西北工業大學
NORTHWESTERN POLYTECHNICAL UNIVERSITY

西 北 工 业 大 学
专业实习报告总结

学　　号	2020302598
班　　级	14012001
姓　　名	王旻安
实习项目	西安各商圈餐饮分析系统
实习单位	四川华迪公司
日　　期	2023.2.16

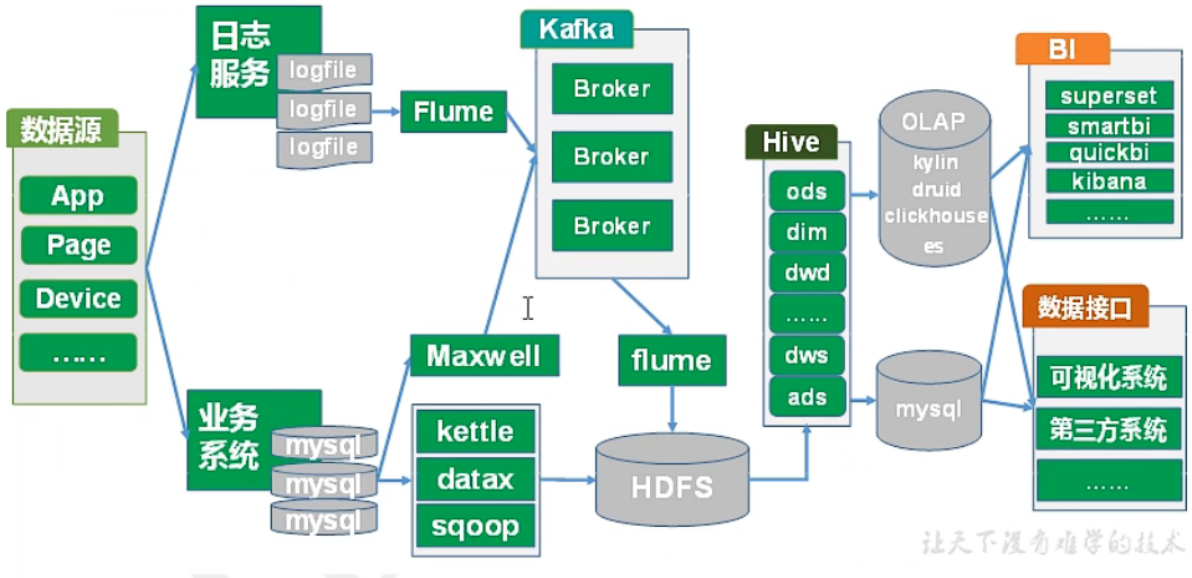
王旻安实习总结报告

0. 概述

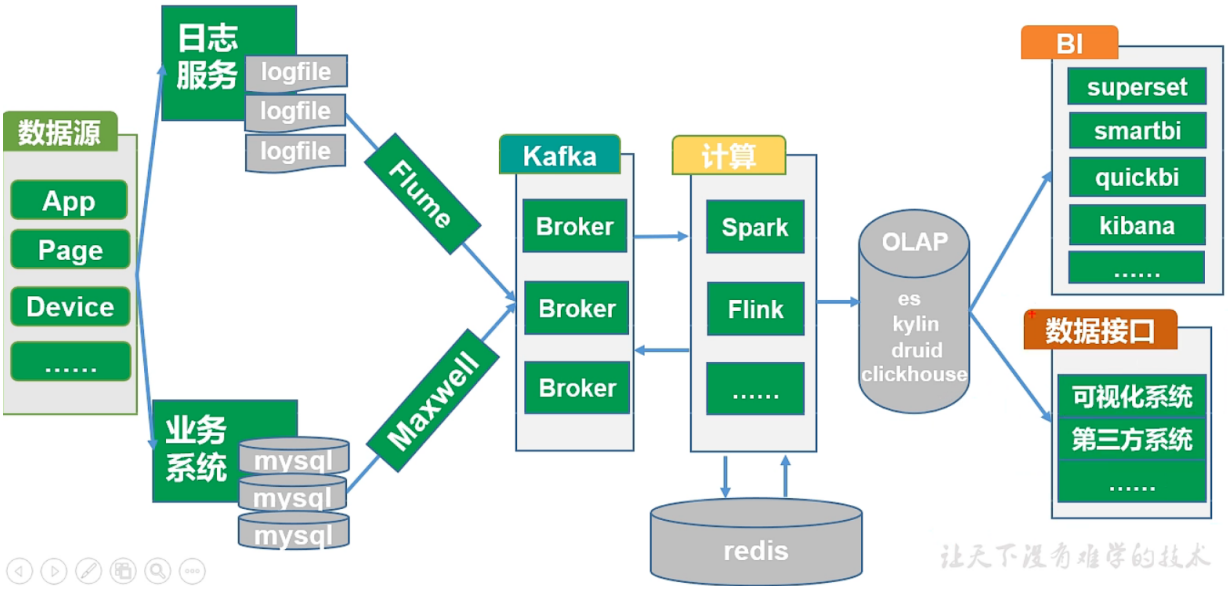
本次实训的主题为大数据商业项目开发实训。作为我校软件学院学生实习的主题这显然是不妥的。我校软件学院大三开设的三个方向并没有大数据，我们在前两年的学习中也鲜有相关方向内容的涉及。当时选择实训机构摸底时也未征求同学对于项目选择上的意见。种种因素最终导致了本次实训从开始就起不到需要的效果。希望学院在之后的主题选择上注意。

