



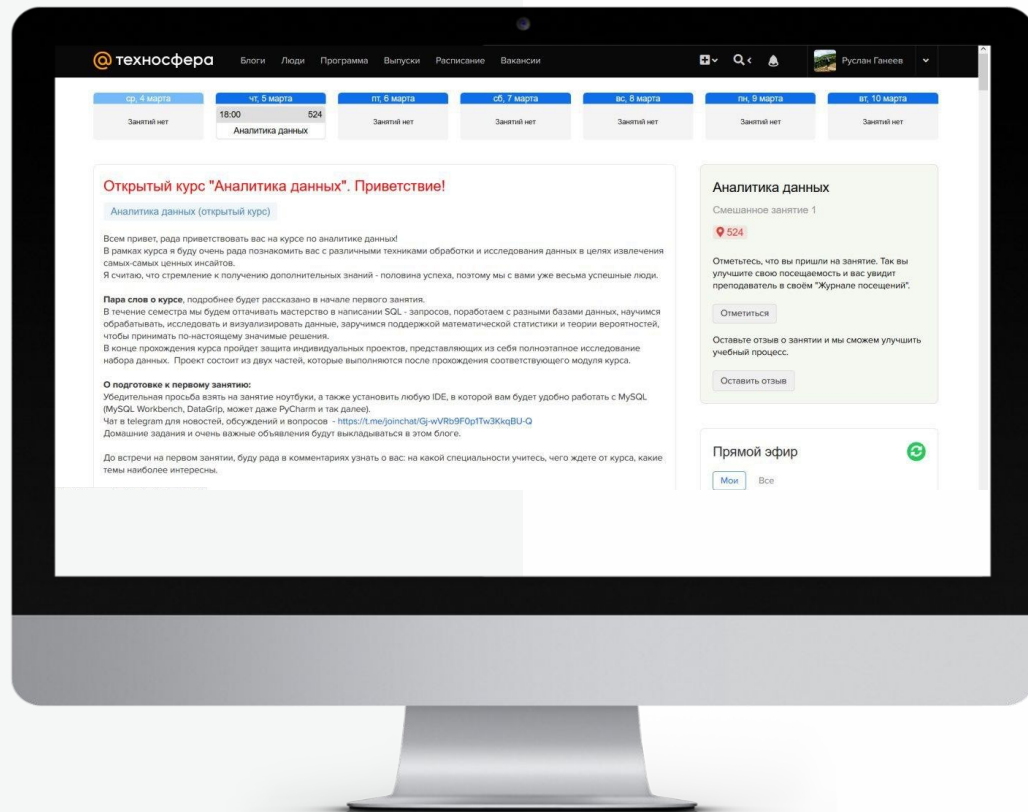
Backend разработка на Python

Лекция 11

Поисковые движки, Elasticsearch

Кандауров Геннадий





Напоминание отметиться на портале

+ отзывы после лекции

Квиз по прошлой лекции #10 (realtime)



Результаты квиза #9






**Потоки и процессы в python это синонимы, реальная
разница отсутствует?**

- Да
- Нет
- В python нет потоков



Семафор это механизм синхронизации, который

- 
- не разрешает менее чем N потокам выполнять некую секцию кода
 - не разрешает более чем N потокам выполнять некую секцию кода
 - разрешает более чем N потокам выполнять некую секцию кода
 - регулирует движение по железным дорогам




GIL используется в python и позволяет

- потоку не отдавать CPU обратно ОС по её требованию
- создавать потоки при возникновении CPU или I/O bound задачи
- только одному потоку использовать интерпретатор python
- распределить потоки по ядрам CPU для реального параллелизма



asyncio реализует следующий подход к многозадачности:

- кооперативная многозадачность
 - вытесняющая многозадачность
- 



Корутина может иметь

- один ВХОД, один ВЫХОД
- один ВХОД, несколько ВЫХОДОВ
- несколько ВХОДОВ, один ВЫХОД
- несколько ВХОДОВ, несколько ВЫХОДОВ



