

Дикасто

Инструкция по установке

Листов: 20

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ ТЕРМИНОВ И СОКРАЩЕНИЙ.....	3
1. ОБЩИЕ СВЕДЕНИЯ.....	4
2. СОСТАВ ДИСТРИБУТИВА.....	5
2.1. Файлы дистрибутива.....	5
2.2. Компоненты системы.....	6
2.3. Альтернатива: сборка из исходных кодов	6
3. ТРЕБОВАНИЯ К АППАРАТНОМУ И ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ	7
3.1. Минимальные системные требования.....	7
3.1.1. Сервер (один контур: платформа + сервисы распознавания).....	7
3.2. Программное обеспечение	7
3.2.1. Сервер (устанавливается администратором)	7
3.2.2. Внешние сервисы (в поставку не входят).....	8
3.2.3. Рабочее место пользователя.....	8
4. ПОРЯДОК УСТАНОВКИ И НАСТРОЙКИ.....	9
4.1. Распаковка дистрибутива	9
4.2. Настройка домашнего каталога системы	10
4.3. Настройка подключения к базе данных	10
4.3.1. Встроенный PostgreSQL (по умолчанию).....	10
4.3.2. Внешний сервер PostgreSQL.....	11
4.4. Конфигурация сервисов.....	13
4.4.1. Платформа: .env.....	13
4.4.2. Платформа: конфигурация бэкенда	13
4.4.3. Сервис извлечения данных (docex-service)	14
4.4.4. OCR-сервис (docex)	15
4.4.5. Согласование параметров	15
4.5. Запуск программного обеспечения.....	15
5. ПРОВЕРКА РАБОТОСПОСОБНОСТИ.....	18
5.1. Состояние контейнеров	18
5.2. Проверка компонентов	18
5.3. Проверка через браузер	18
5.4. Проверка распознавания документов (при наличии LLM-шлюза)	19
5.5. Типовые проблемы при первом запуске	19
ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ.....	21

ПЕРЕЧЕНЬ ТЕРМИНОВ И СОКРАЩЕНИЙ

Термины и сокращения, используемые в документе, приведены в таблице 1.

Таблица 1 – Список терминов и сокращений

Термин/сокращение	Определение/расшифровка
БД	База данных
ГБ	Гигабайт
МБ	Мегабайт
ОЗУ	Оперативное запоминающее устройство
ОС	Операционная система
ПО	Программное обеспечение

1. ОБЩИЕ СВЕДЕНИЯ

Документ описывает установку системы «с нуля» на подготовленный сервер: от распаковки дистрибутива до проверки работоспособности.

2. СОСТАВ ДИСТРИБУТИВА

Система поставляется в виде набора Docker-образов и исходных каталогов трёх подсистем. Все компоненты серверной части работают в контейнерах; на сервер устанавливаются только Docker Engine и Docker Compose. Прикладное ПО на рабочие места пользователей не устанавливается.

2.1. Файлы дистрибутива

Перечень файлов дистрибутива представлен в таблице 2.

Таблица 2 – Описание файлов дистрибутива

№	Файл / каталог	Содержимое	Обязательность
1	efrsb-images- <version>.tar.gz	Архив Docker-образов (docker save): 1. Собственные образы: – efrsb/backend, – efrsb/frontend, – efrsb/docex-service, – efrsb/docex-ocr 2. Сторонние: – nginx:stable-alpine, – postgres:16-alpine, – quay.io/keycloak/keycloak:25.0.2, – rabbitmq:3-management	обязательно
2	efrsb-images- <version>.sha256	Контрольная сумма архива образов	обязательно
3	bankruptcy- platform/	Исходный каталог платформы: docker-compose.yml, docker-compose.stage.yml, .env.stage, Makefile, infra/ (nginx, realm и темы Keycloak, init-скрипт БД, образ Logdy infra/logdy/logdy-image.tar.gz, оверлей Filebeat), backend/src/main/resources/application- test.yml (монтируемая конфигурация бэкенда), docs/	обязательно
4	docex-service/	Исходный каталог сервиса извлечения данных: docker- compose.yml, config/application.yml.example	обязательно
5	docex/	Исходный каталог OCR-сервиса: docker- compose.yml	обязательно
6	ocr-models.tar.gz	Архив тома paddleocr_models с предзагруженными моделями PaddleOCR	обязательно для контура без доступа в интернет; иначе модели скачиваются при первом обращении

№	Файл / каталог	Содержимое	Обязательность
7	docs/	Эксплуатационная документация: настоящая инструкция, требования к ПО, руководство администратора, ранбуки, release notes	обязательно

где <version> – номер релиза (например, 0.12), а не latest. Все образы и архивы одного релиза хранятся вместе с контрольными суммами.