1. 技术选型与架构设计

我们了解到传统的数仓设计分为离线数仓和实时数仓两种。离线数仓用于大量数据的处理，处理时间长，运算实时性要求不高。所以主要运用的技术为HDFS等Apache系的数据湖工具。



实时数仓对系统的延迟要求较低，要求从前端实时获取数据，同时具有较短的读写延迟。所以主要运用到的处理工具为Spark与Redis。

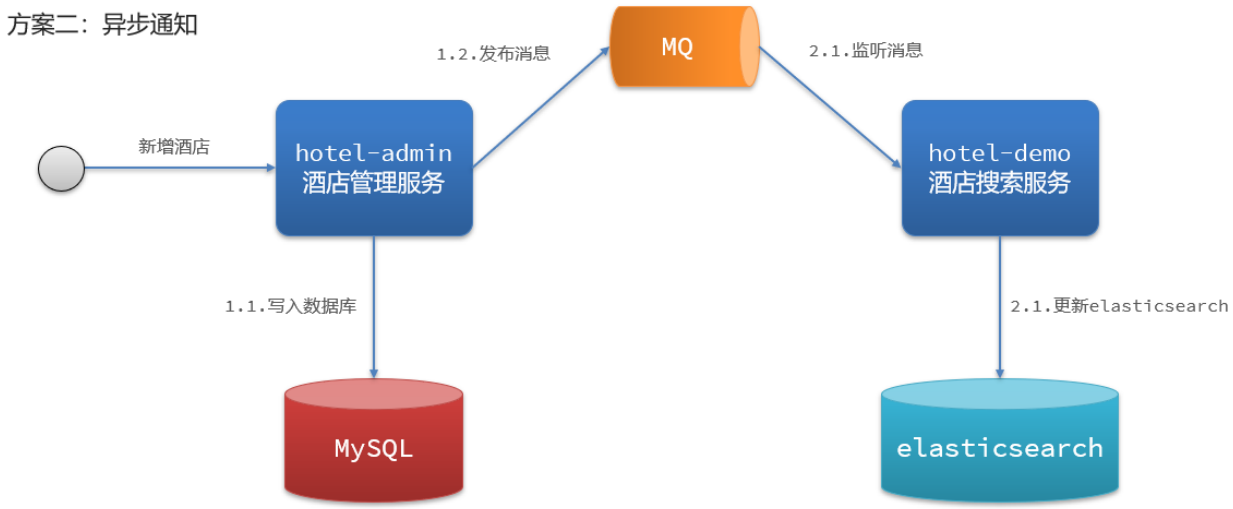


上面两种数仓都对我们而言都存在难以实现的问题，主要困难有：

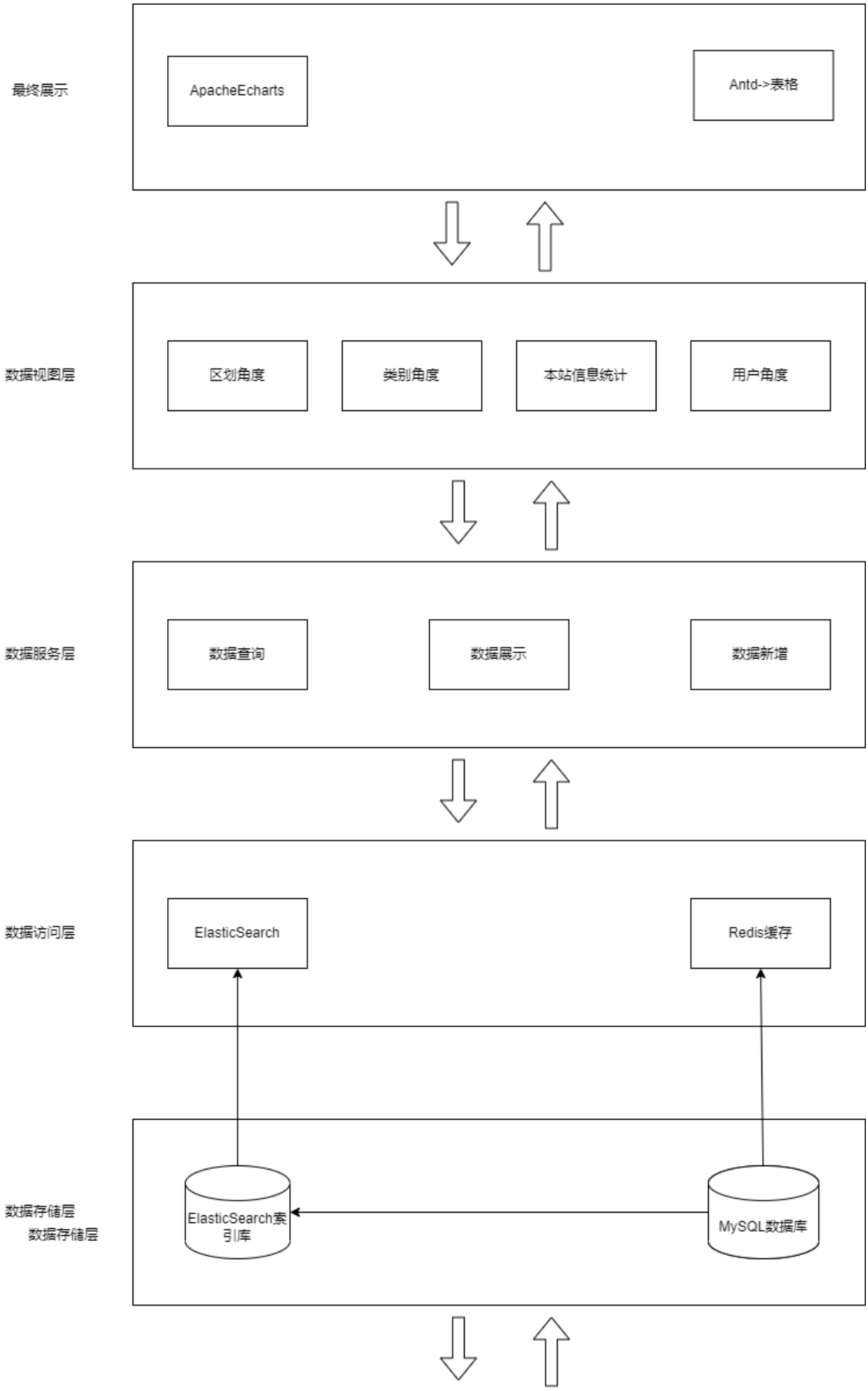
- 我们完全不熟悉数仓中的工具，包括但不限于Apache HDFS全家桶，Maxwell，Flink等等。
- 我们的爬虫完全无法做到实时获取数据，需要大量的手工操作介入。(详见具体实现)

