Тестовое задание ITQ Group

Нужно сделать backend-сервис по работе с документами.
Документы создаются, переводятся по статусам, по изменениям статуса ведётся история.
Дополнительно нужна утилита для массового создания документов и фоновая обработка документов пачками.

Стек

Java + Spring Boot,
PostgreSQL (Docker Compose),
JPA/Hibernate,
Liquibase,
Maven/Gradle.

Функциональные требования

Документ

Поля: внутренний id, уникальный номер (генерируется при создании), автор, название, статус, даты создания/обновления.
Статусы: `DRAFT`, `SUBMITTED`, `APPROVED`.

История

На каждом переводе статуса сохраняйте запись: кто выполнил действие, когда, какое действие, комментарий (может быть пустым).
Действия: `SUBMIT`, `APPROVE`.
Во всех запросах на создание/перевод статуса передаётся инициатор.

Реестр утверждений

При успешном утверждении документа создаётся запись в реестре утверждений (отдельная таблица/хранилище в рамках проекта).
Если запись в реестр создать не удалось, утверждение документа должно быть отменено.

API

1) Создать документ

Создаёт документ в статусе `DRAFT`. Номер генерируется автоматически.

2) Получить документы

API должно уметь:

- вернуть один документ вместе с историей,
- вернуть список документов по списку id (пакетное получение).

Способ задания режима/параметров - на ваше усмотрение.
Обязательно должна быть пагинация и сортировка.

3) Отправить на согласование

Принимает список id (от 1 до 1000) и пытается перевести каждый документ `DRAFT -> SUBMITTED`.
Для каждого id вернуть результат: успешно / конфликт / не найдено.
Обработка каждого документа атомарная, результаты - по каждому id (частичные успехи допустимы).

4) Утвердить

Принимает список id (от 1 до 1000) и пытается перевести каждый документ `SUBMITTED -> APPROVED`.
Для каждого id вернуть результат: успешно / конфликт / не найдено / ошибка регистрации в реестре.
Обработка каждого документа атомарная, результаты - по каждому id.

При успешном утверждении:

- пишется запись в историю,
- создаётся запись в регистре утверждений.
  Если запись в регистр не создана - утверждение документа должно быть отменено.

5) Поиск документов

Фильтры: статус, автор, период дат. Вернуть список найденных документов.
Как трактовать период (по дате создания или обновления) - выберите сами и укажите в README.

6) Проверка конкурентного утверждения

Нужен отдельный API, который запускает несколько параллельных попыток утвердить один и тот же документ (параметры threads и attempts) и возвращает сводку: сколько попыток прошло успешно, сколько завершилось конфликтом/ошибкой, какой финальный статус документа.
Ожидаемое поведение: ровно одна попытка должна перевести документ в APPROVED и создать запись в реестре.
Остальные попытки должны завершиться без изменения документа и без повторной записи в регистр**.**

Утилита + фоновая обработка

Утилита генерации

Сделайте отдельный модуль/утилиту, которая читает из файла параметров число `N` и создаёт `N` документов через API сервиса.

Фоновые процессы в сервисе

В сервисе должны работать два фоновых процесса (в отдельных потоках/задачах). Размер пачки должен задаваться параметром `batchSize` (в конфиге/файле параметров) и использоваться в обработке.

1. **SUBMIT-worker**
   Регулярно проверяет БД и отправляет документы со статусом `DRAFT` на согласование пачками по `batchSize` через пакетное API.
2. **APPROVE-worker**
   Регулярно проверяет БД и отправляет документы со статусом `SUBMITTED` на утверждение пачками по `batchSize` через пакетное API.

`batchSize` задаётся параметром.

Частичные ошибки не должны останавливать обработку.

Логи

По логам должно быть понятно:

- сколько документов задано к созданию (N) и прогресс создания;
- время выполнения шагов: создание, каждая отправка пачки на согласование, каждая отправка пачки на утверждение;
- ход фоновой обработки (сколько обработано/осталось).

Ошибки и базовая валидация

Ошибки и базовая валидация

- Единый формат ошибок: код + сообщение.
- “не найдено” для отсутствующего документа.
- “конфликт/недопустимая операция” для недопустимого перехода статуса.
- Проверка входных данных (пустые значения и т.п.).

Что предоставить

- Ссылка на Git репозиторий.
- Docker Compose для PostgreSQL.
- Liquibase миграции (поднятие схемы с нуля).
- README: как запустить сервис и утилиту, как проверить прогресс по логам.
- `EXPLAIN.md`: пример поискового запроса + `EXPLAIN (ANALYZE)` + короткое пояснение по индексам.
- Тесты (минимум): happy-path по одному документу, пакетный submit, пакетный approve с частичными результатами, откат approve при ошибке записи в регистр.

Опционально

Эти пункты можно реализовать в коде или кратко описать в README:

- что бы вы поменяли, чтобы обработка одного запроса уверенно работала с 5000+ id;
- как бы вы вынесли реестр утверждений в отдельную систему (отдельная БД или отдельный HTTP-сервис).