2.2. Компоненты системы

Описание компонентов системы дано в таблице 3.

Таблица 3 – Описание компонентов системы

Подсистема	Компонент	Контейнер	Порт внутри сети	Публикуется наружу
bankruptcy-platform	nginx – единая точка входа	efrsb-nginx	80	80
	frontend (Next.js)	efrsb-frontend	3000	нет
	backend (Spring Boot, вкл. print-service)	efrsb-backend	3080	нет
	PostgreSQL 16	efrsb-postgres	5432	PG_EXTERNAL_PORT (служебный)
	Keycloak 25	efrsb-keycloak	8080	нет (через nginx /auth/)
	Logdy – просмотр логов (stage-оверлей)	efrsb-logdy	8080	нет (через nginx /logs/)
docex-service	сервис извлечения данных (Java)	efrsb-docex-service	8080	DOCEX_HTTP_PORT (служебный)
	RabbitMQ (используется и платформой)	efrsb-docex-rabbitmq	5672, 15672	5672, 15672 (служебные)
docex	OCR-сервис (Python/FastAPI, PaddleOCR)	efrsb-docex-ocr	8001	нет

Все контейнеры работают в общей docker-сети efrsb-network. Внешняя зависимость, в дистрибутив не входящая, – LLM-шлюз (Ollama либо иной сервис с OpenAI-совместимым API); без него не работает только AI-распознавание документов и корреспонденции.

2.3. Альтернатива: сборка из исходных кодов

Если сервер имеет доступ к интернету (или к корпоративным зеркалам Maven Central, npm registry, PyPI, Docker Hub, quay.io), образы можно собрать на месте – тогда архив образов (п. 1, 2 таблицы 2) не нужен, а вместо исходных каталогов клонируются репозитории bankruptcy-platform, docex-service, docex.

3. ТРЕБОВАНИЯ К АППАРАТНОМУ И ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

3.1. Минимальные системные требования

3.1.1. Сервер (один контур: платформа + сервисы распознавания)

Минимальные системные требования для сервера представлены в таблице 4.

Таблица 4 – Минимальные системные требования для сервера

Ресурс	Минимум	Рекомендуется	Примечание
Процессор	6 vCPU x86-64	8 vCPU	2 vCPU – под OCR; распознавание идёт на CPU, GPU не используется
ОЗУ	12 ГБ	16 ГБ	PostgreSQL shared_buffers 1 ГБ, Keycloak 512 МБ, backend ~2 ГБ, OCR до 4 ГБ (mem_limit: 4g), docex-service ~1 ГБ, RabbitMQ ~0,5 ГБ
Диск	40 ГБ	от 100 ГБ	Образы (~5 ГБ), тома БД, загруженные документы (uploads_data, docex_uploads), логи. Объём растёт с числом загруженных документов
Сеть	100 Мбит/с	1 Гбит/с	Пользователям достаточно порта 80; в рантайме нужен доступ к LLM-шлюзу и (при использовании) к SMTP и Elasticsearch

Ресурсы указаны на один контур. Сборочный хост (п. 2.3) дополнительно требует ~10 ГБ под кэши Maven/npm/BuildKit.

3.1.1.1. Рабочее место пользователя

Минимальные системные требования для рабочего места пользователя представлены в таблице 5.

Таблица 5 – Минимальные системные требования для рабочего места пользователя

Ресурс	Требование
ОЗУ	от 4 ГБ
Экран	от 1366×768 (рекомендуется 1920×1080)
Канал до сервера	от 5 Мбит/с

3.2. Программное обеспечение

3.2.1. Сервер (устанавливается администратором)

Описание программного обеспечения для сервера дано в таблице 6.

Таблица 6 – Описание программного обеспечения для сервера

№	Наименование	Версия	Назначение
1	ОС Linux x86-64	ядро 5.4+	Проверенные: Ubuntu Server 22.04/24.04 LTS, Debian 12, RHEL / AlmaLinux / Rocky Linux 9. Astra Linux SE 1.7+, РЕД ОС 7.3+, ALT Server 10 – при наличии Docker Engine 24+
2	Docker Engine	24.0+	Среда исполнения контейнеров
3	Docker Compose	v2.24.4+ (команда docker compose)	Оркестрация; теги !reset!/override в оверлеях (п. 4.3.2, п. 4.5)
4	make, tar, gzip, sha256sum	штатные	Цели Makefile, распаковка и проверка дистрибутива
5	git	любая	Только при установке из репозитория (п. 2.3)

Windows и macOS в качестве сервера не поддерживаются: стендовые оверлеи используют bind-mount абсолютных путей Linux.