Асинхронщина позволяет

- Рассчитывать CPU задачи более эффективно
- Не блокировать поток на I/O bound задачах
- Распределять корутины одного потока по ядрам для одновременного выполнения
- Снизить негативный эффект переключения контекста по сравнению с потоками




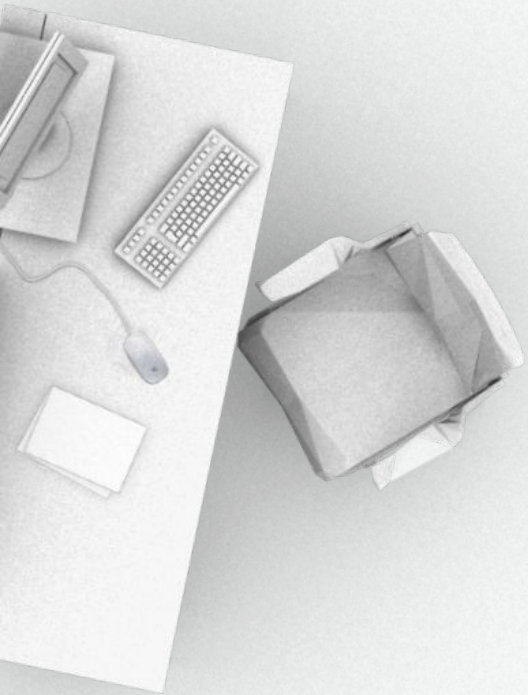
Синтаксис "async def" позволяет определять

- Функции, которые будут запущены на выполнение в фоне сразу при объявлении
- **Нативные корутины**
- Специальные итераторы для блокирующих вызовов
- Обычные функции, как и при использовании "def"



Содержание занятия

- Поисковые платформы
 - Elasticsearch
- 




Поисковые платформы

Поисковые платформы






Elasticsearch

- 
- Open source (1619 контрибьюторов на 14 мая 2021 г.)
 - Масштабируемость и отказоустойчивость
 - Удобный API (Restfull API)
 - Гибкие настройки
 - Динамический маппинг
 - Геопоиск
 - СЖК



Где используется Elasticsearch?

- 
- GitHub (поиск репозитория)
 - Uber
 - Microsoft (хранилище для MSN)
 - stackoverflow
 - ebay
 - docker (поиск репозитория)



Основные термины поисковых систем

- **Морфология**

Раздел грамматики, который оперирует формами слов.

- **Стемминг**

Приближённый эвристический процесс, в ходе которого от слов отбрасываются окончания в расчёте на то, что в большинстве случаев это себя оправдывает (running -> run).

- **Нечеткий поиск**

По заданному слову найти в тексте или словаре размера n все слова, совпадающие с этим словом (или начинающиеся с этого слова) с учетом k возможных различий.



Основные термины поисковых систем

- **Лемматизация**

Точный процесс с использованием лексикона и морфологического анализа слов, в результате которого удаляются только флексивные окончания и возвращается основная, или словарная, форма слова, называемая леммой (ran -> run).


- **N-грамма**

n каких-то элементов. Это более абстрактное понятие.

- **Стоп-слова**




Что мы получаем из коробки

- 
- Огромные возможности для поиска документа;
 - Около 50 видов агрегаций на все случаи жизни (максимальное, минимальное, среднее);
 - Гео-поиск;
 - Подсказки (suggester);
 - Гибкая работа и настройка всего, что есть в Elasticsearch;
 - И ещё много чего!