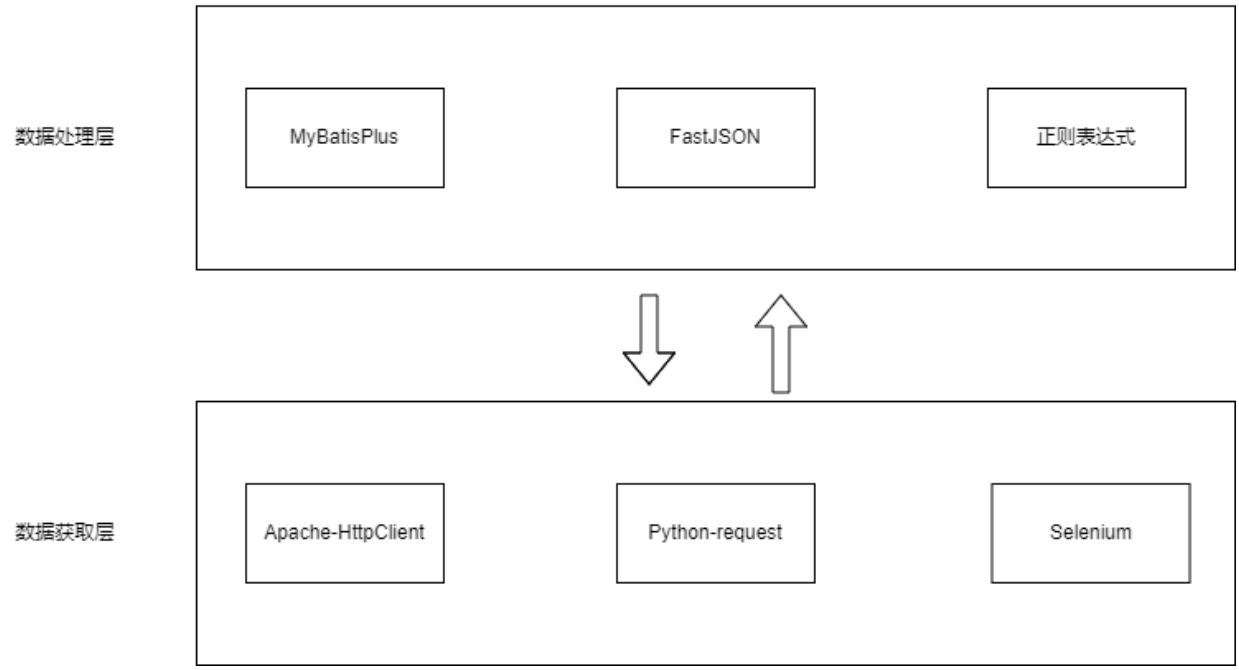
所以上述两种数仓的实现架构在我们看来都是不可行的。只能寻找替代的解决方案。

在寒假期间我学习了黑马程序员的微服务课程。其基础篇的随堂项目为黑马旅游，使用ElasticSearch实现酒店查询。其技术架构如下图所示。这个方案的右半部分最终被采纳，郑炜我们项目实现的部分。



我们最终确定的技术架构如下


最终展示
ApacheEcharts
Antd->表格
数据视图层
区划角度
类别角度
本站信息统计
用户角度
数据服务层
数据查询
数据展示
数据新增
数据访问层
ElasticSearch
Redis缓存
数据存储层
ElasticSearch索引库
MySQL数据库
数据存储层



2. 爬虫

说得好听叫爬虫，实际上我们数据获取的方式非常之原始。与携程等不需要登录就能获取大量数据的网页不同，在美团网页上的所有操作都需要经过鉴权。同时我们能明显地感知到美团对于请求数量有自己的监控方式。在请求过高时会在用户登录时添加更多校验，如补全之前使用美团时下的订单号，选择使用美团服务到访过的商圈名称。

在实训开始实际操作前的培训中，我们完全没有讲过应对这样的情况应该如何处理。所有爬虫的课程只会讲豆瓣——豆瓣也讲不全——只讲豆瓣的电影TOP250.这仅仅是个爬虫的入门罢了。

