

# Fly Potato

(ДПиИРГМ и ФЦБАС / «Fly Potato Team»)

## Руководство администратора

**Код проекта:** UAVD

**Дата:** 2 октября 2025 г.

**Версия:** 0.1

**Страниц:** 15

### СОГЛАСОВАНО / УТВЕРЖДЕНО

**От заказчика:** ..... **От исполнителя:** .....

Должность, ФИО, подпись, дата

Должность, ФИО, подпись, дата

# Содержание

<b>1</b>	<b>Архитектура и состав</b>	<b>3</b>
1.1	Компоненты и профили	3
1.2	Профили запуска	3
1.3	Схема	3
1.3.1	DEV	3
1.3.2	PROD	4
1.4	Порты/протоколы	4
1.4.1	DEV	4
1.4.2	PROD	4
<b>2</b>	<b>Требования к инфраструктуре</b>	<b>5</b>
2.1	Платформа и версии	5
2.2	Ресурсы и sizing (минимум для PROD)	6
2.3	Сеть и сегментация	6
2.4	DNS / TLS / Время	6
2.5	Доступы и безопасность	6
2.6	База данных (Postgres/PostGIS)	7
2.7	Мониторинг и алерты	7
2.8	Интеграция с DataLens	7
2.9	Разделение сред	7
<b>3</b>	<b>Установка и запуск</b>	<b>8</b>
3.1	DEV	8
3.1.1	Подготовка окружения	8
3.1.2	База данных (PostGIS) и миграции	8
3.1.3	backend	8
3.1.4	Авторизация (Keycloak + oauth2-proxy + nginx)	8
3.1.5	Мониторинг	8
3.2	PROD	9
3.2.1	Предварительные требования	9
3.2.2	Подготовка конфигов	9
3.2.3	Настройка Nginx	9
3.2.4	Настройка Keycloak	9
3.2.5	Запуск	10
3.3	DataLens	10
3.3.1	Предпосылки и безопасность	10
3.3.2	Подготовка	10
3.3.3	Импорт воркбука	10
3.3.4	Подключение к БД	11
3.3.5	Встраивание дашборда в сайт (через JWT)	11

---

<b>4</b>	<b>Конфигурация (DEV + PROD)</b>	<b>11</b>
4.1	Переменные окружения	11
<b>5</b>	<b>Зависимости (DEV + PROD)</b>	<b>11</b>
<b>6</b>	<b>Резервное копирование и восстановление</b>	<b>12</b>
6.1	DEV	12
6.2	PROD	12
6.2.1	Команды (пример)	13
6.2.2	Восстановление (PITR, кратко)	13
6.2.3	Политики	13
<b>7</b>	<b>Мониторинг и оповещения</b>	<b>13</b>
7.1	Prometheus targets (пример)	13
7.2	Базовые правила (alert.rules.yaml)	14
<b>8</b>	<b>Доступы и безопасность</b>	<b>14</b>
8.1	DEV	14
8.2	PROD	14
<b>9</b>	<b>Эксплуатационные регламенты</b>	<b>15</b>
9.1	DEV	15
9.1.1	Пересоздать БД	15
9.1.2	Горячая перезагрузка backend	15
9.1.3	Логи быстрого доступа	15
9.2	PROD	15

