Задание 1

Создать микросервис для WebAPI. Основные требования:

* Платформа кода .NET Core 3.1 или выше
* Микросервис должен запускаться в контейнере Docker с типом контейнера Linux
* Микросервис должен реализовать Health Check
* Микросервис должен включать стандартное логгирование
* Микросервис должен предоставлять доступ к описанию методов через Swagger
* Миросервис должен реализовать следующие методы:
  + Метод для получения фекового пользователя – id, firstName, lastName
  + Метод для добавления фейкового пользователя
  + Метод для получения всех фейковых пользователей

Задание 2

Подключение реальной базы данных.

* БД – MS SQL Server запускается в Docker контейнере (Linux)
* К Docker контейнеру должны быть подключен volume для хранения файлов БД
* БД изначально нет – её необходимо сгенерировать при первом зауске и разместить в Volume
* Использовать EF Core для доступа к БД
* Создать скрипты для запуска контейнеров в нужной последовательности

Задание 3

Обеспечение безопасности.

* Необходимо модифицировать API и БД для регистрации нового пользователя, изменения его данных, хранения хэша пароля. Дополнительными полями будут: login, email, passwordHash.
* Создать API метод для восстановления пароля
* Создать метод для логина. За основу аутентификации/авторизации на базе JWT без шифрации. За полезный payload в JWT токене взять id пользователя.
* Обеспечить авторизацию всех методов API, за исключением акций: create user, login user, restore password.
* Логгировать авторизацию централизованно – выводить id пользователя в качестве payload.

Задание 4

Дополнительные сервисы.

* Необходимо создать 2 дополнительных сервиса, по структуре напоминающих первый
* Один сервис предназначен для регистрации транзакций. Он должен иметь свою базу данных (можно реиспользовать всё тот-же MS SQL Server, но другая БД). Основная операция – это регистрация транзакции. Состав транзакции: id пользователя, время проведения транзакции, сумма транзакции, валюта. Операция должна быть авторизована и привязана к пользователю через JWT-токен.
* Следущий сервис – хранение кошелька пользователя. Операции должна быть авторизована и привязана к пользователю через JWT-токен. Основные операции:
  + изменение сумм в разных валютах для кошелька (проверка при списании суммы, что такая имеется)
  + текущий баланс кошелька в разных валютах
* Оба новых сервиса должны соответствовать требованиям самого первого сервиса: логгирование, безопасность, запуск в контейнере Docker, Health Check, Swagger, проверка БД при запуске, JWT-токен.
* Добавить скрипты, позволяющие запускать совместно все 3 сервиса и БД в Docker.

Задание 5

Общение между сервисами.

* Необходимо запустить RabbitMQ в Docker контейнере
* Для сервиса с транзакциями для операции создания транзакции необходимо добавить в очередь сообщение об изменении баланса кошелька.
* Для сервиса кошелька необходимо принять сообщение из очереди RabbitMQ и провести операцию по изменению баланса кошелька пользователя.
* Для обеспечения транзакционности, применить паттерн Saga при изменении данных на обоих сервисах.
* Для демонстрации отказоустойчивости и консистенси транзакций, обеспечить генерацию ошибки на сервисе кошелька, когда сумма изменений кошелька пользователя превысит определенный порог (задаётся в конфиге).