我最终参考了Bilibili上的一些美团教学。选取了https://xa.meituan.com/meishi/api/poi/getPoiList作为有限数据的获取接口，使用python中的request库作为请求库向美团服务器发送请求。

```
def spider_crawl(page_num: string, cate_id: string, area_id: string):
    url = 'https://xa.meituan.com/meishi/api/poi/getPoiList'
    # (.*?):(.*)  ->  '$1': '$2'
    data = {
        'cityName': '西安',
        'cateId': cate_id,
        'areaId': area_id,
        'sort': '',
        'dinnerCountAttrId': '',
        'page': page_num,
        'userId': '876385598',
        'uuid': '639fcb29c3ff4c2392e8.1676422834.1.0.0',
        'platform': '1',
        'partner': '126',
        'originUrl': 'https%3A%2F%2Fxa.meituan.com%2Fmeishi%2Fpn' + page_num +
'%2F',
        'riskLevel': '1',
        'optimusCode': '10',
```

```
        '_token':
'eJxVjktvqkAYhv/LbCHCAAOOSRfCsQotcikqpOmCojLIXYaLNv3vZ05jF2f1Xr7nS94vcDWPYAFFEYsiD4b
TFSwAnIkzFfCAduyiaqoiyRCLczjnQfJ/hxF7+rzu/4DFuyYpvAaVj3+Fz/I7RLLKz1XWPCxUP3gGST+MyRB
AKG26hSBM8aw8ZbSPq1lSlwLzHcmEBEKBzQAMLwOGM8OfGj+U/mab7WZsl6UVcydr3F8SSJe3lUfCYYoidRz
19p4HZOXmh5fzxtaJJcHcPOajvORW7Hkh3hptXe7s3HIO7AYcLlHao4KmW8XHRDdcQ3dzpa8HiM8unp8TM9l
11u1uyL4NN42ZpJ/keGntBNeTZyCyoibSXuMX/Hyo7pngTNGO2zgW5O8IyfVWXg56FDph2DdvrY1jM9h3rWo
XBb7QVX+lzOc8afvpbno9VQaucg6kwMNgeWtKaNFsM5kLcnh+tau2sepybYSyVyN/fdndxtSTNFfiurcxfwL
ffwFG4pLG'
        }
    headers = {
        'Accept': 'application/json',
        'Accept-Encoding': 'gzip, deflate, br',
        'Accept-Language': 'zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6',
        'Cache-Control': 'no-cache',
        'Connection': 'keep-alive',
        'Cookie': '_lxsdk_cuid=185800e34c5c8-08b84285c3d6f5-26021151-144000-
185800e34c6c8; WEBDFPID=w0xz8v050899572uyzy3504z3w9v6w5w814199v570x979588uy8z0v2-
1988250693759-1672890693004SEEAWICfd79fef3d01d5e9aadc18ccd4d0c95071372; ci=42;
rvct=42; _hc.v=f9f8afb3-0904-1d77-2134-4f2c91221449.1672890846;
iuuid=00786FB10D3457523E34EF26842EB1372AA68F83C0060CF56BA30C2AA072E60E;
_lxsdk=00786FB10D3457523E34EF26842EB1372AA68F83C0060CF56BA30C2AA072E60E;