# 1 Архитектура и состав

## 1.1 Компоненты и профили

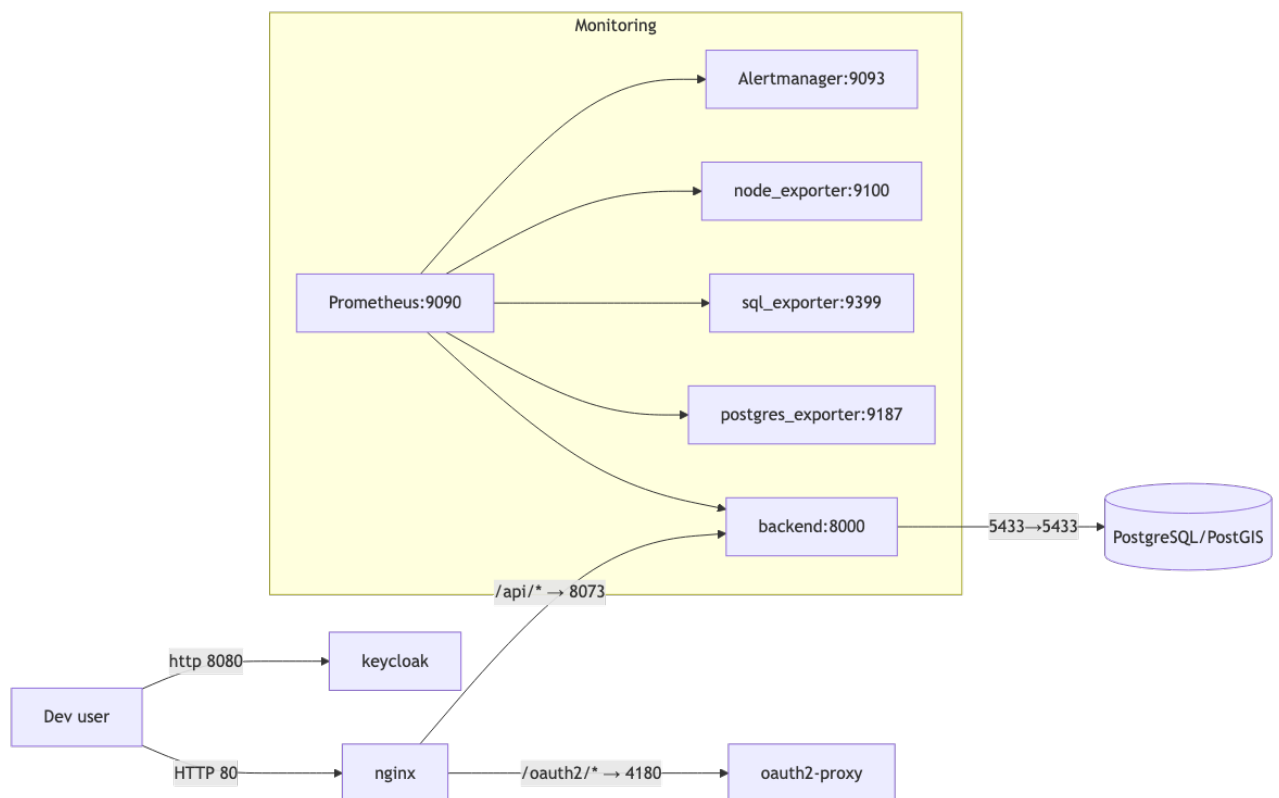
- core: postgres (PostGIS), migration (alembic), backend (FastAPI).
- monitoring: prometheus, alertmanager, node\_exporter, postgres\_exporter, sql\_exporter.
- auth: keycloak, oauth2-proxy, nginx.

## 1.2 Профили запуска

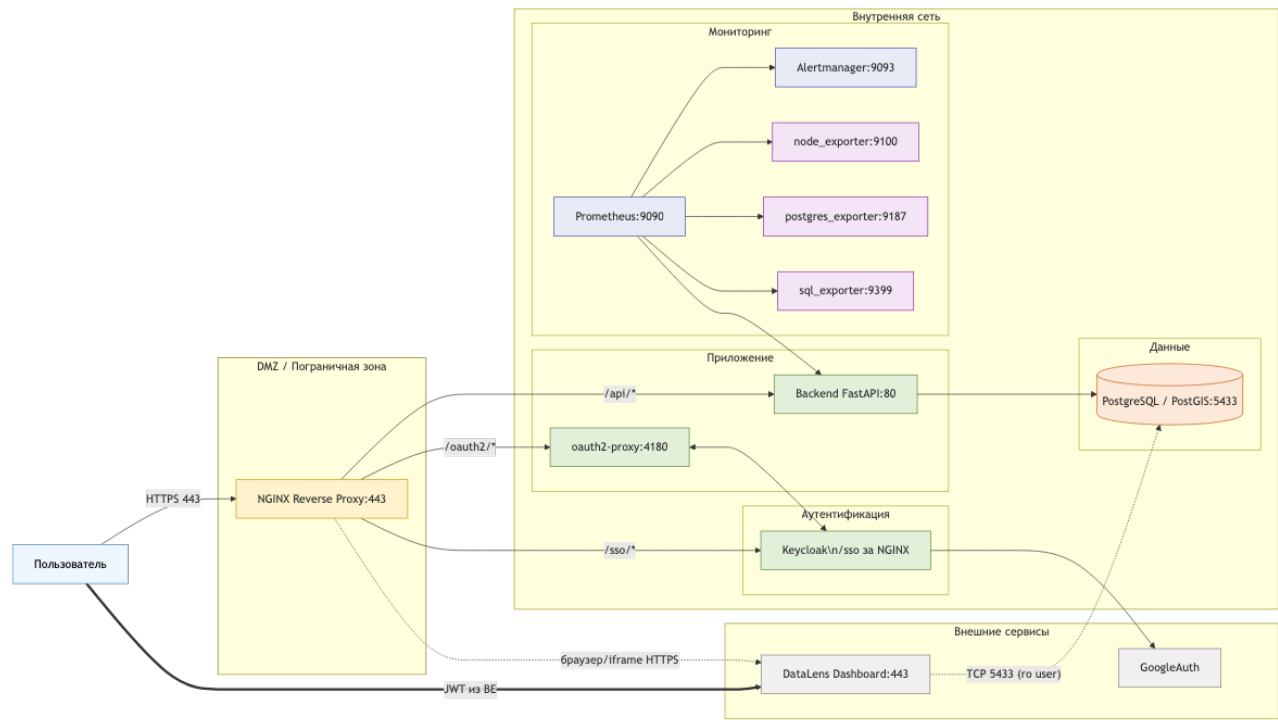
- Только ядро: `docker compose --profile core up -d`
- Ядро + авторизация: `docker compose --profile core --profile auth up -d`
- Всё (DEV полный): `docker compose --profile all up -d`

