Грибанов Данил, 6233

Ход работы

Apache Nifi

Полная схема в системе NiFi представлена на рисунке 1.

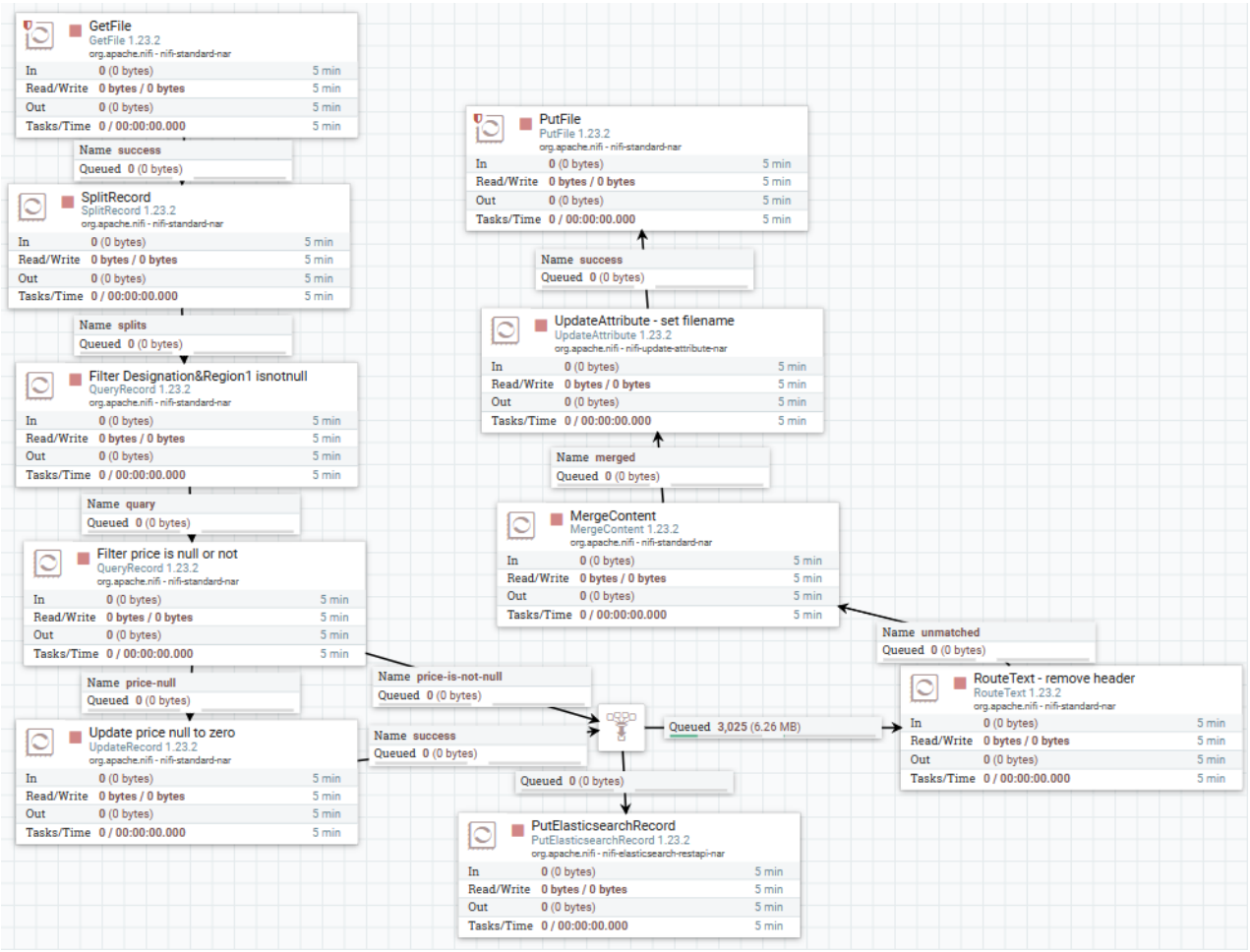

Рисунок 1 – Схема в системе NiFi

По схеме и с учетом названий смысл компонентов описывается. Однако есть момент с RouteText и MergeContent. Одна из проблем, что MergeContent соединяет csv данные дублируя шапку (header), поэтому итоговый csv файл имеет дубликаты шапки внутри файла как итог.

