

Установка тарификатора из дистрибутива

Требования к системному окружению:

ОС хоста: Linux, архитектура x86-64

Системное ПО:

- docker
- docker compose v2
- nginx (либо другой обратный прокси)

Аппаратные требования (минимальные для запуска, реальные же зависят от конкретного профиля нагрузки):

- 2 vCPU
- 4 GB RAM
- 16 GB свободного места на диске

Установка программного обеспечения “Тарификатор доставки” из дистрибутива

После распаковки архива с дистрибутивом командой `tar -zxvf tarifikator-distrib.tar.gz`, перейдите в директорию с распакованным архивом: `cd tarifikator-distrib`

После распаковки дистрибутива в директории проекта `tarifikator-distrib` необходимо скопировать файл `.env.example` в `.env` и выполнить необходимые настройки файла `.env`.

Дополнительно нужно в директории с `tarifikator-backend` скопировать файл `.env.example` в `.env` и произвести в нем необходимые настройки, в частности `APP_URL` должен указывать на адрес сервера на котором расположен backend (напр. `APP_URL=https://<домен_установленного_тарификатора>:8097`).

Остальные параметры (доступы к бд, настройки почты и пр.) настраиваются соответственно имеющемуся окружению (доступы к бд задаются в конфигурации `docker-compose` и по умолчанию совпадают между ним и `backend`, таким образом их настройка требуется, только если в конфигурации `compose` они были изменены).

Далее можно выполнить сборку и запуск проекта командой `docker compose up -d` (необходимо находиться в директории с проектом `tarifikator-distrib`)

Далее нужно очистить кэш и собрать фронтенд:
`docker compose exec php composer install`

После окончания сборки нужно выполнить миграции (только в случае, если сборка выполнена впервые и база данных еще пуста):
`docker compose exec php php artisan migrate --force`

```
docker compose exec php php artisan config:clear
docker compose exec php php artisan cache:clear
docker compose exec php php artisan route:cache
docker compose exec php php artisan view:cache
docker compose exec php npm ci
docker compose exec php npm run build
```

Заполнение БД тестовыми данными и создание первичной учетной записи:

Для проверки корректности развёртывания программного обеспечения может быть создана тестовая учётная запись администратора и выполнено заполнение базы данных тестовыми данными.

Данный этап является необязательным и предназначен для тестового или демонстрационного контура. Тестовые данные не требуются для промышленной эксплуатации программного обеспечения.

Для создания тестовой учётной записи администратора выполнить команду:

```
docker compose exec php php artisan db:seed --class=UsersTableSeeder
```

Будет создана тестовая учётная запись:

логин: `admin@test.com`
пароль: `password123`

Для наполнения базы данных тестовыми данными выполнить команду:

```
docker compose exec php php artisan app:testing-data
```

При промышленном развёртывании команду заполнения базы тестовыми данными выполнять не следует. Если тестовые учётные записи создавались для проверки установки, до ввода программного обеспечения в промышленную эксплуатацию необходимо изменить

их пароли либо удалить тестовые учётные записи и создать рабочие учётные записи в соответствии с политикой информационной безопасности заказчика.

```
docker compose exec php php artisan db:seed --class=UsersTableSeeder
```

Переменные окружения

Файл env:

APP_NAME=Laravel

Название приложения, которое используется в заголовках, уведомлениях и других местах (стандартная переменная laravel).

APP_ENV=local

Определяет текущую среду выполнения приложения (например, local, production, testing) (стандартная переменная laravel).

APP_KEY=

Ключ шифрования для Laravel, который используется для защиты данных (генерируется командой `php artisan key:generate`) (стандартная переменная laravel).

APP_DEBUG=true

Включает или отключает режим отладки (вывод ошибок). В production рекомендуется установить значение false (стандартная переменная laravel).

APP_TIMEZONE=Europe/Moscow

Часовой пояс приложения (стандартная переменная laravel).

APP_URL=http://localhost:8097

URL приложения, который используется для генерации ссылок внутри приложения и др. (адрес сервера на котором расположен backend).

APP_LOCALE=ru

Основной язык приложения (стандартная переменная laravel).

APP_FALLBACK_LOCALE=en

Язык резервной локализации, если основной язык недоступен (стандартная переменная laravel).

APP_FAKER_LOCALE=en_US

Локаль для генерации данных через библиотеку Faker (стандартная переменная laravel).

APP_SQL_DEBUG=true

Включает или отключает вывод SQL-запросов в логи (логируются все sql запросы в файлы разделенные по дням, расположение файлов storage/logs/, используется для отладки на проде рекомендуется значение false).

APP_MAINTENANCE_DRIVER=file

Драйвер для управления режимом обслуживания (например, file или database) (стандартная переменная laravel).

