Clevertec

Система управления новостями

Разработать RESTful web-service, реализующей функционал для работы с системой управления новостями.

Основные сущности:

- **news** (новость) содержит поля: id, date, title, text и comments (list).
- **comment** содержит поля: id, date, text, username и id_news.

Требования:

1. Использовать Spring Boot
2. Разработать API согласно подходов REST (UI не надо). Для потенциально объемных запросов реализовать постраничность. Основые API:
   - CRUD для работы с новостью
   - CRUD для работы с комментарием
   - просмотр списка новостей (с пагинацией)
   - просмотр новости с комментариями относящимися к ней (с пагинацией)
   - полнотекстовый поиск по различным параметрам
3. Разместить проект в любом из публичных git-репозиториев (Bitbucket, github, gitlab)
4. Использовать Gradle в качестве сборщика
5. Код должен быть легко читаемый и содержать комментарии
6. Реализовать на основе Spring @Profile (e.g. test & prod) подключение к базе данных - встроенная и внешняя соответственно. БД PostgreSQL или MongoDB
7. В тестовом режиме база данных должна наполняться автоматически - проект должен включать data.sql-файл, по которому генерируются необходимые таблицы и наполняются таблицы тестовыми данными (20 новостей и 10 комментариев, связанных с каждой новостью)
8. Сущности веб интерфейса (DTO) должны генерироваться при сборке проекта из .proto файлов (см. https://github.com/google/protobuf-gradle-plugin)
9. Весь код должен быть покрыт юнит-тестами
10. Реализовать логирование запрос-ответ в аспектном стиле, а также логирование по уровням в отдельных слоях приложения
11. Предусмотреть обработку исключений и интерпретацию их согласно REST (см. https://spring.io/blog/2013/11/01/exception-handling-in-spring-mvc)
12. Все настройки должны быть вынесены в *.yml
13. Код должен быть документирован @JavaDoc, а назначение приложения и его интерфейс и настройки должны быть описаны в README.md файле
14. * Использовать Spring REST Docs или другие средства автоматического документирования (например asciidoctor

https://asciidoctor.org/docs/asciidoctor-gradle-plugin/ и т.д) и/или Swagger (OpenAPI 3.0)

15. * Написать интеграционные тесты
16. * Использовать WireMock в тестах
17. * Spring Security:
    - API для регистрации пользователей с ролями admin/journalist/subscriber
    - Администратор (role admin) может производить CRUD-операции со всеми сущностями
    - Журналист (role journalist) может добавлять и изменять/удалять только свои новости
    - Подписчик (role subscriber) может добавлять и изменять/удалять только свои комментарии
    - Незарегистрированные пользователи могут только просматривать новости и комментарии
18. * Использовать Docker (написать Dockerfile, docker-compose.yml для поднятия БД и приложения в контейнерах и настроить взаимодействие между ними)
19. * Реализовать кеширование
20. * Настроить Spring Cloud Config (вынести настройки в отдельный сервис и настроить разрабатываемый сервис на получение их в зависимости от профиля)

“*” - Необязательно, но будет существенным плюсом. К этим пунктам лучше приступать после качественного решения базовых задач с применением принципов SOLID, декларативных подходов, оптимальных алгоритмов.