Гранков Георгий, Б05-211

Проект по джаве

**Запуск**: можно запускать через мавен (spring-boot:run) или через main в классе TestApiApplication

**SqlConfig**: для того, чтобы запускался sql скрипт. В init.sql заполняются тестовые данные. Если нужно что-то дополнить, то можно расширить последние три строчки в init.sql. Этот файл сам ищет sql скрипт и запускает его, когда data source поднимается как бин.

**SwaggerConfig**: используем документацию, нужен для того, чтобы появился UI сваггер. Заходим на сайт для сваггера. В самом первом get можно проверить и убедиться, что бд заполняется на самом деле. Запустили – сразу получили ответ. Нужно учитывать, что каждый раз, когда мы запускаем приложение, бд создается по-новой, то есть предыд. запуски не сохраняются, т.к. бд лежит в памяти и когда заканчивается работа приложения, то все стирается.

**SequrityConfig**: мы храним пользователей в памяти, бд мы не используем для упрощения. Чтобы это все-таки хранить, нужно сделать то же самое, как со статьями, только с пользователями. В нашем случае мы просто настраиваем секьюрити.

**Сущности**: достаточно все просто, у нас есть предметная область статьи, есть неотъемлемая часть id, потом название

**Файл init.sql** – 22 строка про основную информацию, когда опубликована статья, про заголовок статьи и ее несколько глав (23 и 24), которые добавляются в отдельную таблицу.

**ArticleController**: методы – добавление статьи, получить статью по id, показать все статьи (permitAll), также мы можем выбрать страницу и ее размер = попросить вторую страницу с размером 1 или первую страницу с размером 10 (список из первых 10 статей)

**StatsController**: статистика (заходим на stats-controller, нужно авторизоваться). Мы увидим, что у нас в определенный день была опубликована статья. Если мы сделаем еще несколько статей, то там будет другое значение.

В package **service** обычная бизнес-логика. Тесты тоже есть на бизнес-логику и контроллеры (как юнит-тесты, то есть как все отрабатывает)

**ArticleRepository** генерирует и отправляет в бд, а sql уже конвертирует. Т.к. это интерфейс и реального класса нет, то мы можем в **ArticleServiceTest** его моккировать (через mockito соответ.) то есть задать ему какое-то значение (34 строка в ArticleServiceTest) Т.к. в тестах (в юнит-тестах) нет бд (она не создается). Мы обязаны сделать 34 строку, иначе выдаст ошибку, т.к. в getList имитируется работа репозитория. К примеру, в **StatisticServiceTest** есть помимо бд есть другая логика, тесты на другие кейсы. В **ArticleServiceTest** тесты на пустой список, с какими-то данными, когда мы ничего не нашли, когда не нашли контент, не нашли дату публикации, когда мы пытаемся сохранить и получим по id. В **StatisticServiceTest** мы проверяем логику собирания статистики, заводим данные, от которых мы ожидаем, что будут возвращены правильные.

По контроллерам тестируем через **ArticleControllerTest**, также тестируем с учетом спринг секьюрити (пытаемся добавить стать неавторизованным пользователем и ждем ошибку, т.к. не можем добавить неавторизованных, в итоге получим какую-то из 401, 403 ошибок (интеграционные тесты). Заметим, что в данном тесте полностью поднимается приложение само, прогоняются тесты и приложение закрывается.

Т.к. бд находится в памяти, то не приходится никаких тест-контейнеров делать, достаточно поднять тесты и все

Как вариант можно доработать проект до следующих условий: использовать более сложные б/д для расширения статистики, для увеличения возможностей администратора (доступ к более важной информации, скрытой от остальных пользователей)