PHP_CLI_SERVER_WORKERS=4

Количество рабочих процессов для встроенного PHP-сервера (стандартная переменная laravel).

BCRYPT_ROUNDS=12

Количество раундов для хеширования паролей с использованием Bcrypt (стандартная переменная laravel).

LOG_CHANNEL=stack

Канал логирования по умолчанию (например, stack, single, daily) (стандартная переменная laravel).

LOG_STACK=single

Список каналов, которые объединяются в стековый канал (стандартная переменная laravel).

LOG_DEPRECATIONS_CHANNEL=null

Канал для записи предупреждений о устаревших функциях (стандартная переменная laravel).

LOG_LEVEL=debug

Минимальный уровень логирования (например, debug, info, error) (стандартная переменная laravel).

DB_CONNECTION=pgsql

Тип подключения к базе данных (например, mysql, pgsql, sqlite) (стандартная переменная laravel).

DB_HOST=pgsql

Хост базы данных (стандартная переменная laravel).

DB_PORT=5432

Порт базы данных (стандартная переменная laravel).

DB_DATABASE=laravel

Имя базы данных (стандартная переменная laravel).

DB_USERNAME=

Имя пользователя для подключения к базе данных (стандартная переменная laravel).

DB_PASSWORD=

Пароль для подключения к базе данных (стандартная переменная laravel).

TEST_DB_HOST=

Хост тестовой базы данных.

TEST_DB_PORT=5433

Порт тестовой базы данных.

TEST_DB_DATABASE=test

Имя тестовой базы данных.

`SESSION_DRIVER=database`

Драйвер для хранения сессий (например, file, database, cookie) (стандартная переменная laravel).

`SESSION_LIFETIME=120`

Время жизни сессии в минутах (стандартная переменная laravel).

`SESSION_ENCRYPT=false`

Включает или отключает шифрование данных сессии (стандартная переменная laravel).

`SESSION_PATH=/`

Путь для cookie сессии (стандартная переменная laravel).

`SESSION_DOMAIN=null`

Домен для cookie сессии (стандартная переменная laravel).

`BROADCAST_CONNECTION=log`

Драйвер для вещания событий (например, log, pusher, redis) (стандартная переменная laravel).

`FILESYSTEM_DISK=local`

Драйвер для работы с файловой системой (например, local, s3, public) (стандартная переменная laravel).

`QUEUE_CONNECTION=database`

Драйвер для работы с очередями (например, sync, database, redis) (стандартная переменная laravel).

`CACHE_STORE=database`

Драйвер для кэширования данных (например, file, database, redis) (стандартная переменная laravel).

`CACHE_PREFIX=`

Префикс для ключей кэша (стандартная переменная laravel).

`MEMCACHED_HOST=127.0.0.1`

Хост Memcached (стандартная переменная laravel).

`REDIS_CLIENT=phpredis`

Клиент Redis (например, phpredis или predis) (стандартная переменная laravel).

`REDIS_HOST=127.0.0.1`

Хост Redis (стандартная переменная laravel).

`REDIS_PASSWORD=null`

Пароль Redis (стандартная переменная laravel).

`REDIS_PORT=6379`

Порт Redis (стандартная переменная laravel).

MAIL_MAILER=log

Драйвер для отправки почты (например, smtp, log, sendmail).

MAIL_SCHEME=null

Схема для почтового сервера (например, http, https) (стандартная переменная laravel).

MAIL_HOST=127.0.0.1

Хост SMTP-сервера (стандартная переменная laravel).

MAIL_PORT=2525

Порт SMTP-сервера (стандартная переменная laravel).

MAIL_USERNAME=null

Имя пользователя для SMTP (стандартная переменная laravel).

MAIL_PASSWORD=null

Пароль для SMTP.

MAIL_FROM_ADDRESS="hello@example.com"

Адрес отправителя по умолчанию (стандартная переменная laravel).

MAIL_FROM_NAME="\${APP_NAME}"

Имя отправителя по умолчанию (стандартная переменная laravel).

VITE_APP_NAME="\${APP_NAME}"

Переменная для Vite с названием приложения.

VITE_YANDEX_API_KEY=

API-ключ для Yandex Maps.

VITE_API_URL="\${APP_URL}"

URL API для Vite.

YANDEX_GEO_API_URL=https://geocode-maps.yandex.ru/1.x

URL для геокодирования.

YANDEX_DISTANCE_API_URL=https://api.routing.yandex.net/v2/distancematrix

URL для расчета расстояний.

YANDEX_DISTANCE_MODE=truck

Режим транспорта для расчета расстояний.

YANDEX_DISTANCE_AVOID_TOLLS=true

Избегать платных дорог.

