл `.env` и измените:

```
APP_PORT=8080
```

Шаг 5: Запуск приложения

```
# Запустить все сервисы
docker compose up -d
```

Первый запуск займет 5-10 минут (Docker будет скачивать образы). Последующие запуски - 30 секунд.

Дождитесь завершения и проверьте статус:

```
docker compose ps
```

Все сервисы должны быть в статусе `Up` или `healthy`.

Шаг 6: Инициализация приложения

```
# Установить зависимости PHP
docker compose exec laravel.test composer install

# Сгенерировать ключ приложения
docker compose exec laravel.test php artisan key:generate

# Создать структуру базы данных
docker compose exec laravel.test php artisan migrate

# Заполнить тестовыми данными (опционально)
docker compose exec laravel.test php artisan db:seed
```

Шаг 7: Проверка работоспособности

Откройте в браузере: - **http://localhost** (или **http://localhost:8080** если меняли порт)

Вы должны увидеть главную страницу приложения.

Вариант 2: Развертывание на VPS сервере

Минимальные требования к серверу

- **ОС**: Ubuntu 20.04 LTS или выше (рекомендуется Ubuntu 22.04 LTS)
- **Процессор**: 2 vCPU
- **Оперативная память**: 4 ГБ
- **Диск**: 20 ГБ SSD
- **Сеть**: выделенный IP адрес

Шаг 1: Подключение к серверу

После создания VPS вы получите: - IP адрес сервера (например: 123.45.67.89) - Логин (обычно `root` или `ubuntu`) - Пароль или SSH ключ

Подключитесь к серверу:

Windows (используйте PuTTY или встроенный SSH):

```
ssh root@123.45.67.89
```

macOS/Linux:

```
ssh root@123.45.67.89
```

Введите пароль когда попросит.

Шаг 2: Обновление системы

После подключения выполните:

```
# Обновить список пакетов
apt update

# Установить обновления
apt upgrade -y
```

Шаг 3: Установка Docker

```
# Установить Docker
curl -fsSL https://get.docker.com -o get-docker.sh
sh get-docker.sh

# Проверить установку
docker --version
docker compose version
```

Шаг 4: Получение проекта

```
# Создать папку для проекта
mkdir -p /var/www/ik-backend
cd /var/www/ik-backend

# Загрузить архив ik-backend.zip на сервер
# Скачайте архив с Яндекс.Диска: https://disk.yandex.ru/d/HIjo61wZfgxwtw

# Вариант 1: Загрузить через scp с локального компьютера
# scp ik-backend.zip root@123.45.67.89:/var/www/ik-backend/

# Вариант 2: Загрузить напрямую на сервер через wget
# wget --content-disposition "https://disk.yandex.ru/d/HIjo61wZfgxwtw" -O ik-backend.zip

# Распаковать архив
apt install unzip -y
unzip ik-backend.zip
```

Шаг 5: Настройка окружения

```
# Скопировать конфигурацию
cp .env.example .env

# Отредактировать настройки
nano .env
```

Измените следующие параметры:

```
APP_URL=http://ваш-ip-адрес
APP_PORT=80

# Если нужен доступ извне к БД (обычно не требуется)
FORWARD_DB_PORT=5432
FORWARD_REDIS_PORT=6379
```

Сохраните файл: `Ctrl+O`, `Enter`, `Ctrl+X`

Шаг 6: Настройка файрвола

```
# Установить UFW (если не установлен)
apt install ufw -y

# Разрешить SSH (ВАЖНО! Иначе потеряете доступ)
ufw allow 22/tcp

# Разрешить HTTP
ufw allow 80/tcp

# Разрешить HTTPS (если планируете использовать)
ufw allow 443/tcp

# Включить файрвол
ufw enable
```

При вопросе подтвердите: `y`

Шаг 7: Запуск приложения

```
# Запустить все сервисы
docker compose up -d
```

Первый запуск займет 5-10 минут.