В составе образов (отдельно не устанавливается): nginx 1.30, PostgreSQL 16.15, Keycloak 25.0.2, Temurin JRE 21, Node.js 20, RabbitMQ 3, Python 3.11 + PaddleOCR. Версии сторонних образов зафиксированы digest-ом (tag@sha256:...).

3.2.2. Внешние сервисы (в поставку не входят)

Описание программного обеспечения для внешних сервисов дано в таблице 7.

Таблица 7 – Описание программного обеспечения для внешних сервисов

Сервис	Требование	Без него
LLM-шлюз	HTTP-доступ с сервера; OpenAI-совместимый путь (/v1); Bearer-токен	Недоступно AI-распознавание документов
SMTP-сервер	Хост, порт, учётные данные, STARTTLS	Не отправляются e-mail-уведомления
Elasticsearch + Kibana 8.x	Только при использовании оверлея Filebeat	Логи остаются в файлах и Logdy

3.2.3. Рабочее место пользователя

Перечень ПО для рабочего места пользователя:

- браузер (Chrome/Chromium 90+, Edge 90+, Яндекс Браузер 22+, Firefox 90+ (в т.ч. ESR), Safari 15+;
- включённые JavaScript и cookies;
- просмотрщик PDF и редактор XLSX.

Примечание. Internet Explorer 11 и Edge Legacy не поддерживаются.

4. ПОРЯДОК УСТАНОВКИ И НАСТРОЙКИ

Последовательность шагов:

1. Распаковка дистрибутива – образы, исходные каталоги, модели OCR
2. Домашний каталог – /opt/Arbitrag, каталог логов, docker-сеть
3. Подключение к базе данных – встроенный PostgreSQL или внешний сервер
4. Конфигурация сервисов – .env платформы, YAML бэкенда, LLM-шлюз, OCR
5. Запуск – OCR → docex-service (+RabbitMQ) → платформа
6. Проверка работоспособности

Все команды выполняются на сервере от пользователя, входящего в группу docker (либо через sudo).

4.1. Распаковка дистрибутива

Шаг 1. Проверить и загрузить образы

```
cd /path/to/distributive
sha256sum -c efrsb-images-<version>.sha256          # должно вывести: ОК
docker load -i efrsb-images-<version>.tar.gz
docker image ls | grep -E 'efrsb|nginx|postgres|keycloak|rabbitmq'
```

В списке должны присутствовать четыре образа efrsb/*:<version> и четыре сторонних образа (п. 1 табл. 2). Архитектура образов должна совпадать с архитектурой сервера (linux/amd64).

Шаг 2. Разместить исходные каталоги

```
sudo mkdir -p /opt/Arbitrag
sudo chown "$USER":"$USER" /opt/Arbitrag
cp -r bankruptcy-platform docex-service docex /opt/Arbitrag/
```

При установке из репозитория (п. 2.3) вместо копирования – git clone трёх репозитория в /opt/Arbitrag/ с переключением на тег релиза.

Шаг 3. Загрузить модели OCR (только для контура без доступа в интернет; при наличии доступа модели скачиваются при первом обращении к сервису):

```
docker volume create paddleocr_models
docker run --rm -v paddleocr_models:/data -v "$PWD":/backup alpine \
  tar xzf /backup/ocr-models.tar.gz -C /data
```

Шаг 4. Загрузить образ Logdy (нужен для stage-оверлея; лежит в исходном каталоге платформы):

```
cd /opt/Arbitrag/bankruptcy-platform
make logdy-image          # docker load -i infra/logdy/logdy-image.tar.gz, если образа ещё нет
```

4.2. Настройка домашнего каталога системы

Домашний каталог системы – /opt/Arbitrag. Путь зафиксирован в docker-compose.stage.yml и infra/filebeat/docker-compose.filebeat.yml платформы (bind-mount каталога логов), поэтому при размещении в другом месте оба файла правятся согласованно.

