# Тестовое задание

Разработать REST API для системы управления конфигурациями в некой web-системе (например, облачной бухгалтерии), позволяющей пользователю сохранять и загружать различные настройки (наборы горячих клавиш, схемы цветов, шрифты, расположение окон) и получать уведомления об изменениях в конфигурациях в реальном времени.

# Основные требования:

1. API для управления конфигурациями:
    - Создание, получение списка конфигураций;
        i. Можно ограничить создание одинаковых названий конфигураций для одного пользователя;
        ii. Для списка предусмотреть фильтрацию конфигураций по названию или дате создания;
    - Редактирование - поддержка версионирования конфигураций с возможностью перехода с одной версии на другую
2. Real-Time с использованием SignalR:
    - При создании и обновлении конфигурации отправлять уведомления о событии всем подписавшимся пользователям (например, “Конфигурация обновлена”);
        i. Поддерживать подписки по конкретным типам событий (например, пользователи могут подписываться на события изменения конфигураций, но игнорировать создание новых).
        ii. При подписке на любые события автоматически отправлять полный список конфигураций
3. Хранение данных:
    - Использовать EF Core. В качестве СУБД использовать SQLite или другую (в том числе NoSQL). Если используется отличная от SQLite СУБД - подготовить проект для запуска в Docker с образом, включающим API и базу данных.

# Дополнительные требования:

1. Архитектура и тестируемость:
    - Обеспечить минимальные зависимости между компонентами, чтобы позволить гибко менять логику фильтрации и обработки конфигураций.
    - Изолировать бизнес-логику от слоя данных и SignalR;
2. Тестирование и документация:
    - Написать unit-тесты для основных операций и бизнес-логики;
    - Использовать Swagger для описания API.
3. Docker *(при использовании СУБД, отличной от SQLite)*:
    - Подготовить Dockerfile для сборки образа с API и базой данных;
    - Добавить Docker Compose для развертывания сервиса и базы данных.

# Критерии оценки:

- Структурированность и читаемость кода, продуманность архитектуры;

- Гибкость реализации и использование SignalR для оповещений в реальном времени;
- Использование паттернов проектирования, обеспечивающих масштабируемость и простоту поддержки;
- Качество тестов и полнота покрытия функционала;
- Возможность запуска и отладки через Docker (при поставке тестового задания в этом варианте).