Задача

Разработать систему, которая решает следующую задачу: на вход подаётся заголовок страницы A и страницы B с Википедии. Нужно найти кратчайший путь (количество переходов от одной страницы к другой) от страницы A до страницы B.

Требования к системе

Аутентификация
Система должна работать по подписочной системе и предоставлять три опции подписки:

- бесплатное использование (1 запрос в день)
- стандартная подписка (20 запросов в день)
- про подписка (неограниченное количество запросов)

Необходимо реализовать сервис который будет генерировать API tokens и предоставлять API для валидации.

+ *Необходимо реализовать пользовательский интерфейс для генерации токенов*

Администрирование
Система должна предоставлять возможность вносить определенные страницы в список запрещенных для поиска/переходов по ним.

+ *Необходимо реализовать пользовательский интерфейс*

Использование
Предоставлен REST API для взаимодействия с системой. REST API доступно и задокументировано через swagger

+ *Необходимо реализовать пользовательский интерфейс с выводом получившегося графа*

Развертывание
Развертывание системы должно производиться с помощью docker compose файла в docker контейнер. *(Должно запуститься на компьютере проверяющего без проблем)*

Масштабирование
Необходимо использовать микросервисную архитектуру позволяющую масштабировать систему как вертикально(многопоточность, параллелизм), так и горизонтально(распределение нагрузки между несколькими инстансами одного микросервиса)

Отказоустойчивость
При падении одного из микросервисов система должна продолжить работу с момента падения и завершить запрос пользователя. (При параллельной индексации страниц разными инстансами, во время падения одного из сервисов, оставшиеся инстансы продолжают индексацию с того места, на котором произошло падение, **не производя переиндексацию с самого начала**)

Обязательные условия:

- Использование многопоточности
- Использование шины данных (Kafka)
- Использование библиотеки https://github.com/andrsuh/tiny-event-sourcing
- Использование event sourcing подхода (опираться на материалы лекций)
- Кэширование промежуточных результатов

- Построение графа переходов
- Ведение истории запросов каждого пользователя(токена) + сохранение результатов запросов в базу
- Инвалидация графа при наличии изменений страницы
- Использование микросервисного подхода

Критерии оценивания:

- использование паттернов ООП, DDD
- использование микросервисной архитектуры
- соответствие требованиям
    - если требование не однозначно интерпретировано исполнителем, исполнитель уточняет требование у заказчика
- отзывчивость пользовательских интерфейсов
- соблюдение обязательных условий в реализации
- уникальность работы и чистота истории в GIT репозитории

Реализация пользовательского интерфейса может быть выполнена в качестве web интерфейса, телеграмм бота или любой другой графической оболочки

Перед началом работы необходимо создать репозиторий на github и поддерживать в актуальном состоянии в течении всего времени выполнения работы

Необходимо составить презентацию на 5-15 слайдов с описанием архитектуры представленной в виде UML диаграмм (минимум 3 вида диаграмм).