Целевая структура:

```

/opt/Arbitrag/
├── bankruptcy-platform/                                # платформа
│   ├── .env                                           # ← создаётся на шаге 3.4.1
│   ├── docker-compose.yml
│   ├── docker-compose.stage.yml
│   └── docker-compose.offline.yml                    # ← создаётся на шаге 3.5 (только для
установки из образов)
│   ├── Makefile
│   ├── backend/logs/                                # ← каталог логов бэкенда (bind-mount)
│   └── backend/src/main/resources/application-test.yml # монтируемая
конфигурация бэкенда
├── infra/                                             # nginx, keycloak, postgres/init, logdy,
filebeat
├── docex-service/                                    # сервис извлечения данных
│   ├── .env                                           # ← DOCEX_HTTP_PORT (опционально)
│   ├── config/application.yml                        # ← создаётся из application.yml.example
│   ├── docker-compose.yml
│   └── docex/                                         # OCR-сервис
└── docker-compose.yml

```

Данные системы хранятся в именованных docker-томах (каталог /var/lib/docker/volumes/ на сервере): postgres_data (основная БД и БД Keycloak), uploads_data (загруженные документы платформы), docex_uploads (файлы сервиса распознавания), rabbitmq_data, paddleocr_models.

Раздел с /var/lib/docker должен обеспечивать объём диска из п. 3.1.

Шаг 1. Создать каталог логов бэкенда:

```

sudo mkdir -p /opt/Arbitrag/bankruptcy-platform/backend/logs
sudo chmod 777 /opt/Arbitrag/bankruptcy-platform/backend/logs # бэкенд пишет из контейнера

```

Каталог читают Logdy (просмотр логов по /logs/) и Filebeat (отгрузка в ELK). Ротацию файлов в нём настраивать штатным logrotate.

Шаг 2. Создать общую docker-сеть (один раз на сервер):

```

docker network create efrsb-network

```

Все три compose-проекта объявляют сеть как external; без неё запуск падает с network efrsb-network declared as external, but could not be found.

4.3. Настройка подключения к базе данных

Система использует PostgreSQL 16 с двумя базами: efrsb (данные платформы) и keycloak (данные сервера аутентификации). Поддерживаются два варианта размещения.

4.3.1. Встроенный PostgreSQL (по умолчанию)

Контейнер efrsb-postgres входит в состав docker-compose.yml платформы и настроен полностью: max_connections=300, shared_buffers=1GB, данные – в томе postgres_data. При первой инициализации пустого тома выполняется infra/postgres/init/01-init.sh, который создаёт роли и базу Keycloak.

Учётные записи, задействованные в подключении описаны в таблице 8.

Таблица 8 – Учётные записи, задействованные в подключении

Роль	Пароль по умолчанию	Назначение	Где задаётся
efrsb	efrsb_super_secret_2026	Владелец БД efrsb; под ним выполняются миграции Flyway при старте бэкенда	POSTGRES_PASSWORD, SPRING_FLYWAY_PASSWORD в docker-compose.yml
app_user	app_user_secret_2026	Ограниченная роль приложения (без superuser); под ней работает бэкенд, действуют RLS-политики	SPRING_DATASOURCE_PASSWORD в docker-compose.yml; CREATE ROLE в 01-init.sh
keycloak	keycloak_pass	Владелец БД keycloak	KC_DB_PASSWORD в docker-compose.yml; CREATE USER в 01-init.sh

Для промышленного контура пароли по умолчанию **заменяются до первого запуска** – init-скрипт выполняется только на пустом томе, после него роли уже созданы со старыми паролями. Порядок:

1. В docker-compose.yml заменить POSTGRES_PASSWORD и SPRING_FLYWAY_PASSWORD (одно и то же значение), SPRING_DATASOURCE_PASSWORD, KC_DB_PASSWORD.
2. В infra/postgres/init/01-init.sh заменить пароли в CREATE ROLE app_user ... PASSWORD '...' и CREATE USER keycloak WITH PASSWORD '...' на те же значения.
3. Если том postgres_data уже создавался (пробный запуск) – удалить его: docker compose down -v в каталоге платформы. Команда удаляет все данные системы.

Внешний порт БД задаётся переменной PG_EXTERNAL_PORT в .env (п. 4.4.1) и нужен только для административного доступа; на защищённом контуре его закрывают firewall.

4.3.2. Внешний сервер PostgreSQL

Допускается PostgreSQL 16 (в т.ч. Postgres Pro как совместимая реализация). Понижение мажорной версии и СУБД иного производителя не допускаются: RLS-политики, реестр политик и схема хранения сессий рассчитаны на PostgreSQL 16.

Требования к серверу СУБД

- Кодировка баз UTF8; max_connections не ниже 150 (пул бэкенда – 50 соединений, плюс Keycloak и административные сессии).
- Роль-владелец БД efrsb должна иметь право создавать расширения uuid-ossr и pg_trgm (в PostgreSQL 13+ они «доверенные» – достаточно привилегии CREATE на базу).
- Сетевая доступность с сервера контейнеров (pg_hba.conf, firewall) по порту 5432.