## 1.3 Схема

### 1.3.1 DEV



1.3.2 PROD



1.4 Порты/протоколы

1.4.1 DEV

Сервис	Порт (хост → контейнер)	Протокол	Назначение
postgres (PostGIS)	5433 → 5433	TCP	БД приложения
backend	8073 → 8000	HTTP	API/health/metrics
nginx	80 → 80	HTTP	Реверс-прокси (dev, без TLS)
keycloak	8080 → 8080	HTTP	IdP (Realm import)
oauth2-proxy	4180 → 4180	HTTP	OIDC прокси
prometheus	9090 → 9090	HTTP	Метрики/графы
alertmanager	9093 → 9093	HTTP	Алерты
node_exporter	9100 → 9100	HTTP	Метрики хоста
postgres_exporter	9187 → 9187	HTTP	Метрики БД
sql_exporter	9399 → 9399	HTTP	Кастомные SQL-метрики

1.4.2 PROD

В PROD наружу открыт только **HTTPS 443 → NGINX** (и опц. HTTP 80 для редиректа). Все остальные сервисы не публикуют порты наружу и доступны лишь из внутренней сети.

Внешние входящие (ingress)

Сервис	Порт (внешний → внутренний)	Протокол	Назначение
<b>nginx</b>	<b>443 → 443</b>	HTTPS	Единственная внешняя точка входа
(опц.) <b>nginx</b>	<b>80 → 80</b>	HTTP	Редирект на HTTPS 443

### Внутренние сервисы (без публикации наружу)

Сервис	Порт (внутри сети)	Протокол	Назначение
<b>backend</b>	80	HTTP	API/health/metrics (доступ только через nginx)
<b>oauth2-proxy</b>	4180	HTTP	OIDC-прокси (nginx ↔ oauth2-proxy)
<b>keycloak</b>	8080	HTTP	IdP; публикуется наружу <b>только</b> через nginx /sso/*
<b>postgres (PostGIS)</b>	5433	TCP	База данных (доступ только из внутренней сети)
<b>prometheus</b>	9090	HTTP	Мониторинг (доступ из VPN/внутренней сети)
<b>alertmanager</b>	9093	HTTP	Оповещения (внутренняя сеть/VPN)
<b>node_exporter</b>	9100	HTTP	Метрики хоста (внутренняя сеть)
<b>postgres_exporter</b>	9187	HTTP	Метрики БД (внутренняя сеть)
<b>sql_exporter</b>	9399	HTTP	Кастомные SQL-метрики (внутренняя сеть)

### Исходящие подключения (egress allow-list)

Источник	Назначение	Порт	Протокол	Назначение
<b>backend/nginx</b>	DNS	53	UDP/TCP	Разрешение имён
<b>nginx/backend</b>	внешние IdP (если есть)	443	HTTPS	OIDC/OpenID Connect

### Особый случай: DataLens (прямой доступ к БД)

Источник	Назначение	Порт	Протокол	Условие безопасности
<b>Yandex DataLens</b>	<b>postgres</b>	<b>5433</b>	TCP	Только read-only учётка

## 2 Требования к инфраструктуре

### 2.1 Платформа и версии

- ОС хостов: Linux x86\_64 (Ubuntu 22.04 LTS / Debian 12 / RHEL 9).
- Контейнеризация: Docker ≥ 24, docker compose v2 (plugin).

