

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

"Российский технологический университет"

МИРЭА

Институт «Кибербезопасности и цифровых технологий»

Кафедра КБ-3 «Разработка программных решений и системного программирования»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1**

**по дисциплине**

**«Проектирование Архитектуры программного обеспечения»**

**Выполнил**

студент группы БСБО-02-20 Смелкин Н.Д.

*ФИО*

**Принял**

к.т.н., доцент Волович М. Е.

**Работа выполнена**

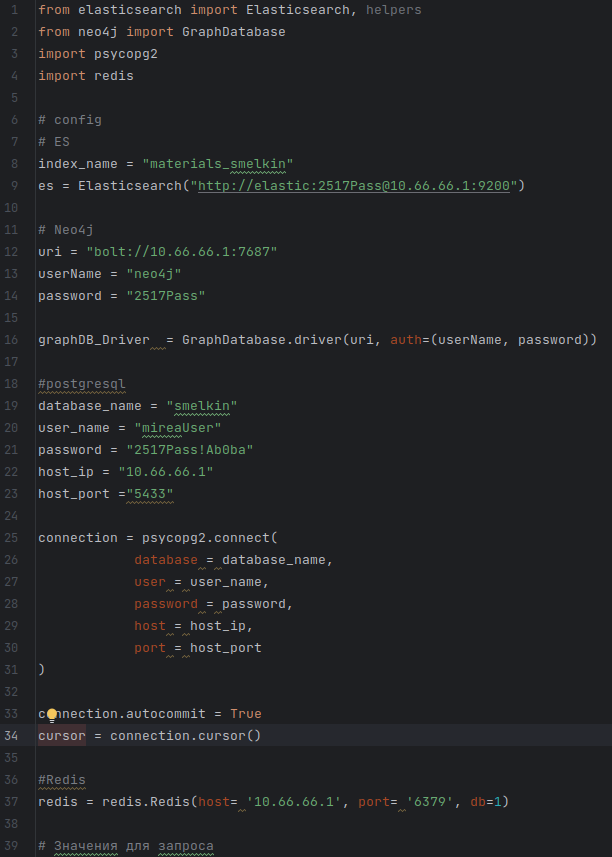
« » н 2023 г.

