Примечания

1. Задание сформулировано для проекта по курсу БД, как самого недавнего большого проекта для студентов. В качестве объекта тестирования может быть выбран любой многокомпонентный проект, написанный или изученный студентом. (Если проект сильно отличается от того, что подразумевалось в курсах ППО и курсовой работе по БД – согласовать с преподавателем.)
2. Проект и тесты могут быть написаны на любых языках программирования (например можно как взять проект на java, а тесты писать на kotlin, так и всё сделать на python).
3. Не надо пытаться сделать свой фреймворк для юнит тестов – нужно использовать типичные для выбранного языка.

Лабораторная работа № 2

Задание

1. Написать integration-тесты для компонентов доступа к данным и бизнес логики выбранного проекта
2. Написать E2E-тест для демонстрационного сценария (например того, который был бы использован при защите демонстрации minimum viable product)
3. Организовать запуск тестов из лабораторной работы №1 и вновь добавленных в рамках работы №2 в любой, на усмотрение студента, CI\CD среде (GitLab, Jenkins, TeamCity, etc.)
4. Проимитировать действия из E2E теста с помощью средства для отправки запросов, снять лог с помощью средства захвата трафика (если в проекте нет подобных действий, то согласовать с преподавателем что делать)

Требования

1. Требуется оформить запуск тестов в виде отдельного контейнера docker: выкачивается репозиторий с кодом, устанавливаются все внешние зависимости и всё для запуск тестов на используемом языке программирования, контейнер вызывается внутри выбранной CI\CD среды (окружение, которое будет тестироваться, тоже лучше развернуть в контейнере)
2. Интеграционные тесты должны включать в себя взаимодействие с хранилищем данных (который может быть любым – sql / nosql / file storage / etc.)
3. Для тестов должен инициализироваться отдельный инстанс хранилища
4. Инстанс хранилища поднимать с помощью последовательного запуска скриптов – например генерация таблиц, заведение тестовых данных, откат состояния на то, которое было до прогона тестов – либо из какого-то образа, который будет идентичен при каждом запуске тестов
5. Порядок запуска тестов в CI\CD: unit → integration → e2e
6. Если один из этапов свалился, последующие запускать не нужно, но автоматический отчет должен сгенерироваться. Часть которая не запускалась помечается как skipped или ignored в зависимости от выбранного средства генерации отчетов

7. Если свалился integration или e2e этап, то требуется принудительно откатить хранилище на состояние до запуска тестов
8. Если используется service bus / message broker / шина / и т.д. и т.п. то после аварийного или успешного завершения тестов требуется произвести вычитывание всех сообщений из очереди
9. Если в проекте используется хранение активных сессией пользователей в том или ином виде, то требуется предусмотреть завершение всех сессий после аварийного или успешного завершения тестов
10. Не требуется включать этап тестирования GUI в E2E тест
11. Убедиться, что интеграционные тесты могут быть запущены несколько раз подряд и при каждой попытке результаты не изменились
12. Убедиться, что тесты могут быть запущены несколькими разработчиками локально одновременно без влияния на результаты друг друга
13. Тесты должны проходить успешно
14. Рекомендации по структуре файлов с тестами:
    a. один файл – один тест из нескольких шагов (секций act, assert)
    b. arrange по возможности желательно сделать один раз в начале теста;
    c. если для unit-тестов рекомендуется использовать суффикс test\Test_test\test_\и т.д. в зависимости от языка программирования, то для интеграционных также лучше использовать соответствующий суффикс, например для Java это ITCase (сокращение от Integration Test Case) – уточняйте для выбранного языка программирования сами