## 2.2 Ресурсы и sizing (минимум для PROD)

Узел/роль	vCPU	RAM	Диск (тип)	Примечания
NGINX + oauth2-proxy	1–2	1–2 ГБ	10 ГБ (SSD)	В DMZ; TLS, rate-limit
Backend (FastAPI)	2	2–4 ГБ	10–20 ГБ (SSD)	Горизонтально масштабируется
Postgres/PostGIS	4	8–16 ГБ	200+ ГБ (NVMe)	Отдельный диск под данные + WAL
Prometheus	2	4 ГБ	50–200 ГБ (SSD)	Ретеншн 7–30 дней
Alertmanager	1	1 ГБ	5 ГБ	Небольшой
Экспортеры	1	1 ГБ	5 ГБ	node/postgres/sql exporter
Keycloak	2	4 ГБ	20 ГБ (SSD)	KC_PROXY=edge, /sso за nginx

## 2.3 Сеть и сегментация

- Зоны: DMZ (только NGINX:443) и internal (backend, DB, Keycloak, мониторинг).
- Ingress: извне открыт только 443/TCP → nginx (80/TCP — опционально для 301 → 443).
- Внутренние порты: backend:80, oauth2-proxy:4180, keycloak:8080, postgres:5433, prom:9090, am:9093, exporters: 9100/9187/9399 — без публикации наружу.
- Egress allow-list: DNS:53, NTP:123, S3/объектное:443 (если используется), SMTP:587/465 (если нужно), внешние IdP/SSO: 443 (по необходимости).
- MTU/overlay: MTU 1500 (или 1450 при overlay/VPN). Включить MSS-clamp на периметре при IPsec/WireGuard.

## 2.4 DNS / TLS / Время

- DNS: A/AAAA/CNAME для YOUR\_DOMAIN, внутренние A-записи для postgres, keycloak, prometheus и т.д.
- TLS: 1.2/1.3, валидные цепочки (Let's Encrypt/корп. PKI), HSTS (макс. возраст  $\geq 31536000$ , includeSubDomains), OCSP stapling.
- Ротация сертификатов: автоматическая, предупреждение за  $\geq 14$  дней до истечения.
- NTP: синхронизация ( $\pm 1$  сек) на всех узлах: критично для OIDC/JWT и TLS.

## 2.5 Доступы и безопасность

- Админ-доступ: только через VPN/бастион, SSH по ключам, MFA для привилегий.
- oauth2-proxy (prod)
  - `--cookie-secure=true`
  - `--cookie-httponly=true`
  - `--cookie-samesite=lax`
  - `--pass-access-token=true`

- Keycloak (prod)
  - KC\_PROXY=edge
  - KC\_HOSTNAME=YOUR\_DOMAIN
  - KC\_HTTP\_RELATIVE\_PATH=/sso, доступ в админку только из internal/VPN.
- Логи: без секретов/ПІ; маскирование токенов/паролей на уровне приложения и прокси.

## 2.6 База данных (Postgres/PostGIS)

- Резервное копирование: полный dump ежедневно (pg\_dump/pg\_basebackup), WAL — постоянно; хранение  $\geq 14$ –30 дней.
- Учётки: отдельный пользователь read-only для DataLens; ограниченные схемы/вьюхи; аудит подключений.

## 2.7 Мониторинг и алерты

- Prometheus: доступ из внутренней сети/VPN; scrape backend /metrics, exporters, БД.
- SLI/SLO ориентиры: доступность API  $\geq 99.5\%$ , P95 latency /api/\*  $\leq 500$  мс, 5xx  $\leq 1\%/5$ м.
- Алерты базовые
  - api\_5xx\_rate  $> 1\%$  (5m)
  - latency\_p95  $> 0.5$ s (5m)
  - db\_down==1
  - cert\_expiry\_days  $< 14$
  - oauth2\_errors\_spike
  - container\_restarts  $> 3/10$ м
- Логи/ретеншн: централизованный сбор (Loki/ELK), хранение 30–90 дней; аудит (Keycloak) 180–365 дней.