Шаг 1. Создать роли и базы – выполнить на сервере СУБД под суперпользователем эквивалент infra/postgres/init/01-init.sh:

```
CREATE ROLE efrsb LOGIN PASSWORD '<пароль-efrsb>';
CREATE DATABASE efrsb OWNER efrsb ENCODING 'UTF8';
```

```

CREATE ROLE app_user LOGIN PASSWORD '<пароль-app_user>' NOSUPERUSER NOCREATEDB NOCREATEROLE;
\connect efrsb
GRANT USAGE ON SCHEMA public TO app_user;
ALTER DEFAULT PRIVILEGES FOR ROLE efrsb IN SCHEMA public
    GRANT SELECT, INSERT, UPDATE, DELETE ON TABLES TO app_user;
ALTER DEFAULT PRIVILEGES FOR ROLE efrsb IN SCHEMA public
    GRANT USAGE, SELECT ON SEQUENCES TO app_user;

CREATE USER keycloak WITH PASSWORD '<пароль-keycloak>';
CREATE DATABASE keycloak OWNER keycloak ENCODING 'UTF8';
\connect keycloak
GRANT ALL ON SCHEMA public TO keycloak;

```

Права app_user выдаются через ALTER DEFAULT PRIVILEGES от имени efrsb: таблицы создаются миграциями Flyway под efrsb, и привилегии на них появляются автоматически.

Шаг 2. Переключить платформу на внешний сервер – создать в каталоге платформы оверлей docker-compose.external-db.yml (теги !reset!/override – Docker Compose 2.24.4+):

```

services:
  postgres:
    profiles: ["builtin-db"]          # встроенный контейнер не запускается
  keycloak:
    depends_on: !reset null
    environment:
      KC_DB_URL: jdbc:postgresql://<db-host>:5432/keycloak
      KC_DB_PASSWORD: <пароль-keycloak>
  backend:
    depends_on: !override
    keycloak:
      condition: service_healthy
    environment:
      SPRING_DATASOURCE_URL: jdbc:postgresql://<db-host>:5432/efrsb
      SPRING_DATASOURCE_PASSWORD: <пароль-app_user>
      SPRING_FLYWAY_URL: jdbc:postgresql://<db-host>:5432/efrsb
      SPRING_FLYWAY_PASSWORD: <пароль-efrsb>

```

и добавить его в COMPOSE_FILE в .env (п. 4.4.1):

```
COMPOSE_FILE=docker-compose.yml:docker-compose.stage.yml:docker-compose.external-db.yml
```

Если сервер СУБД – на том же хосте вне Docker, обращаться к нему по адресу host.docker.internal с extra_hosts: ["host.docker.internal:host-gateway"] у сервисов backend и keycloak.

Схема БД в обоих вариантах создаётся автоматически: миграции Flyway (V001...V158) выполняются при первом старте бэкенда под ролью efrsb. Ручного создания таблиц не требуется.

4.4. Конфигурация сервисов

4.4.1. Платформа: .env

```
cd /opt/Arbitrag/bankruptcy-platform
cp .env.stage .env
```

Таблица 9 – Описание конфигурации

Переменная	Значение по умолчанию	Что задать
COMPOSE_FILE	docker-compose.yml:docker-compose.stage.yml	Добавить :docker-compose.offline.yml при установке из образов и :docker-compose.external-db.yml при внешней БД
PG_EXTERNAL_PORT	5433	Свободный порт хоста для административного доступа к БД
NOTIFICATIONS_RABBITMQ_URI	amqp://guest:guest@efrsb-docex-rabbitmq:5672	Менять только вместе с учётными данными RabbitMQ в docex-service/docker-compose.yml
NEXT_PUBLIC_APP_ENV	production	Оставить. Build-time: изменение требует пересборки образа фронтенда
NEXT_PUBLIC_API_URL	пусто	Оставить пустым — фронтенд делает относительные вызовы API. Build-time
BASE_URL, APP_BASE_URL	не заданы	Не задавать: система работает по любому адресу (IP, DNS-имя) без правки конфигов. Задать только для жёсткой привязки к адресу — тогда дополнительно раскомментировать KC_HOSTNAME в docker-compose.yml и сузить redirectUri в realm Keycloak

4.4.2. Платформа: конфигурация бэкенда

Бэкенд читает YAML-файл, монтируемый в контейнер как /config/application.yml: backend/src/main/resources/application-test.yml. Правка файла применяется рестартом контейнера, без пересборки. Значения по умолчанию рассчитаны на поставочный realm Keycloak и поставочный конфиг docex-service. Параметры которые нужно менять при установке перечислены в таблице 10.