__mta=216497705.1672890710892.1673359740975.1673361948597.22; mtcdn=K; client-
id=f90ccc59-f3bf-424b-b431-9587a2e0ac33; uuid=639fcb29c3ff4c2392e8.1676422834.1.0.0;
userTicket=HuSeRcoGseVIsaeeJfFbfbwmBelnkOgNsQyPqtwy;
_yoda_verify_resp=VWmKCx88JlgeqoIAM5enHdw2Y3DDkV2fKoVSjKu7nY5BXt%2Bppn8vSJmrQOGSDAbq
JAzCXw9bO0vaKzImtXM916jq9NJlqSJYInKPrp3fT2ptfALrUiIokaJUsx0yQlccYZ%2FpOzi7P6ZLxu4Sp6
c0BefvG0SBapok%2FexJVNUjMjl2TU2uOYv7mHtVN4zxCDHNvVHS8i8O2Q6avOxmDB4Vz3anRE7qOesmzYJY
Xe1nSPXJ2dteOwKYTGZs87CPeRhf4DJmDqko3nUhtI49YPdNijuwFLG6ZjEm3fg3BD6e2rdv2xB8ZNtnDvxA
%2Fuo35W%2FKZweenuFaMVmQH4GTjUbCxi9KBMFWew3hHSMHEmZnxWW2LavDnO7V%2Bg%2Fr8ym0SdV6;
_yoda_verify_rid=168f829fcb8350 4d; u=876385598; n=%E5%AF%82%E7%8E%84%E6%A5%9A;
lt=AgEYJDOCaRVz4eO-JIQrXsu4cVVt5z_v0K8KQylHyQt-
3XfnC0Pvq_VQyLrIFMlqQSLCvV7gqoFMgwAAAAC9FgAAoLlZ7Ee8T91ln3bAI4DtVp4hLCaNfgJOejiMtyzL
yP0S0RkHb7bszejDBskgdVdp; mt_c_token=AgEYJDOCaRVz4eO-JIQrXsu4cVVt5z_v0K8KQylHyQt-
3XfnC0Pvq_VQyLrIFMlqQSLCvV7gqoFMgwAAAAC9FgAAoLlZ7Ee8T91ln3bAI4DtVp4hLCaNfgJOejiMtyzL
yP0S0RkHb7bszejDBskgdVdp; token=AgEYJDOCaRVz4eO-JIQrXsu4cVVt5z_v0K8KQylHyQt-
3XfnC0Pvq_VQyLrIFMlqQSLCvV7gqoFMgwAAAAC9FgAAoLlZ7Ee8T91ln3bAI4DtVp4hLCaNfgJOejiMtyzL
yP0S0RkHb7bszejDBskgdVdp; token2=AgEYJDOCaRVz4eO-JIQrXsu4cVVt5z_v0K8KQylHyQt-
3XfnC0Pvq_VQyLrIFMlqQSLCvV7gqoFMgwAAAAC9FgAAoLlZ7Ee8T91ln3bAI4DtVp4hLCaNfgJOejiMtyzL
yP0S0RkHb7bszejDBskgdVdp; _lxsdk_s=1865296670e-9eb-b9a-88f%7C%7C19',
        'Host': 'xa.meituan.com',
        'Pragma': 'no-cache',
        'Referer': 'https://xa.meituan.com/meishi/c229/',
        'sec-ch-ua': '"Not?A_Brand";v="8", "Chromium";v="108", "Microsoft
Edge";v="108"',
        'sec-ch-ua-mobile': '?0',
        'sec-ch-ua-platform': '"Windows"',
        'Sec-Fetch-Dest': 'empty',
        'Sec-Fetch-Mode': 'cors',
        'Sec-Fetch-Site': 'same-origin',
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36'
    }
```

```
    response = requests.get(url, params=data, headers=headers)
    # 如果404了
    if response.status_code == 404:
        return "404"
    return response.text
```

