**Руководство администратора**

**1. Термины и сокращения**

**1.1 Определение терминов**

- Система: Комплекс программного обеспечения, включающий фронтенд, бэкенд, базы данных и связанные сервисы, которые обеспечивают функциональность приложения.

- Микросервис: Независимый модуль системы, который выполняет определенную функцию и может быть развернут и масштабирован отдельно от других компонентов.

- Фронтенд: Часть системы, отвечающая за пользовательский интерфейс.

- Бэкенд: Серверная часть системы, отвечающая за бизнес-логику и обработку данных, реализованная на Java.

- Docker: Платформа для контейнеризации, которая позволяет упаковывать приложения и их зависимости в контейнеры для облегчения развертывания.

- Dockerfile — это сценарий, содержащий команды для сборки Docker-образа для каждого компонента системы.

**1.2 Список сокращений**

- ИС: Информационная система

- CI/CD: Непрерывная интеграция и непрерывная доставка (Continuous Integration/Continuous Deployment)

- API: Интерфейс программирования приложений (Application Programming Interface)

- DB: База данных (Database)

**2. Порядок ввода системы в эксплуатацию**

**2.1 Общая информация**

Ввод системы в эксплуатацию включает в себя развертывание всех модулей и компонентов системы, настройку окружения, конфигурацию микросервисов и запуск связанных сервисов. Важно следовать инструкциям для обеспечения корректной работы всех компонентов.

**2.2 Перечень модулей и компонентов системы**

**2.2.1 Фронтенд (React)**

Фронтенд разработан с использованием TypeScript и отвечает за отображение пользовательского интерфейса и взаимодействие с пользователями через API.

**2.2.2 Бэкенд (Java)**

Бэкенд реализован на Java и отвечает за бизнес-логику приложения, обработку запросов и взаимодействие с базами данных.

**2.2.3 Базы данных**

Система использует реляционные базы данных для хранения и управления данными.

**2.3 Настройка Docker и Docker Compose**

**2.3.1 Dockerfile для бэкенда:**FROM maven:3.9.2-eclipse-temurin-17-alpine AS build

WORKDIR /build

COPY pom.xml .

RUN mvn dependency:go-offline

COPY src ./src

RUN mvn clean package -DskipTests

FROM openjdk:17-jdk-alpine

WORKDIR /app

COPY --from=build /build/target/t1-0.0.1-SNAPSHOT.jar /app/app.jar

CMD ["java", "-jar", "/app/app.jar"]

EXPOSE 8080

**2.3.2 Конфигурация docker-compose.yml бэкенда:**

version: '3'

services:

app:

build:

context: .

dockerfile: Dockerfile

container\_name: spring-boot-app

ports:

- "8080:8080"

env\_file:

- .env

depends\_on:

- database

database:

image: postgres:13

container\_name: postgres-db

volumes:

- postgres\_volume:/var/lib/postgresql/data

env\_file:

- .env

volumes:

postgres\_volume:

**2.4.3 Конфигурация docker-compose.yml фронта:**

services:

frontend:

build:

context: .

dockerfile: Dockerfile

ports:

- "3000:3000"

restart: always

**2.4.4 Dockerfile для фронта:**

# Используем официальный образ Node.js как базовый

FROM node:16.16.0 AS build

# Создаем рабочую директорию внутри контейнера

WORKDIR /frontend

# Копируем package.json и package-lock.json в рабочую директорию контейнера

COPY package\*.json ./

# Устанавливаем зависимости

RUN npm install

# Копируем все файлы проекdocker build -t my-app .та в рабочую директорию контейнера

COPY . .

# Собираем проект (если это необходимо)

RUN npm run build

# FROM nginx:alpine

# COPY --from=build /app/build /usr/share/nginx/html

# Команда по умолчанию для запуска приложения

CMD ["npm", "start", "serve", "-s", "build"]

# Указываем, какой порт должен быть открыт

EXPOSE 3000

**2.5.1. Предварительные условия:**

* Установленный и настроенный сервер с операционной системой (например, Ubuntu, CentOS).
* Установленный Nginx.
* Наличие прав администратора на сервере.
* Сборка фронтенда (обычно в виде статических файлов HTML, CSS, JavaScript).

**2.5.5. Проверка:**

* Откройте веб-браузер и перейдите по адресу 10.4.56.84:3001.
* Вы должны увидеть ваш работающий фронтенд.