Основная задача RouteText удалить шапку везде, и при этом в MergeContent шапка вставляется сама, через следующие настройки на рисунке 2. MergeContent позволяет вручную вставить шапку, дабы нивелировать данную проблему. Однако, когда в UpdateAttribute указывается filename и далее PutFile сохраняет файл, происходит немного странное с тем, что шапка соединяется с первой строкой записи (см. рисунок 3). Основное решение данной проблемы, это вставление новой строки в шапку, указанной на рисунке 2. Однако я не понял, как это сделать, и такие вставки в конец как ${‘\n’}, ‘\n’, ${\n} – не помогли.

Также про сохранение файла с именем. У файла filename берется из GetFile операции, но необходимо задавать ему какое-то более значимое имя. Я сделал так: идентификатор имени и случайное число (см. рисунок 4). Была проблема с тем, что файл мог создаваться несколько раз с одним именем, из-за чего NiFi выдавал ошибку, ибо такое делать нельзя. То есть имя должно быть уникальным. При этом идентификатор для разных сегментов одного файла – одинаковый, поэтому я присоединяю еще случайное число дабы была некоторая уникальность.

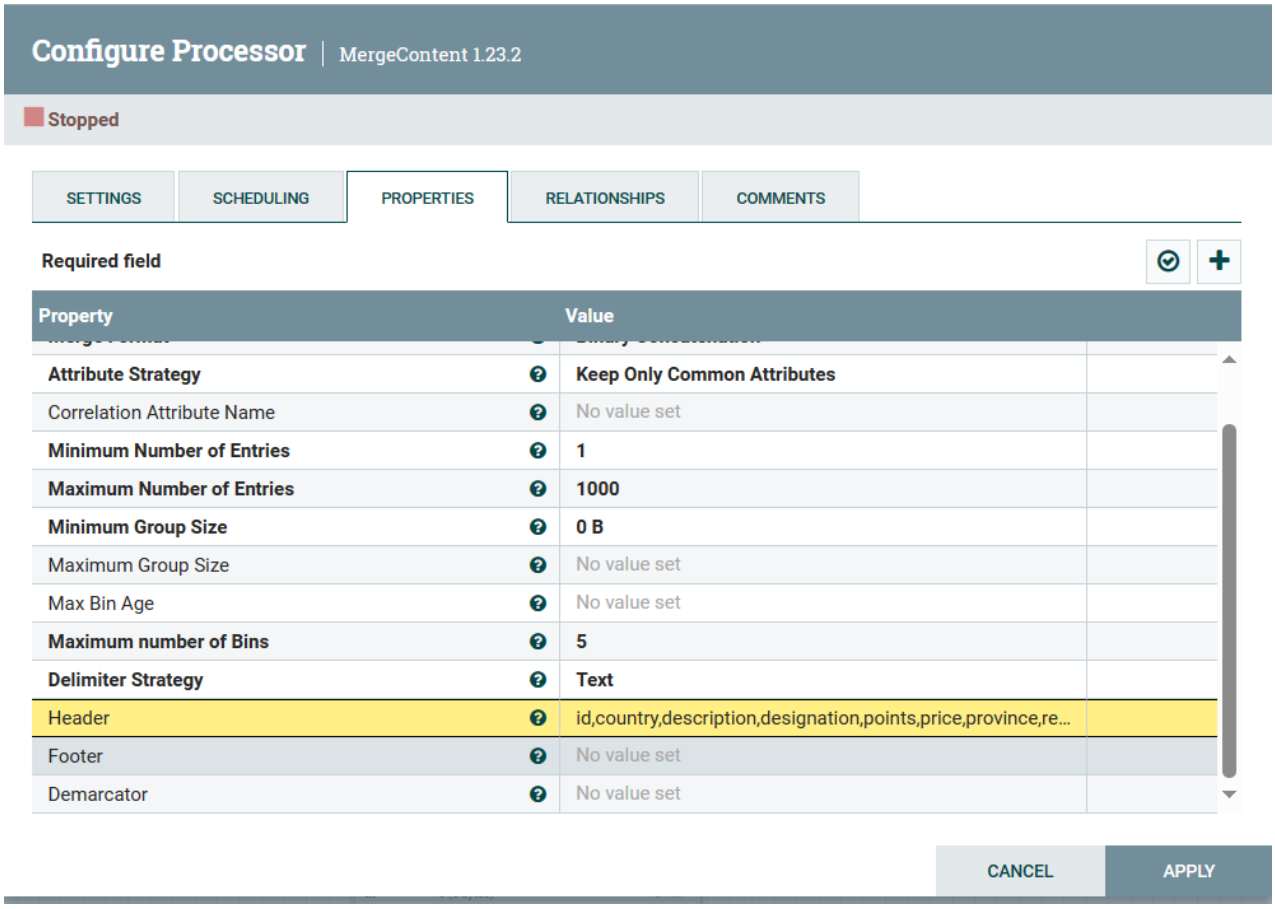

Рисунок 2 – Настройки MergeContent

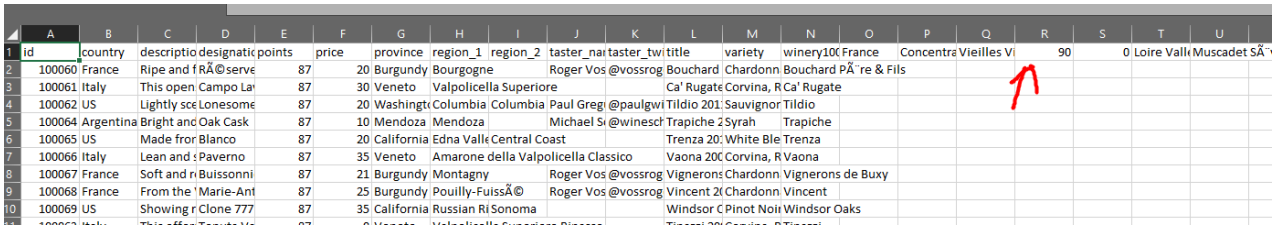

Рисунок 3 – Отсутствие новой строки при соединении шапки и строк файла

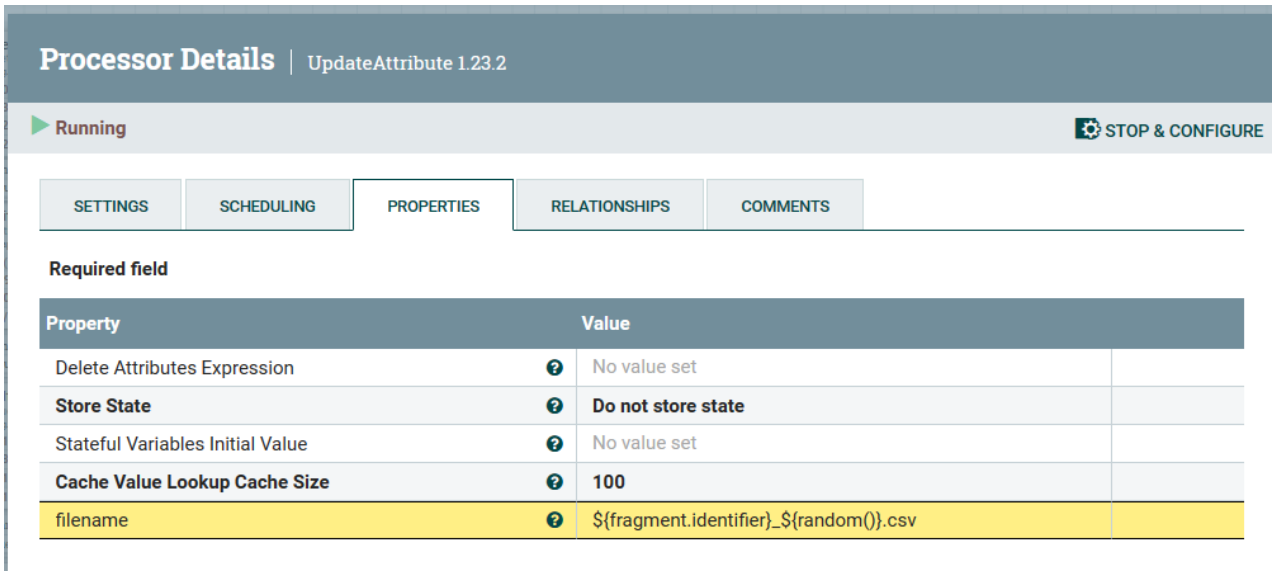

Рисунок 4 – Установка нового имени файла для сохранения

В NiFi создается соответствующий компонент PutElasticsearchRecord для отправки отфильтрованных данных (см. рисунок 5). Также необходима настройка сервиса для отправки (см. рисунок 6).