Elasticsearch концепты сверху

- 
- Нода
 - Кластер
 - Шард
 - Реплика



Elasticsearch концепты внутри

- 
- Индекс
 - Тип
 - Документ
 - Поле
 - Отображение (mapping)
 - Query DSL

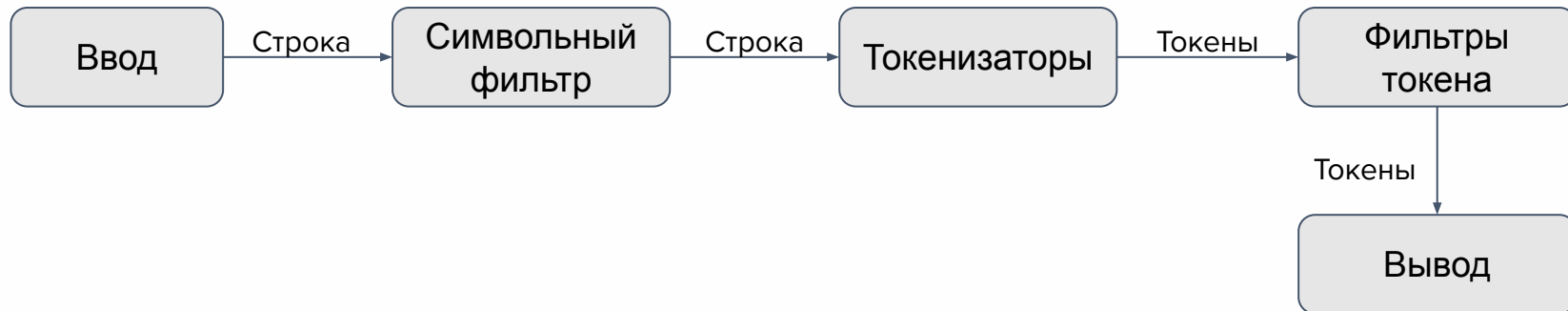


Elasticsearch концепты внутри

Мир реляционных БД	Elasticsearch
База данных (Database)	Индекс (Index)
Таблица (Table)	Тип (Type)
Запись (Row)	Документ (Document)
Колонка (Column)	Поле (Field)
Схема (Schema)	Отображение (Mapping)
SQL	Query DSL

Анализаторы

Цель - из входной фразы получить список токенов, которые максимально отражают её суть.



Расстояние Левенштейна

Минимальное количество операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для превращения одной строки в другую.

Цены операций могут зависеть от вида операции

- $w(a, b)$ — цена замены символа a на символ b
- $w(\epsilon, b)$ — цена вставки символа b
- $w(a, \epsilon)$ — цена удаления символа a

Частный случай задачи - Расстояние Левенштейна

- $w(a, a) = 0$
- $w(a, b) = 1$ при $a \neq b$ $w(\epsilon, b) = 1$
- $w(a, \epsilon) = 1$

Пример анализатора

```
PUT /your-index/_settings
{
  "index": {
    "analysis": {
      "analyzer": {
        "customHTMLSnowball": {
          "type": "custom",
          "char_filter": ["html_strip"],
          "tokenizer": "standard",
          "filter": ["lowercase", "stop", "snowball"]
        }
      }
    }
  }
}
```



Установка

<https://www.elastic.co/downloads/elasticsearch>

Ubuntu

```
apt install elasticsearch  
sudo -i service elasticsearch start
```

MacOS

```
brew install elasticsearch  
brew services start elasticsearch
```


Нужно установить Java \geq version 7.

<http://localhost:9200/>

```
pip install elasticsearch
```



Mappings



```
PUT my_index
{
  "mappings": {
    "_doc": {
      "properties": {
        "title": {"type": "text"},
        "name": {"type": "text"},
        "age": {"type": "integer"},
        "created": {
          "type": "date",
          "Format": "strict_date_optional_time||epoch_millis"
        }
      }
    }
  }
}
```

Создание индекса

Создание индекса

PUT http://localhost:9200/blogs

```
{
  "settings": {
    "index": {
      "number_of_shards" : 5,
      "number_of_replicas" : 3
    }
  }
}
```