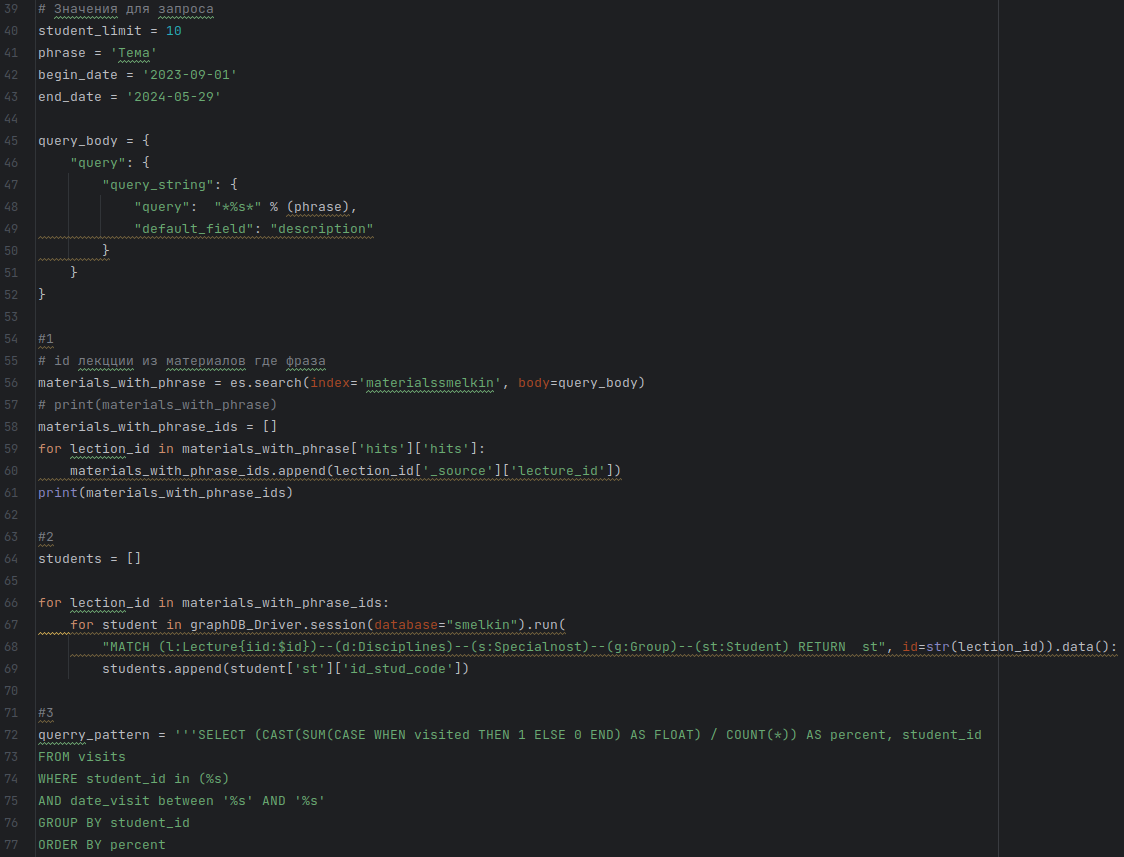
Москва 2023

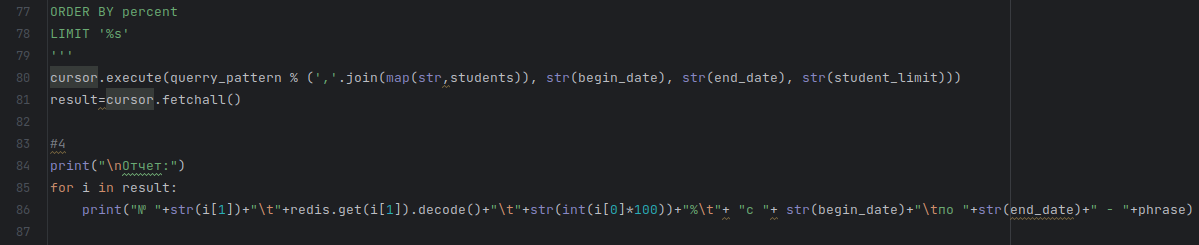
Задание:

Выполнить запрос к структуре хранения информации о группах учащихся, курсах обучения, лекционной программе и составу лекционных курсов и практических занятий, а также структуре связей между курсами, специальностями, студентами кафедры и данными о посещении студентами занятий, для извлечения отчета о 10 студентах с минимальным процентом посещения лекция, содержащих заданный термин или фразу, за определенный период обучения. Состав полей должен включать Полную информацию о студенте, процент посещения, период отчета, термин в занятиях курса.

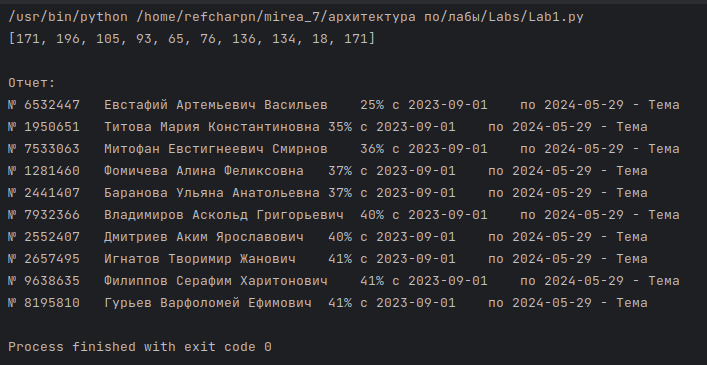
Код:







Результат работы(отчёт):



Обоснование структуры:

Для начала в программе инициализируются объекты для подключения к базам данных. Далее необходимо найти все занятия у которых в описании есть ключевое слово. Для этого лучше использовать elasticsearch, т.к. именно эта система предназначена для полнотекстового поиска. Был сформирован запрос для поиска всех объектов. Далее из полученных записей были извлечены идентификаторы(по умолчанию выводится 10 элементов, но можно вывести и все, добавив соответствующий параметр) и выведены на экран. Далее необходимо найти студентов, которые должны ходить на эти занятия. Для это использовалась система Neo4j, т. к. в ней легко искать сильно связанные данные, а занятия и студенты связаны через: занятия→ дисциплины → специализации → группы → студенты. В реляционных СУБД(postgresql) пришлось бы использовать join, что существенно замедлило бы выборку, более того, было бы задействовано куда больше сущностей, чем в neo(как было сказано в практике 3, некоторые таблицы превратились в связи). Также построение структуры запроса куда проще, чем в postgresql(достаточно указать через какие связанные сущности нужно искать). Для большей эффективности выбирались только идентификаторы. После этого производится расчёт процента посещаемости студента по id. Использовался postgresql, т.к. именно в pg содержится вся информация о посещении(в одной таблице) и достаточно легко рассчитать среднее значение для каждого идентификатора. Также сортируем значения по убыванию и лимитом, который указан в задании. Все данные получены, можно формировать отчёт. Для этого на экран выводится: номер билета студента, его ФИО(через redis, т.к. выборка такого количества данных займёт несколько мсек), процент посещаемости, с какого числа идёт выборка, по какое число идёт выборка, ключевое слово, по которому выбирались занятия.