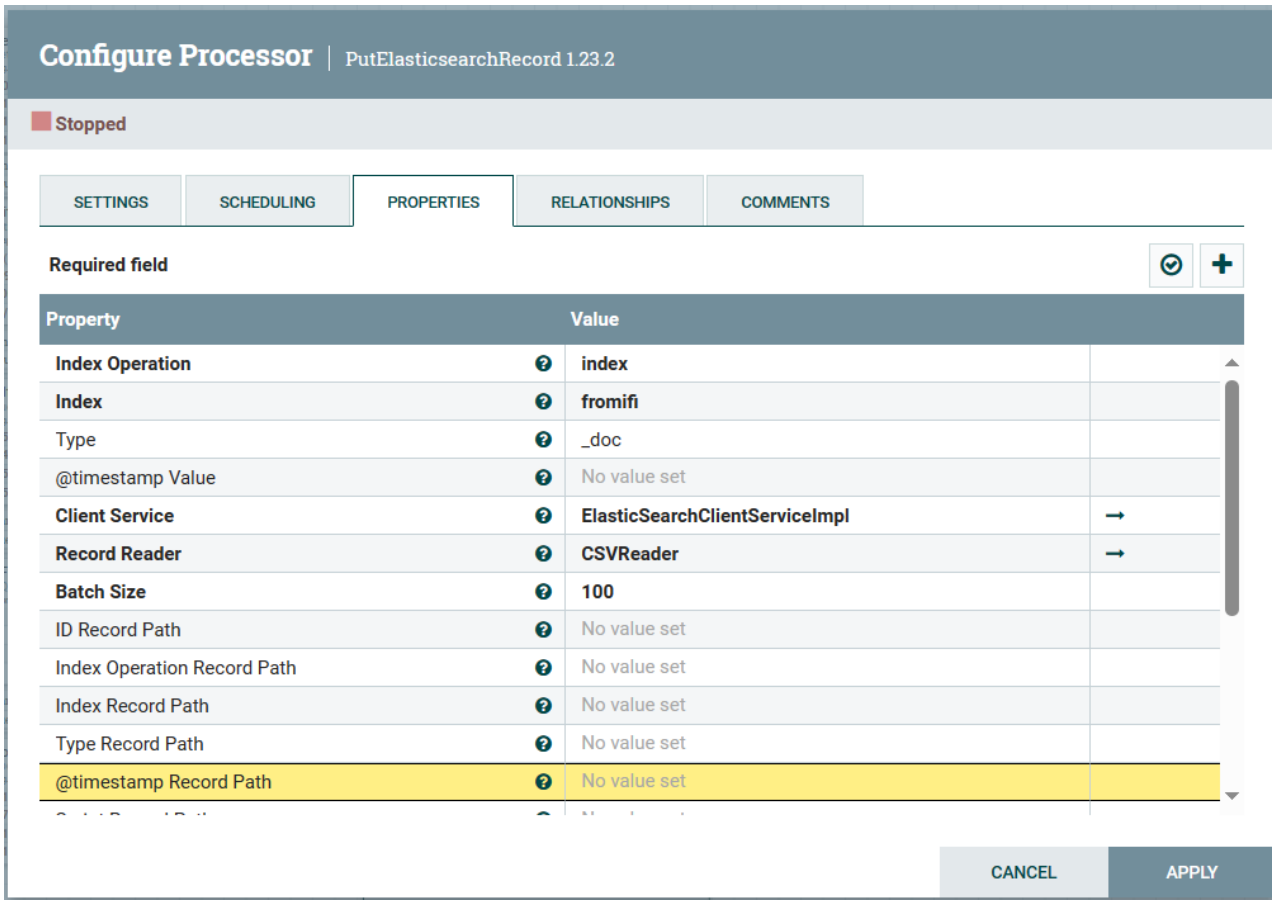

Рисунок 5 – Создание PutElasticsearchRecord

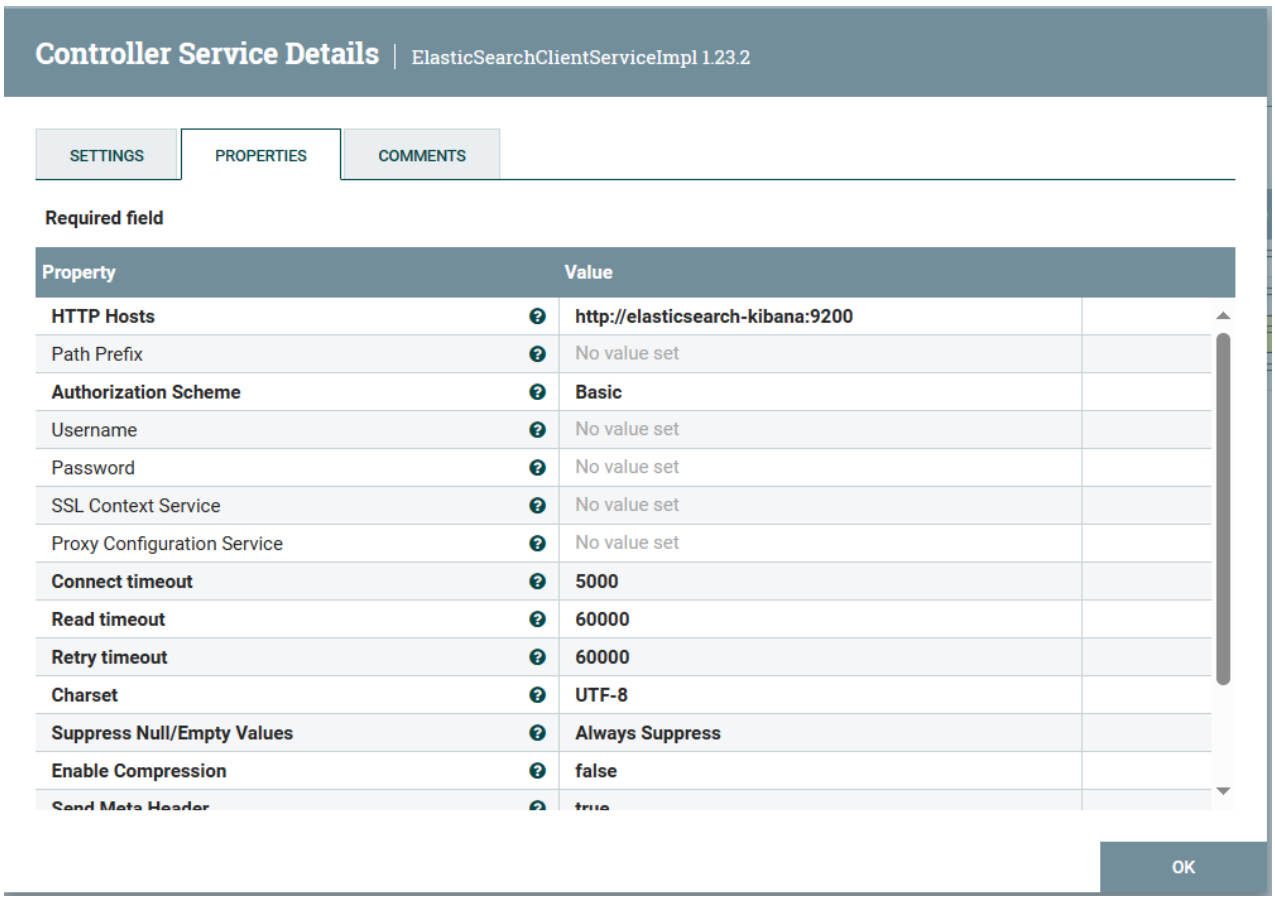

Рисунок 6 – Настройка сервиса для отправки