Таблица 10 – Описание конфигурации

Параметр	По умолчанию	Когда менять
spring.security.oauth2.client.registration.keycloak.client-secret	backend-secret-2026	Вместе с секретом клиента efrsb-backend в infra/keycloak/realm-efrsb.json (промышленный контур)

Параметр	По умолчанию	Когда менять
docex.api-key	key-test-002	Вместе с картой api.keys в docex-service/config/application.yml (§3.4.3)
server.servlet.session.cookie.secure	закомментировано	Включить (true) при публикации контура по HTTPS
management.tracing.sampling.probability	1.0	На промышленном контуре снизить (например, 0.1)

SMTP для e-mail-уведомлений задаётся переменными окружения контейнера backend (добавить в environment через оверлей либо в docker-compose.yml): NOTIFICATIONS_SMTP_HOST, NOTIFICATIONS_SMTP_PORT, NOTIFICATIONS_SMTP_USER, NOTIFICATIONS_SMTP_PASSWORD, NOTIFICATIONS_SMTP_STARTTLS_REQUIRED, NOTIFICATIONS_EMAIL_FROM. Без них уведомления по почте не отправляются, остальная функциональность не затрагивается.

Keycloak. Учётная запись администратора задаётся в docker-compose.yml (KEYCLOAK_ADMIN / KEYCLOAK_ADMIN_PASSWORD, по умолчанию admin / admin) – заменить до первого запуска. Realm efrsb импортируется из infra/keycloak/realm-efrsb.json только при пустой БД Keycloak; дальнейшие изменения – через admin-консоль /auth/. Поставочный realm содержит демонстрационные учётные записи (test_au и др.) и шаблонные redirectUris (http://*, https://*); перед вводом в эксплуатацию их заменяют.

4.4.3. Сервис извлечения данных (docex-service)

```
cd /opt/Arbitrag/docex-service
cp config/application.yml.example config/application.yml
```

В config/application.yml заполнить параметры, перечисленные в таблице 11.

Таблица 11 – Перечень параметров

Параметр	Значение
ollama.base-url	Адрес LLM-шлюза, OpenAI-совместимый путь (http://<host>:<port>/v1 или .../ollama/v1). Если шлюз на том же сервере вне Docker — http://host.docker.internal:<port>/... (extra_hosts в compose уже настроен)
ollama.model	Имя модели на шлюзе
ollama.token	Bearer-токен шлюза. Секрет, имеет срок действия
ollama.temperature, ollama.top-p	0 и 1.0 — детерминированная генерация; иные значения ломают разбор ответов модели
ocr.service-url	http://efrsb-docex-ocr:8001 (оставить)
api.keys	Карта «ключ → потребитель». Ключ платформы должен совпадать с docex.api-key бэкенда (§3.4.2). Ключи для разработки из шаблона удалить

Параметр	Значение
logging.level.com.docsex	INFO (в шаблоне DEBUG)

Каталог config/ монтируется в контейнер только для чтения; изменение файла применяется командой `docker compose restart docex-service`.

Если порт 8080 на сервере занят – создать .env рядом с docker-compose.yml:

DOCEX_HTTP_PORT=8088

4.4.4. OCR-сервис (docex)

Настраивается переменными окружения в docker-compose.yml каталога docex; значения по умолчанию пригодны для запуска. Значимые параметры: OCR_MAX_PAGES (сколько первых страниц распознавать; 0 – все), OCR_RENDER_DPI (разрешение растеризации), OCR_ENGINE (v3/v5), mem_limit (потолок памяти контейнера, 4g – держать ниже объёма ОЗУ сервера с запасом на остальные сервисы).

4.4.5. Согласование параметров

При изменении любого значения проверить парное согласно таблицы 12.

Таблица 12 – Перечень параметров

Параметр	Должен совпадать с
docex.api-key (бэкенд)	ключом в api.keys (docex-service)
client-secret (бэкенд)	секретом клиента efrsb-backend в realm Keycloak
SPRING_FLYWAY_PASSWORD, SPRING_DATASOURCE_PASSWORD, KC_DB_PASSWORD	паролями ролей efrsb, app_user, keycloak в БД
NOTIFICATIONS_RABBITMQ_URI, SPRING_RABBITMQ_HOST/PORT	учётными данными и именем контейнера RabbitMQ в docex-service/docker-compose.yml
SPRING_SERVLET_MULTIPART_MAX_* (500 МБ)	client_max_body_size в infra/nginx/default.conf
LOGGING_FILE_PATH (/var/log/app)	bind-mount каталога логов в docker-compose.stage.yml

4.5. Запуск программного обеспечения

Порядок запуска важен: бэкенд платформы при старте подключается к RabbitMQ, который поднимается вместе с docex-service, а docex-service обращается к OCR-контейнеру.

