API Monitoring Service

Задание

Разработать Spring Boot-приложение, которое опрашивает публичный API по таймеру, сохраняет данные в базу, обрабатывает ошибки с помощью Spring Retry, отправляет сообщения в Kafka и предоставляет защищённое REST API.

Стек технологий

- Java 17+
- Spring Boot 3+
- Spring Data JPA
- Spring Security (Basic Auth или JWT)
- Spring Retry
- Apache Kafka
- Lombok (опционально)
- PostgreSQL (или H2)
- Docker Compose (опционально)
- Swagger / OpenAPI (опционально)

Функциональные требования

Периодический опрос API

- Каждую минуту (через `@Scheduled`) опрашивать публичный API, например:
  https://api.coindesk.com/v1/bpi/currentprice.json
- Ответ сохраняется в базу данных.
- Использовать `Spring Retry` с 3 попытками и экспоненциальной задержкой.

Интеграция брокера сообщений

- В случае успешного ответа — отправить сообщение в Kafka-топик "api-data".
- В случае неудачи после всех ретраев — отправить сообщение в Kafka-топик "api-errors".

База данных

- Использовать PostgreSQL или H2.
- Сущность `ApiDataEntity`:

- "id": UUID
- "createdAt": дата запроса
- "success": boolean
- "payload": текст ответа или сообщение об ошибке

REST API с авторизацией

Реализовать следующие эндпоинты:

Метод	URI	Доступ	Описание
GET	"/status"	USER/ADMIN	Проверка статуса сервиса
GET	"/data"	ADMIN	Получение последних 10 записей

Использовать Spring Security:

- Роли: `ROLE_USER`, `ROLE_ADMIN`
- Auth: Basic Auth или JWT (на выбор разработчика)

Бонус (необязательно)

- Описать API через Swagger/OpenAPI
- Добавить Docker Compose для Kafka + PostgreSQL
- Написать unit/integration тесты хотя бы на один компонент
- Реализовать глобальный `@ControllerAdvice` для обработки ошибок

Что сдавать

- Ссылку на GitHub-репозиторий
- Краткое описание архитектуры и принятых решений
- Инструкции по запуску