Теперь, отправим данные в данный сервис. В сервисе увидим, что данные пришли (см. рисунок 7).

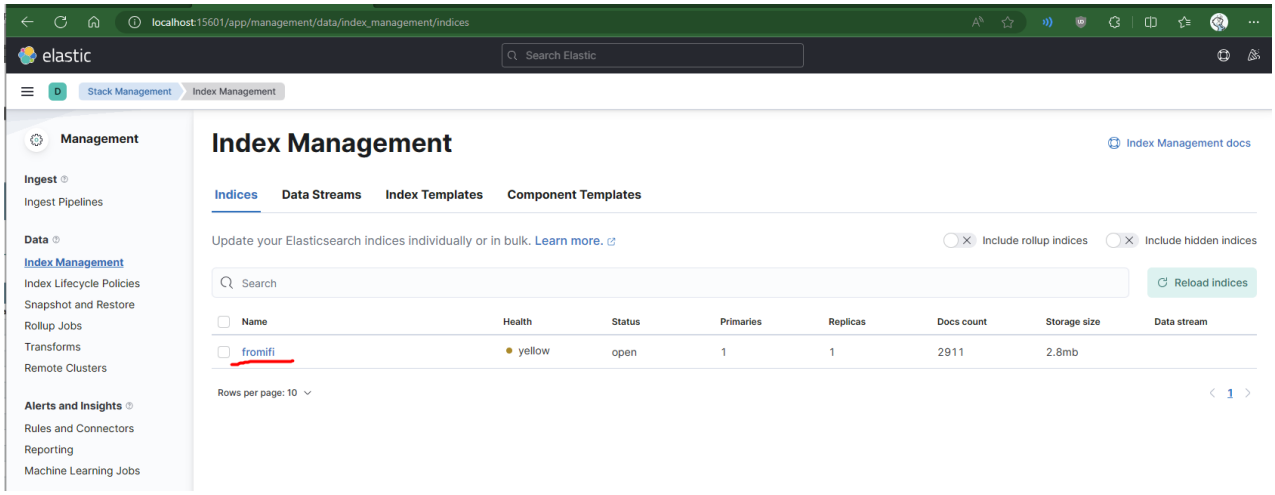

Рисунок 7 – Присланные данные из NiFi

Теперь в вкладке Kibana нужно для этого индекса создать индекс паттерн (см. рисунок 8), где указаны поля данных.

Теперь с данными созданным индексом паттернов можно приступать к визуализации (см. рисунок 9).

