Описание решения:

Внедрен в рамках связки теории с практикой Gunicorn. Neo4j хранит связи между пользователями и контентом. Python является основным языком разработки. В requirements.txt описаны зависимости.

docker-compose up -d

Логин: neo4j, пароль: password, можете своё вписать.

Ссылка: http://localhost:7474/browser/

Для создания 2 пользователей и связи между ними: CREATE (u:User {name: "Alice"})-[:FOLLOWS]->(v:User {name: "Bob"})

Проверка пользователей, выполнять в терминале: curl <http://localhost:5000/users>

Создание поста с хэштегом: CREATE (p:Post {text: "Hello #world!"})-[:USES]->(h:Hashtag {tag: "world"})

Проверка в терминале: curl <http://localhost:5000/hashtags/popular>

Запуск:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

Запросы:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Добавлены 2 метки, созданы 2 узла, заданы 2 свойства, создано 1 отношение, завершено через 39 мс.

Изображение выглядит как текст, снимок экрана, Шрифт, чек

Контент, сгенерированный ИИ, может содержать ошибки.

Пример проверки через само приложение:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Проверка в терминале:

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.