Шаг 0 (только при установке из образов). Запретить сборку и обращение к registry.

По умолчанию compose собирает образ сам, если не находит подходящий, а make up вызывает docker compose up --build. Чтобы compose брал загруженные образы, в каждом из трёх каталогов создаётся оверлей docker-compose.offline.yml:

```
# /opt/Arbitrag/bankruptcy-platform/docker-compose.offline.yml
services:
  backend:
    image: efrsb/backend:<version>
    pull_policy: never
    build: !reset null
  frontend:
    image: efrsb/frontend:<version>
    pull_policy: never
    build: !reset null
  nginx:
    pull_policy: never
  postgres:
    pull_policy: never
  keycloak:
    pull_policy: never
```

```
# /opt/Arbitrag/docex-service/docker-compose.offline.yml
services:
  docex-service:
    image: efrsb/docex-service:<version>
    pull_policy: never
    build: !reset null
  rabbitmq:
    pull_policy: never
```

```
# /opt/Arbitrag/docex/docker-compose.offline.yml
services:
  docex-ocr:
    image: efrsb/docex-ocr:<version>
    pull_policy: never
    build: !reset null
```

В .env платформы оверлей добавляется в COMPOSE_FILE; для docex-service и docex он передаётся флагом -f в командах ниже. make up на таком контуре **не использовать**.

```
Шаг 1. OCR-сервис
cd /opt/Arbitrag/docex
docker compose -f docker-compose.yml -f docker-compose.offline.yml up -d
# при сборке из исходников: docker compose up -d --build
```

```
docker compose logs -f --tail=50      # дождаться сообщения uvicorn о старте на :8001, Ctrl+C
```

Шаг 2. Сервис извлечения данных и RabbitMQ

```
cd /opt/Arbitrag/docex-service
docker compose -f docker-compose.yml -f docker-compose.offline.yml up -d
# при сборке из исходников: docker compose up -d --build
docker compose logs -f --tail=50 docex-service # дождаться "Started ... Application", Ctrl+C
```

Топология брокера (exchange docex.exchange, очередь docex.jobs) создаётся приложением автоматически.

Шаг 3. Платформа

```
cd /opt/Arbitrag/bankruptcy-platform
docker compose up -d      # COMPOSE_FILE из .env: base + stage [+ offline] [+ external-db]
# при сборке из исходников: make up (создаёт сеть, грузит logdy, docker compose up --build -d)
docker compose logs -f --tail=100 backend
```

Порядок внутри платформы обеспечивают healthcheck-и compose: postgres → keycloak → backend → frontend → nginx. Первый старт занимает несколько минут: инициализация БД, импорт realm Keycloak (~40 с), прогон 158 миграций Flyway. Признак успеха в логе бэкенда – Successfully applied N migrations и Started EfrsbApplication.

Остановка: docker compose down в каждом каталоге в обратном порядке (платформа → docex-service → docex). Флаг -v удаляет тома с данными – использовать осознанно.

Автозапуск после перезагрузки сервера обеспечивается политикой restart: unless-stopped контейнеров при условии, что служба Docker включена в автозагрузку (systemctl enable docker).

5. ПРОВЕРКА РАБОТОСПОСОБНОСТИ

5.1. Состояние контейнеров

```
docker ps --format 'table {{.Names}}\t{{.Status}}'
```

Ожидаемый результат – девять контейнеров в состоянии Up (для efrsb-postgres и efrsb-keycloak – Up ... (healthy)), ни одного в Restarting:

- efrsb-nginx, efrsb-frontend, efrsb-backend, efrsb-postgres, efrsb-keycloak;
- efrsb-logdy, efrsb-docex-service, efrsb-docex-rabbitmq, efrsb-docex-ocr.

5.2. Проверка компонентов

Список компонентов для проверки представлен в таблице 13.