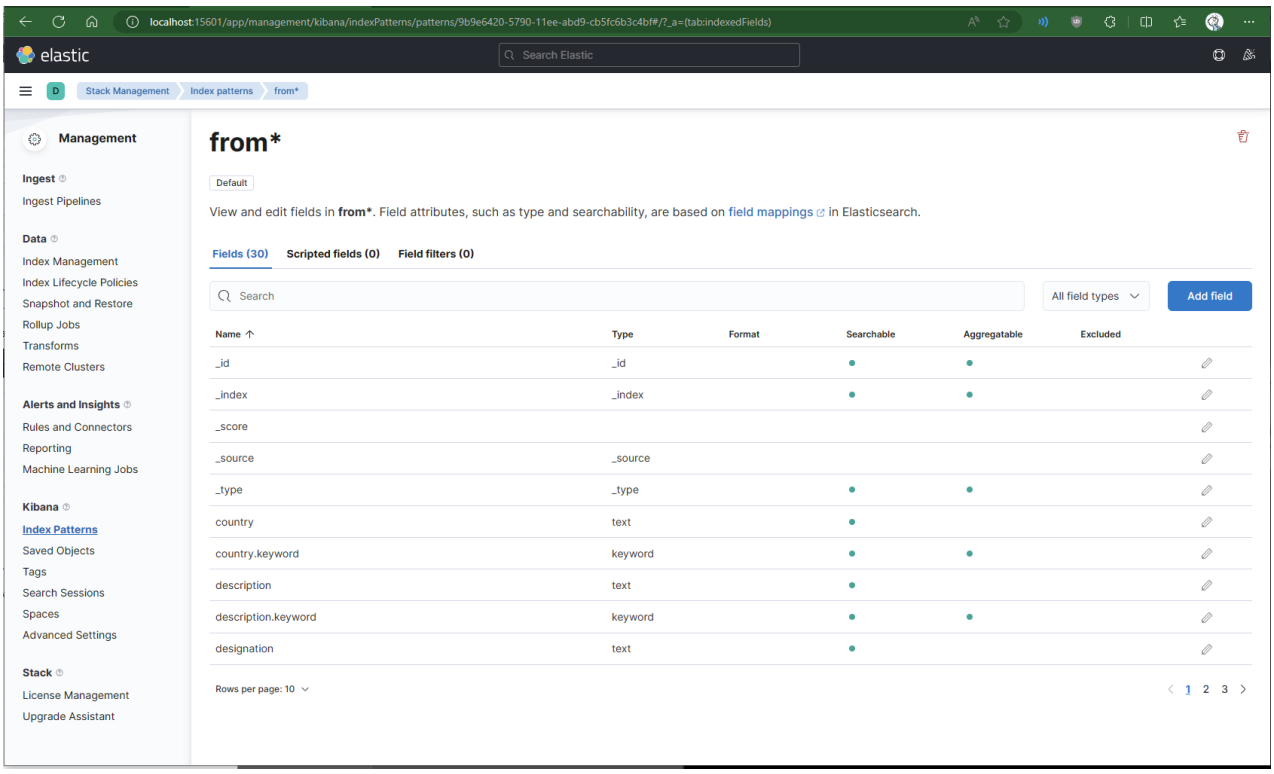

Рисунок 8 – Индекс паттерн для присланных данных

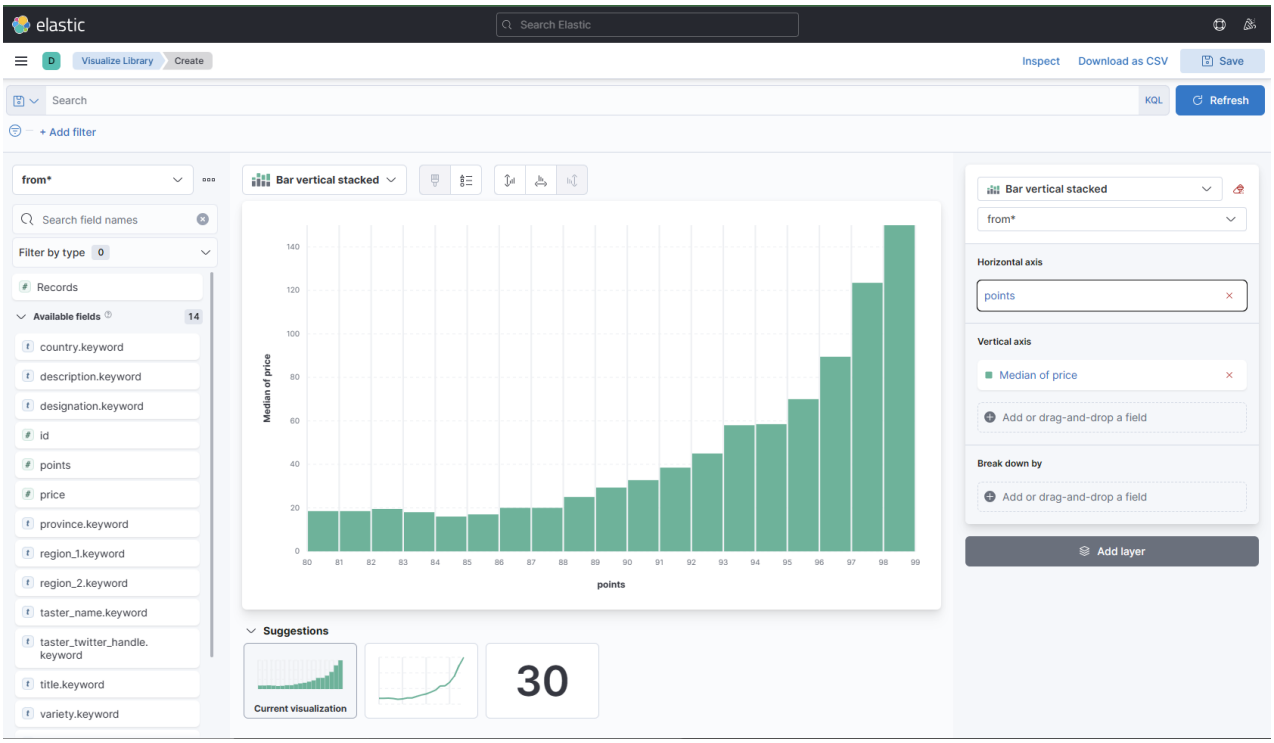

Рисунок 9 – Гистограмма зависимости цены от очков

Apache Airflow

Код компонента DAG:

```
1.  @dag(
2.      start_date=datetime(2021, 1, 1),
3.      schedule_interval=None,
4.      catchup=False,
5.      doc_md=__doc__,
```

```
)
def data_clean_pipeline():
    filename_path_save = '/opt/airflow/data/lab1_output/lab1_output.csv'
    load_data_folder = '/opt/airflow/data/lab1'

    @task
    def load_data(path_folder: str):
        csv_files = glob.glob(os.path.join(path_folder, '*.csv'))
        print(f'Found csv files: {len(csv_files)}')

        return pd.concat(
            (
                pd.read_csv(single_file)
                for single_file in csv_files
            ),
            ignore_index=True
        )

    @task
    def preprocessing(df: pd.DataFrame):
        df = df[df['designation'].notnull()]
        df = df[df['region_1'].notnull()]
        df.loc[df["price"].isnull(), "price"] = 0.0

        return df

    @task
    def save_data(df: pd.DataFrame, filename_path_save: str):
        df.to_csv(filename_path_save)

    @task
    def save_data_to_elastic(df: pd.DataFrame):
        from elasticsearch import Elasticsearch
        from elasticsearch.helpers import bulk
        es = Elasticsearch(['elasticsearch-kibana'], port=9200)

        data = df.to_json(orient='records')
        actions = [
            {
                '_index': 'my_index',
                '_type': 'my_type',
                '_id': i,
                '_source': doc
            }
            for i, doc in enumerate(data)
        ]
        # bulk(es, actions)
        es.bulk(actions)

    df = load_data(load_data_folder)
    pd_preprocessed = preprocessing(df)
    save_data(pd_preprocessed, filename_path_save)
    save_data_to_elastic(pd_preprocessed)
```