Шаг 8: Инициализация приложения

```
# Установить зависимости
docker compose exec laravel.test composer install

# Сгенерировать ключ
docker compose exec laravel.test php artisan key:generate

# Создать базу данных
docker compose exec laravel.test php artisan migrate
```

```
# Тестовые данные (опционально)
docker compose exec laravel.test php artisan db:seed
```

Шаг 9: Проверка работоспособности

Откройте в браузере: - **http://ваш-ip-адрес**

Например: **http://123.45.67.89**

Проверка работоспособности

После развертывания любым способом выполните:

1. Проверка статуса сервисов

```
docker compose ps
```

Результат должен показывать все сервисы в статусе `Up`:

```
NAME                    STATUS
ik-laravel.test-1       Up (healthy)
ik-pgsql-1              Up (healthy)
ik-redis-1              Up (healthy)
ik-minio-1              Up (healthy)
```

2. Проверка логов

```
# Посмотреть последние логи
docker compose logs --tail=50

# Следить за логами в реальном времени
docker compose logs -f laravel.test
```

Не должно быть ошибок типа `ERROR`, `FATAL`, `Connection refused`.

3. Проверка базы данных

```
docker compose exec laravel.test php artisan migrate:status
```

Должен показать список выполненных миграций.

4. Проверка веб-доступа

Откройте браузер и перейдите: - Локально: **http://localhost** - VPS: **http://IP-адрес-сервера**

Страница должна загрузиться без ошибок.

Доступ к дополнительным сервисам

MinIO (файловое хранилище)

- **Адрес**: http://localhost:8900 (локально) или http://IP-адрес:8900 (VPS)
- **Логин**: `sail`

- **Пароль**: `password`

База данных PostgreSQL

Для подключения внешним клиентом (DBeaver, pgAdmin):

- **Хост**: localhost (или IP сервера)
- **Порт**: 5432
- **База**: название из `.env` (параметр `DB_DATABASE`)
- **Пользователь**: название из `.env` (параметр `DB_USERNAME`)
- **Пароль**: из `.env` (параметр `DB_PASSWORD`)

Управление приложением

Остановка

```
docker compose stop
```

Запуск после остановки

```
docker compose start
```

Перезапуск

```
docker compose restart
```

Полное удаление (включая все данные)

```
docker compose down -v
```

Внимание: эта команда удалит все данные из базы данных и файлы!

Решение типичных проблем

“Cannot connect to Docker daemon”

Решение для Linux:

```
sudo usermod -aG docker $USER
```

После этого полностью перелогиньтесь в системе.

Решение для Windows/Mac: Убедитесь что Docker Desktop запущен (иконка в трее).

“Port is already allocated” (порт уже занят)

Измените порт в `.env`:

```
APP_PORT=8080
```

Перезапустите:

```
docker compose down
docker compose up -d
```

“Connection refused” при подключении к БД

```
# Проверить статус PostgreSQL
docker compose exec pgsql pg_isready

# Если не отвечает - пересоздать
docker compose restart pgsql

# Подождать 30 секунд и проверить снова
```

Ошибки при выполнении миграций

```
# Удалить базу и создать заново
docker compose down -v
docker compose up -d

# Подождать 1 минуту пока запустится PostgreSQL
docker compose exec laravel.test php artisan migrate
```

“Permission denied” ошибки

```
docker compose exec laravel.test chmod -R 777 storage bootstrap/cache
```

На VPS не открывается сайт

1. Проверьте что файрвол разрешает порт 80:

```
ufw status
```

2. Проверьте что приложение запущено:

```
docker compose ps
```

3. Проверьте что используете правильный IP:

```
curl ifconfig.me
```

Просмотр логов для диагностики

```
# Все логи приложения
docker compose logs -f laravel.test

# Логи базы данных
docker compose logs -f pgsql

# Логи конкретного сервиса за последние 100 строк
docker compose logs --tail=100 laravel.test
```

Технические характеристики

- **PHP**: 8.3
- **Framework**: Laravel 12.0
- **База данных**: PostgreSQL 17
- **Кеш/Очереди**: Redis (Alpine)
- **Файловое хранилище**: MinIO (S3-совместимое)

- **Веб-сервер**: встроенный сервер Laravel через Sail