上述请求头中会随着登录变化的数据为`data`中的`uuid,_token`以及`headers`中的`cookie`。在没有有关密码学的基础的情况下，这三个数据只有通过重新登陆获取。

根据这样的情况，我们最终拿到的数据是典型的json字符串。包含及其有限的店铺信息。情况如下


```
{
    "status": 0,
    "data": {
        "totalCounts": 793,
        "poiInfos": [
            {
                "poiId": 186246617,
                "frontImg":
"https://img.meituan.net/600.600/msmerchant/b34af837149d711da28b4f7d822a601b184324.j
pg",
                "title": "兔头一绝（东关店）",
                "avgScore": 4.4,
                "allCommentNum": 25,
                "address": "东门外",
                "avgPrice": 44,
                "dealList": [
                    {
                        "title": "特色菜品2选1，提供免费WiFi",
                        "price": 12.8,
                        "soldCounts": 0
                    }
                ],
                "hasAds": false,
                "adsClickUrl": "",
                "adsShowUrl": ""
            }
        ]
    }
}
```


结合爬取时在请求头中加入的地点或分类信息，我们能获取到一家店的基本情况。

3. 数据整合与清洗

数据清洗的主要工作非由我完成。

最终实现的数据持久化SQL-DDL如下

```
DROP TABLE IF EXISTS `food`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
```

```
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `food` (
  `poiId` varchar(100) NOT NULL,
  `frontImg` varchar(100) NOT NULL,
  `title` varchar(100) NOT NULL,
  `category` varchar(100) DEFAULT NULL,
  `district` varchar(100) NOT NULL,
  `businessDistrict` varchar(100) NOT NULL,
  `allCommentNum` int NOT NULL,
  `avgScore` float NOT NULL,
  `avgPrice` int NOT NULL,
  `longitude` double NOT NULL,
  `latitude` double NOT NULL,
  PRIMARY KEY (`poiId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

以上是我们的数据库中最终包含的字段。

4. 数据索引与转储

在建立索引时，我们选用kibana发送RESTful的请求。https://kibana.wangminan.me

建立索引的语句如下

```
PUT /meituan
{
  "settings": {
    "analysis": {
      "analyzer": {
        "text_anlyzer": {
          "tokenizer": "ik_max_word"
        },
        "completion_analyzer": {
          "tokenizer": "keyword",
          "filter": "py"
        }
      },
      "filter": {
        "py": {
          "type": "pinyin",
          "keep_full_pinyin": false,
          "keep_joined_full_pinyin": true,
          "keep_original": true,
          "limit_first_letter_length": 16,
          "remove_duplicated_term": true,
          "none_chinese_pinyin_tokenize": false
        }
      }
    }
  },
  "mappings" : {
    "properties": {
```