YANDEX_DISTANCE_TRAFFIC=disabled

Учитывать пробки.

YANDEX_DISTANCE_DEFAULT_HEIGHT=3

Высота транспорта по умолчанию в метрах.

YANDEX_DISTANCE_DEFAULT_PAYLOAD=5

Грузоподъемность транспорта по умолчанию в тоннах.

SANCTUM_STATEFUL_DOMAINS=localhost:8097

Домены, для которых Sanctum будет работать в stateful-режиме.

EXPORT_SYSTEM_BASE_URL=http://<адрес_внешней_системы>/endpoint

Базовый URL для экспорта тарифов в 1С. (ВАЖНО: URL не должен заканчиваться закрывающим слэшем)

EXPORT_SYSTEM_USERNAME=Tarificator

Имя пользователя для экспорта данных.

EXPORT_SYSTEM_PASSWORD=

Пароль для экспорта данных.

LOKI_URL=http://loki:3100

URL для Loki (система логирования).

LOKI_USER=

Имя пользователя для Loki.

LOKI_PASSWORD=

Пароль для Loki.

Генерация файлов для подключения виджета

Чтобы виджет отображался через nginx

Необходимо сначала войти в контейнер `docker compose exec widget sh`

Указать ключ yandex карт в .env `VITE_YANDEX_API_KEY=`

Затем перейти в директорию `cd /usr/share/nginx/html/`

Затем выполнить `npm install`

Затем выполнить `npm build:web`

В виджете имеется команда `npm build:lib` которая генерирует 2 файла `tariffs-widget.css`, `tariffs-widget.umd.js`.

Файлы должны быть обновлены каждый раз при внесении изменений в фронт виджета.

Данные файлы должны быть размещены на сервере который доступен извне, и чтобы данные файлы можно было подключить к ресурсу в который внедряется виджет.

Нужно выдавать данные актуальные файлы напрямую для размещения на сервере ресурса, куда внедряется виджет. Если файлы будут выдаваться в ручную, то при изменении в репозитории виджета, нужно актуализировать данные файлы на всех сервисах.

В директории `tarifikator-widget` нужно скопировать файл `.env.example` в `.env` и внести туда ссылку на бэкенд тарификатора и ключ для Яндекс-карт, это нужно сделать до запуска команды сборки.

Сборка выполняется командой `docker compose exec widget npm build:lib`

Настройки CORS

Домены, на которых вызывается виджет должны быть прописаны в файле `config/cors.php` в директории `tarifikator-backend`:

```
'allowed_methods' => ['*'],

'allowed_origins' => [
    'http://localhost:5173',
    'http://localhost:5174',
    'http://localhost:8099',
    'http://localhost:8097',
    'http://домен_для_бэкенда:8097',
    'http://домен_для_виджета:8099'
],

'allowed_origins_patterns' => [''],

'allowed_headers' => ['*'],

'exposed_headers' => ['*'],

'max_age' => 0,

'supports_credentials' => true,
```

Настройка nginx в качестве входного реверс-прокси

Файлы с примерами конфигурации nginx находятся в директории nginx-inbound. Однако, для корректной работы требуется https (файлы в примерах отражают только http).

В системе два пользовательского сервиса на портах по умолчанию:

- backend на порту 8097
- widget на порту 8099

Соответственно прокси надо настраивать на них

Пользователи по умолчанию

После развёртывания проекта по умолчанию в системе будут созданы следующие группы пользователей:

- Администратор
- Руководитель группы
- Пользователь
- Системный пользователь

Группы пользователей влияют на доступный функционал в административной панели согласно матрице прав:

Функционал / Роль	Администратор	Руководитель группы	Пользователь	Системный пользователь
Регионы (зоны доставки)	CRU	CRU	CRU	—
Тарифы	R	R	R	—
Праздничные дни	R	R	R	—
Товары исключения	R	R	R	—
Виджет справочный	R	R	R	—
Виджет окончательный	R	R	R	—
Пользователи	CRUD	RU	R	—
Код виджета	R	—	—	—
Экспорт зон	RU	RU	R	—
Настройки	RU	—	—	—

При выполнении инициализации тестовыми данными в системе могут быть созданы следующие тестовые учётные записи:

Администратор: admin@test.com / password123

Руководитель группы: manager@test.com / password123

Пользователь: user@test.com / password123

Системный пользователь: system@test.com / password123

Указанные учётные записи предназначены только для тестового и демонстрационного использования. При вводе программного обеспечения в промышленную эксплуатацию тестовые учётные записи должны быть удалены либо их пароли должны быть изменены.

Рабочие учётные записи создаются и настраиваются администратором в соответствии с требованиями заказчика.