Таблица 13 – Список компонентов

Компонент	Команда	Ожидаемый результат
PostgreSQL	docker exec efrsb-postgres pg_isready -U efrsb -d efrsb	accepting connections
Схема БД	docker exec efrsb-postgres psql -U efrsb -d efrsb -c "SELECT version, success FROM flyway_schema_history ORDER BY installed_rank DESC LIMIT 1;"	Последняя миграция релиза, success = t
Keycloak	docker exec efrsb-keycloak curl -sf http://localhost:8080/auth/health/ready	{"status": "UP", ...}
Backend	docker exec efrsb-backend wget -qO- http://localhost:3080/actuator/health	{"status": "UP"}
docex-service	docker exec efrsb-docex-service wget -qO- http://localhost:8080/v1/schemas \ head -c 200	JSON со списком типов документов
OCR	docker run --rm --network efrsb-network curlimages/curl -s -o /dev/null -w '%{http_code}\n' http://efrsb-docex-ocr:8001/docs	200
RabbitMQ	docker exec efrsb-docex-rabbitmq rabbitmqctl list_queues	В списке очередь docex.jobs
nginx	curl -sI http://<host>/ \ head -1	HTTP/1.1 302 (редирект на форму входа)

При внешней БД (п. 4.3.2) команды pg_isready/psql выполняются на сервере СУБД либо через psql с сервера контейнеров.

5.3. Проверка через браузер

1. Открыть <http://<host>/> – должен произойти редирект на форму входа Keycloak (/auth/realms/efrsb/protocol/openid-connect/auth?...) с брендированной темой.
2. Войти демонстрационной учётной записью поставочного realm test_au / password123 (роль ROLE_AU) либо учётной записью, заведённой в Keycloak (руководство

администратора, §7.7). После входа открывается главная страница приложения, список дел загружается без ошибок.

3. Убедиться, что платформа создала профиль пользователя при первом входе:

```
docker exec efrsb-postgres psql -U efrsb -d efrsb -c \
"SELECT username, sro_id, last_login_at FROM user_profiles ORDER BY last_login_at DESC LIMIT 3;"
```

4. Служебные страницы: <http://<host>/swagger-ui.html> (Swagger UI), <http://<host>/api-docs> (OpenAPI), <http://<host>/auth/> (admin-консоль Keycloak), <http://<host>/logs/> (Logdy – после входа в приложение; заполняется, когда бэкенд запишет первые строки лога).
5. Выход из системы (/logout) возвращает на форму входа.

5.4. Проверка распознавания документов (при наличии LLM-шлюза)

Из контейнера docex-service проверить доступность шлюза:

```
docker exec efrsb-docex-service wget -qO- --header="Authorization: Bearer <token>" \
<ollama.base-url>/models | head -c 300
```

Затем в приложении открыть дело, загрузить документ основания требования (PDF с текстовым слоем) и убедиться, что через 1–3 минуты поля требования заполнены результатом распознавания. Первый вызов OCR на контуре без предзагруженных моделей занимает дольше – идёт скачивание моделей в том paddleocr_models.

5.5. Типовые проблемы при первом запуске

Список типовых проблем представлен в таблице 14.

Таблица 14 – Список типовых проблем

Симптом	Причина	Действие
network efrsb-network ... not found	Не создана docker-сеть	docker network create efrsb-network (п. 4.2)
pull access denied / compose пытается собрать образ	Не подключён docker-compose.offline.yml либо образ не загружен	Проверить COMPOSE_FILE, docker image ls (п. 4.1, п. 4.5)
pull access denied ... logdy	Не загружен образ Logdy	make logdy-image
port is already allocated	Порт занят на хосте	Изменить PG_EXTERNAL_PORT / DOCEX_HTTP_PORT
Бэкенд падает: ошибка подключения к БД, role "app_user" does not exist	Init-скрипт не отработал (том создан ранее) либо пароли рассогласованы	Проверить п. 4.3; при пустом контуре – docker compose down -v и повторный запуск
Бэкенд падает: ошибка Flyway	Неуспешная миграция	Руководство администратора, (flyway repair)
В логе бэкенда ошибки подключения к RabbitMQ, уведомления не доставляются	Не запущен docex-service (шаг 2 п. 4.5)	Запустить docex-service, перезапустить backend

Симптом	Причина	Действие
Циклический редирект на форму входа	Заданы APP_BASE_URL/BASE_URL, не совпадающие с адресом, по которому открыт контур	Убрать переменные либо привести к фактическому адресу (п. 4.4.1)
/logs/ пустой	Каталог логов пуст или не смонтирован	Дождаться первых записей бэкенда; проверить путь bind-mount (п. 4.2)

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

№ версии док-та	Дата изменения	Автор изменений	Изменения
1.0	14.09.2026	Алексеев И.И.	Первоначальная версия документа