## 2.8 Интеграция с DataLens

- Встраивание:
  - backend выпускает JWT для iframe
  - ротация ключей JWT
  - ограничение origin (CSP).
- Прямой доступ к БД:
  - read-only учётка
  - запрет публикации ports: наружу.

## 2.9 Разделение сред

- DEV/TEST/PRE-PROD/PROD: изоляция сетей и секретов; доступ к PROD-БД только с PROD-сервисов; запрет кросс-доступа.
- Артефакты: образы с фиксированным sha256 для PROD, единый реестр (private).



## 3 Установка и запуск

### 3.1 DEV

#### 3.1.1 Подготовка окружения

```
git clone https://github.com/limness/mode-float
cd mode-float

cp .env.example .env.dev
```

#### 3.1.2 База данных (PostGIS) и миграции

```
docker compose --profile core up -d postgres
docker compose --profile core run --rm migration
```

#### 3.1.3 backend

```
docker compose --profile core up -d backend
curl -f http://localhost:8073/health
curl -f http://localhost:8073/metrics | head
```

#### 3.1.4 Авторизация (Keycloak + oauth2-proxy + nginx)

```
docker compose --profile auth up -d keycloak
# keycloak dev на http://localhost:8080 (admin/admin по умолчанию из env
# импорт realm: volume ./data/keycloak-import смонтирован, команда '--
import-realm' уже задана

docker compose --profile auth up -d oauth2-proxy nginx
# dev callback настроен как http://localhost/oauth2/callback
```

#### 3.1.5 Мониторинг

```
docker compose --profile monitoring up -d
# prometheus: http://localhost:9090
# alertmanager: http://localhost:9093
```

## 3.2 PROD

### 3.2.1 Предварительные требования

- Подготовлен сервер(а) PROD: dmz (nginx) и internal (backend, postgres, keycloak, monitoring) или один хост с двумя docker-сетями.
- Открыт наружу только 443/TCP → nginx (80/TCP опц. для редиректа).
- Выпущены TLS-сертификаты для YOUR\_DOMAIN (Let's Encrypt или корпоративные).
- Созданы секреты/учётки
  - POSTGRES\_PASSWORD, POSTGRES\_URI для бэкенда.
  - Keycloak
    - \* KC\_HOSTNAME=YOUR\_DOMAIN
    - \* KC\_PROXY=edge
    - \* realm
    - \* confidential-клиент (client\_id/secret)
  - oauth2-proxy
    - \* OAUTH2\_CLIENT\_ID
    - \* OAUTH2\_CLIENT\_SECRET
    - \* OAUTH2\_COOKIE\_SECRET
  - Postgres read-only пользователь для DataLens
- .env.prod создан из .env.example

### 3.2.2 Подготовка конфигов

```
cp .env.example .env.prod
```

### 3.2.3 Настройка Nginx

Замените в конфиге:

- server\_name — ваш домен (например, example.ru www.example.ru);
- ssl\_certificate / ssl\_certificate\_key — настоящие пути к сертификатам.

### 3.2.4 Настройка Keycloak

1. Войти в Keycloak → создать **Realm**.
2. Clients → Create client:
  - **Client type**: OpenID Connect
  - **Client ID**: oauth2-проху (или своё)
  - **Name**: по желанию
3. Next:
  - **Root URL**:
  - **Valid redirect URIs**: ВАШ\_ДОМЕН>/oauth2/callback
  - **Web origins**:

- **Home URL** (опц.):
4. Next:
- **Client authentication**: ON (тип клиента: confidential)
  - **Standard Flow**: ON
  - Сохранить.
5. На вкладке **Credentials** скопировать **Client secret** — он нужен oauth2-проху.
- Protocol Mappers** (клиент → Mappers):
- Добавьте мапперы, чтобы нужные поля попадали в токены:
- email:
    - **Mapper type**: User Property
    - **Property**: email
    - **Token Claim Name**: email
    - **Add to ID token**: ON, **Add to access token**: ON
  - preferred\_username:
    - **Mapper type**: User Property
    - **Property**: username
    - **Token Claim Name**: preferred\_username
    - **Add to ID token**: ON, **Add to access token**: ON
  - (опц.) groups:
    - **Mapper type**: Group Membership
    - **Token Claim Name**: groups
    - (по желанию) Full group path

### 3.2.5 Запуск

```
make all-prod
```

## 3.3 DataLens

### 3.3.1 Предпосылки и безопасность

- Прямой доступ DataLens к БД: по read-only учётке.
- Минимизируйте права: отдельная роль, доступ к нужным схемам/вьюхам, аудит подключений.