Создание и заполнение индекса

Заполнение индекса пачкой

```
POST http://localhost:9200/blogs/_bulk
{ "index":{"_index":"blogs", "_type":"posts", "_id":"10"} }
{ "title":"Test1", "description":"First test description" }
{ "index":{"_index":"blogs", "_type":"posts", "_id":"11"} }
{ "title":"Test2", "description":"Second test description" }
```

или

```
POST http://localhost:9200/blogs/post/
{ "title":"Test3", "description":"Third test description" }
```

Получение результатов

Получение по id

GET http://localhost:9200/blogs/posts/1

Поиск по индексам index1, index2, index3 и по полю

GET http://localhost:9200/index1,index2,index3/_search

```
{
  "query" : {
    "match" : { "title": "test" }
  }
}
```

Поиск по определённому полю

GET http://localhost:9200/_search?q=name:central

Синтаксис запросов

+ signifies **AND** operation

| signifies **OR** operation

- negates a single token

" wraps a number of tokens to signify a phrase for searching

* at the end of a term signifies a prefix query

(and) signify precedence

~N after a word signifies edit distance (fuzziness)

~N after a phrase signifies slop amount

Внедряем в приложение. Вариант 1

```
from elasticsearch import Elasticsearch

es = Elasticsearch()
es.indices.create(index='my-index', ignore=400)
es.index(index="my-index", id=42, body={"any": "data", "timestamp": datetime.now()})
{'_index': 'my-index',
 '_type': '_doc',
 '_id': '42',
 '_version': 1,
 'result': 'created',
 '_shards': {'total': 2, 'successful': 1, 'failed': 0},
 '_seq_no': 0,
 '_primary_term': 1}

es.get(index="my-index", id=42)['_source']
```


Внедряем в приложение. Вариант 2

```
from rest_framework_elasticsearch import es_views, es_pagination, es_filters
class BlogView(es_views.ListElasticAPIView):
    es_client = es_client
    es_model = BlogIndex
    es_pagination_class = es_pagination.ElasticLimitOffsetPagination
    es_filter_backends = (
        es_filters.ElasticFieldsFilter,
        es_filters.ElasticFieldsRangeFilter,
        es_filters.ElasticSearchFilter,
        es_filters.ElasticOrderingFilter,
        es_filters.ElasticGeoBoundingBoxFilter
    )
```

Внедряем в приложение. Вариант 2. Продолжение

```
class BlogView(es_views.ListElasticAPIView):
    ...
    es_ordering = 'created_at'
    es_filter_fields = (es_filters.ESFieldFilter('tag', 'tags'),)
    es_range_filter_fields = (es_filters.ESFieldFilter('created_at'),)
    es_search_fields = ( 'tags', 'title', )
    es_geo_location_field = es_filters.ESFieldFilter('location')
    es_geo_location_field_name = 'location'
```

Внедряем в приложение. Вариант 3

```
# documents.py
from django_elasticsearch_dsl import Document
from django_elasticsearch_dsl.registries import registry
from .models import Car


@registry.register_document
class CarDocument(Document):
    class Index:
        name = 'cars'
        settings = {'number_of_shards': 1,
                    'number_of_replicas': 0}
    ...
```

Внедряем в приложение. Вариант 3

```
# ... продолжение
class Django:
    model = Car # The model associated with this Document
    # The fields of the model you want to be indexed in Elasticsearch
    fields = [
        'name',
        'color',
        'description',
        'type',
    ]
```



Внедряем в приложение. Вариант 3



```
./manage.py search_index --rebuild  
s = CarDocument.search().filter("term", color="blue")[:30]  
qs = s.to_queryset()
```



Домашнее задание по лекции 11

ДЗ #11

03.06.202

срок сдачи

- Написать функцию, которая будет считать расстояние Левенштейна между двумя словами;
- Развернуть и заполнить тестовыми данными Elasticsearch;
- Реализовать поиск по пользователям, продуктам (сущностям);
- Реализовать метод API для поиска по указанным сущностям и отображения результатов.

**СПАСИБО
ЗА ВНИМАНИЕ**