```
      "poiId": {
        "type": "keyword"
      },
      "title": {
        "type": "text",
        "analyzer": "text_anlyzer",
        "search_analyzer": "ik_smart",
        "copy_to": "all"
      },
      "avgPrice": {
        "type": "integer"
      },
      "avgScore": {
        "type": "long"
      },
      "allCommentNum": {
        "type": "integer"
      },
      "category": {
        "type": "keyword",
        "copy_to": "all"
      },
      "district": {
        "type": "keyword",
        "copy_to": "all"
      },
      "businessDistrict": {
        "type": "keyword",
        "copy_to": "all"
      },
      "frontImg": {
        "type": "keyword",
        "index": false
      },
      "location":{
        "type": "geo_point"
      },
      "all":{
        "type": "text",
        "analyzer": "text_anlyzer",
        "search_analyzer": "ik_smart"
      },
      "suggestion":{
          "type": "completion",
          "analyzer": "completion_analyzer"
      }
    }
  }
}
```

采用了`ik_max_word`搭配`pinyin`分词器做数据处理。其中前者负责数据检索，后者参与输入联想。

完成索引建立后使用MyBatisPlus从MySQL中导出数据，转储到ElasticSearch。

```
@Test
void testBulkRequest() throws IOException {
    // 查询所有的酒店数据
    List<Food> list = foodService.list();

    // 1.准备Request
    BulkRequest request = new BulkRequest();
    // 2.准备参数
    for (Food food : list) {
        // 2.1.转为FoodDoc
        FoodDoc foodDoc = new FoodDoc(food);
        // 2.2.转json
        String json = JSON.toJSONString(foodDoc);
        // 2.3.添加请求
        request.add(new
IndexRequest(MEITUAN_INDEX_NAME).id(food.getPoiId()).source(json,
XContentType.JSON));
    }

    // 3.发送请求
    client.bulk(request, RequestOptions.DEFAULT);
}
```

5. 数据整合与搜索

使用`Aggregations`对数据进行聚合。ElasticSearch的`RestHighLevelClient`和MyBatisPlus的语法还是非常相似的。可以指定字段进行搜索、聚合、累加等。对比MySQL的Select，Sum与GroupBy语法还是非常容易实现的。就当是复习和巩固寒假学习的ES语法了。下面分别举一个聚合和搜索的样例。

- 聚合

```
/**
 * 根据区名获取商户数量
*/
@Override
public R getMerchantNumberByDistrict() {
    // 发送搜索请求
    SearchResponse response =
            getSearchResponse(QueryBuilders.matchAllQuery(),
                    DISTRICT_AGGREGATION_NAME, DISTRICT_FIELD_NAME);
    // 解析聚合结果
    Aggregations aggregations = response.getAggregations();
    Map<String, Integer> aggByDistrict = getDistrictDtoCount(aggregations);
    return R.ok().put(RESULT, aggByDistrict);
}
```

- 搜索

```
/**
 * 响应搜索请求
*/
@Override
public R query(QueryDto queryDto) {
    try {
        // 1.准备Request
        SearchRequest request = new SearchRequest(MEITUAN_INDEX_NAME);
        // 2.准备请求参数
        // 2.1.query 加入请求参数
        buildBasicQuery(queryDto, request);
        // 2.2.分页
        int page = queryDto.getPageNum();
        int size = queryDto.getPageSize();
        request.source().from((page - 1) * size).size(size);
        // 3.发送请求
        SearchResponse response = restHighLevelClient.search(request,
RequestOptions.DEFAULT);
        // 4.解析响应 fastJson
        PageResult pageResult = handlePageResponse(response);
        Map<String,Object> result = new HashMap<>();
        result.put(RESULT, pageResult);
        return R.ok(result);
    } catch (IOException e) {
        throw new SearchException("搜索失败");
    }
}
```

6. 总结

项目类别一出来就觉得这可能不是一个适合我的项目。实际上也确实。有这时间我可能更希望先复习一下开学考，或者去跟着做一下黑马的微服务项目，把Java并发编程JUC课看了，然后巩固一下我的前端知识，学一下vite和electron。真不知道花了这么多时间写个这玩意还要做组内协调是在干什么。