### 3.3.2 Подготовка

1. Требуется аккаунт в Yandex Cloud / DataLens с доступом к коллекциям.
2. Дождитесь готовности каталога/облака в УС.

### 3.3.3 Импорт воркбука

- Скачайте файл воркбука

- Зайдите на страницу с Воркбуками
- В правом верхнем углу нажмите кнопку Создать
- Импортируйте ранее скаченный файл
- Заполните поля названия и описания

### 3.3.4 Подключение к БД

- Внутри вашего воркбука → вкладка Подключения → выбрать flight-database-connection.
- Заполнить
  - Каталог: ваш текущий
  - Хост: IP/имя сервера Postgres
  - Порт: 5433
  - Пользователь и пароль
- Сохранить и проверить соединение.

### 3.3.5 Встраивание дашборда в сайт (через JWT)

Для встраивания DataLens в iframe бэкенд выдаёт JWT, который DataLens принимает для SSO/-доступа.

TODO

## 4 Конфигурация (DEV + PROD)

### 4.1 Переменные окружения

Переменная	Значение по умолчанию	Описание
APP_TITLE	Fly Potato	Название приложения
APP_VERSION	0.1.0	Версия приложения
APP_DEBUG	False	Режим отладки (True/False)
APP_PROMETHEUS_HOST	0.0.0.0	Хост, на котором запускается приложение
APP_PROMETHEUS_PORT	8000	Порт приложения
POSTGRES_URI	–	URI подключения к базе данных
GRAFANA_CLOUD_USERNAME		Имя пользователя Grafana Cloud
GRAFANA_CLOUD_TOKEN		Токен Grafana Cloud для метрик
DATA_SOURCE_NAME	–	Имя/ссылка источника метрик

## 5 Зависимости (DEV + PROD)

Библиотека	Минимальная версия	Назначение
<b>python</b>	^3.12	-
<b>sqlalchemy</b>	^2.0.39	ORM и конструктор SQL-запросов
<b>alembic</b>	^1.15.1	Инструмент миграций для SQLAlchemy
<b>asyncpg</b>	^0.30.0	Асинхронный драйвер PostgreSQL
<b>pydantic-settings</b>	^2.8.1	Управление настройками через Pydantic
<b>greenlet</b>	^3.1.1	Лёгкие сопрограммы (используются SQLAlchemy для «зелёных потоков»)
<b>pydantic</b>	^2.11.7	Валидация данных и создание моделей
<b>geopy</b>	^2.4.1	Геокодирование и географические вычисления
<b>numpy</b>	^2.3.1	Научные вычисления и работа с массивами
<b>colorlog</b>	^6.9.0	Раскрашенный логгинг
<b>httpx</b>	^0.28.1	Асинхронный HTTP-клиент
<b>prometheus-client</b>	^0.22.1	Метрики и мониторинг Prometheus
<b>aiohttp</b>	^3.12.14	Асинхронный HTTP-клиент/сервер
<b>fastapi</b>	^0.117.1	Веб-фреймворк для создания REST/GraphQL API
<b>uvicorn</b>	^0.37.0	ASGI-сервер для запуска FastAPI и других ASGI-приложений
<b>pandas</b>	^2.3.2	Анализ и обработка табличных данных
<b>openpyxl</b>	^3.1.5	Работа с Excel-файлами (XLSX)
<b>geoalchemy2</b>	^0.18.0	Геопространственные расширения для SQLAlchemy/PostGIS
<b>pyjwt</b>	^2.10.1	Работа с JSON Web Token (JWT)
<b>python-multipart</b>	^0.0.20	Парсинг multipart/form-data (загрузка файлов и форм в FastAPI)

## 6 Резервное копирование и восстановление

### 6.1 DEV

```
PGPASSWORD=postgres pg_dump -h localhost -p 5433 -U postgres -d float-
mode -Fc \
-f ./data/postgres-backups/float_$(date +%Y%m%d).dump
```

### 6.2 PROD

- Ежедневно: полный дамп БД `pg_dump -Fc` или `pg_basebackup`.
- Постоянно: архив WAL (для PITR).

### 6.2.1 Команды (пример)

```
# Полный дамп ежедневно  ()
PGPASSWORD=*** pg_dump -h postgres -U app_user -d float_mode -Fc \
    -f /backups/float_$(date +%Y%m%d).dump

# Включить WAL архив- (postgresql.conf)
# wal_level=replica
# archive_mode=on
# archive_command='test ! -f /wal-archive/%f && cp %p /wal-archive/%f'
```

### 6.2.2 Восстановление (PITR, кратко)

```
# Остановить приложение
docker compose stop backend

# Пересоздать БД из дампа
dropdb -h postgres -U app_user float_mode || true
createdb -h postgres -U app_user float_mode
pg_restore -h postgres -U app_user -d float_mode -j 4 /backups/
    float_YYYYMMDD.dump

# если( PITR) - recovery.conf/standby.signal с recovery_target_time=...
# и доступом к /wal-archive, затем старт postgres

# Запустить приложение
docker compose start backend
```

### 6.2.3 Политики

- Ретеншн: дампы 14–30 дней, WAL 7–14 дней.
- Ежедневно: тестовое восстановление на стенде.

## 7 Мониторинг и оповещения

### 7.1 Prometheus targets (пример)

```
scrape_configs:
  - job_name: 'backend'
    static_configs: [ { targets: [ 'backend:8000' ] } ]
    metrics_path: /metrics

  - job_name: 'postgres_exporter'
```

```
static_configs: [ { targets: [ 'postgres_exporter:9187' ] } ]

- job_name: 'node_exporter'
  static_configs: [ { targets: [ 'node_exporter:9100' ] } ]

- job_name: 'sql_exporter'
  static_configs: [ { targets: [ 'sql_exporter:9399' ] } ]
```

## 7.2 Базовые правила (alert.rules.yaml)

- InstanceDown для backend и БД.
- Ошибки аутентификации (рост 401/403 на nginx в PROD; в DEV можно смотреть вручную по логам).
- P95 latency для /api/\* (в DEV — информативно).

# 8 Доступы и безопасность

## 8.1 DEV

- Всё доступно локально по HTTP; внешнюю публикацию DEV не делать.
- Ключи/секреты в .env.dev держать только локально; не коммитить.
- Для отладки ролей — использовать группы/клеймы в Keycloak (preferred\_username, groups).

## 8.2 PROD

- Периметр: наружу открыт только 443/TCP → nginx (опц. 80→443 редирект). Backend/DB/-мониторинг без ports:.
- Доступ админов: только через VPN/бастион, SSH по ключам, MFA для привилегированных учёток.
- Секреты: в Vault/SSM/Pass; на узлах — только runtime (.env.prod с правами 600), ротация ≥ каждые 90 дней.
- oauth2-проху, включены
  - cookie-secure
  - httponly
  - samesite=lax
  - прокидываем Authorization и access-token
- Keycloak: работает за /sso с KC\_PROXY=edge, админка закрыта сетевом (только internal/VPN).

## 9 Эксплуатационные регламенты

### 9.1 DEV

#### 9.1.1 Пересоздать БД

```
docker compose --profile core down -v
rm -rf ./data/postgres
docker compose --profile core up -d postgres
docker compose --profile core run --rm migration
```

#### 9.1.2 Горячая перезагрузка backend

```
docker compose --profile core up -d --build backend
```

#### 9.1.3 Логи быстрого доступа

```
docker compose logs -f backend
docker compose logs -f postgres
docker compose logs -f oauth2-proxy nginx keycloak
```

### 9.2 PROD

- Ежедневно: короткий smoke через периметр (/health), взгляд на 5xx и P95; проверка свободного места БД и срока TLS (>14 дней).
- Еженедельно: тест восстановления бэкапа на стенде (в т.ч. PITR); проверка актуальности алертов и on-call контактов.
- Релиз: пины образов по sha256; порядок — бэкап → pull/up → миграции → smoke; откат — теги назад + alembic downgrade .
- Журналы/РП: маскирование секретов; централизованный сбор, ретеншн согласно политике (коротко сослаться на раздел про мониторинг).
- Инциденты: быстрая локализация (rate-limit/feature-flag/отключение интеграции), затем rollback/-фикс; пост-мортем и САРА в регламентные сроки.