**Грибанов Данил, 6233**

**Ход работы**

**Apache Nifi**

Полная схема в системе NiFi представлена на рисунке 1.

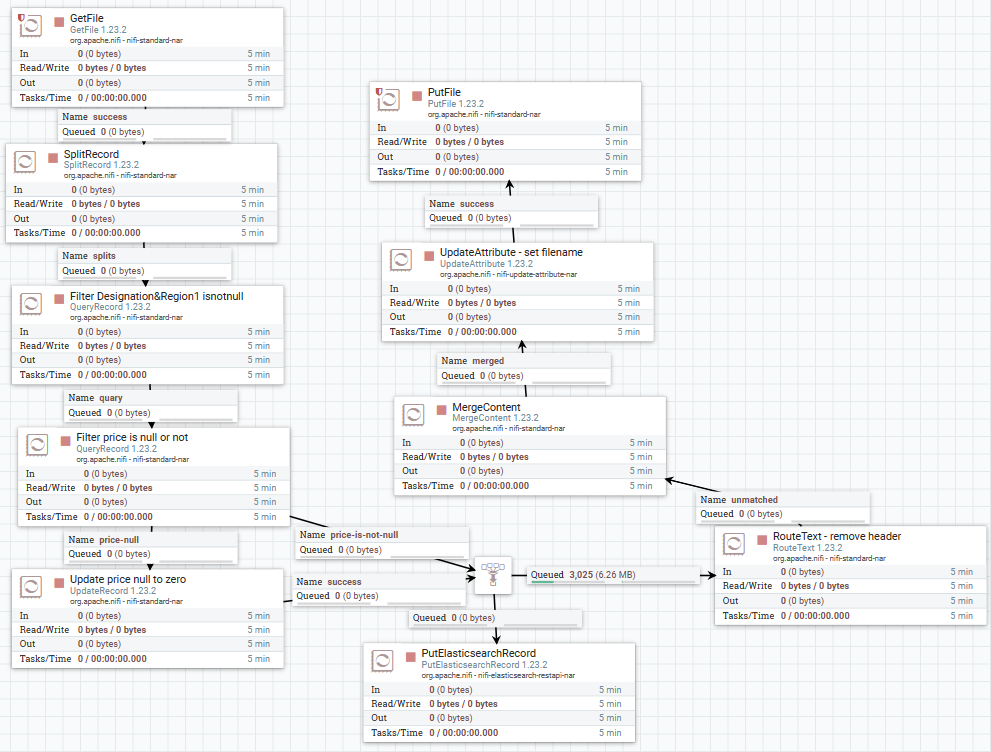


Рисунок 1 – Схема в системе NiFi

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.

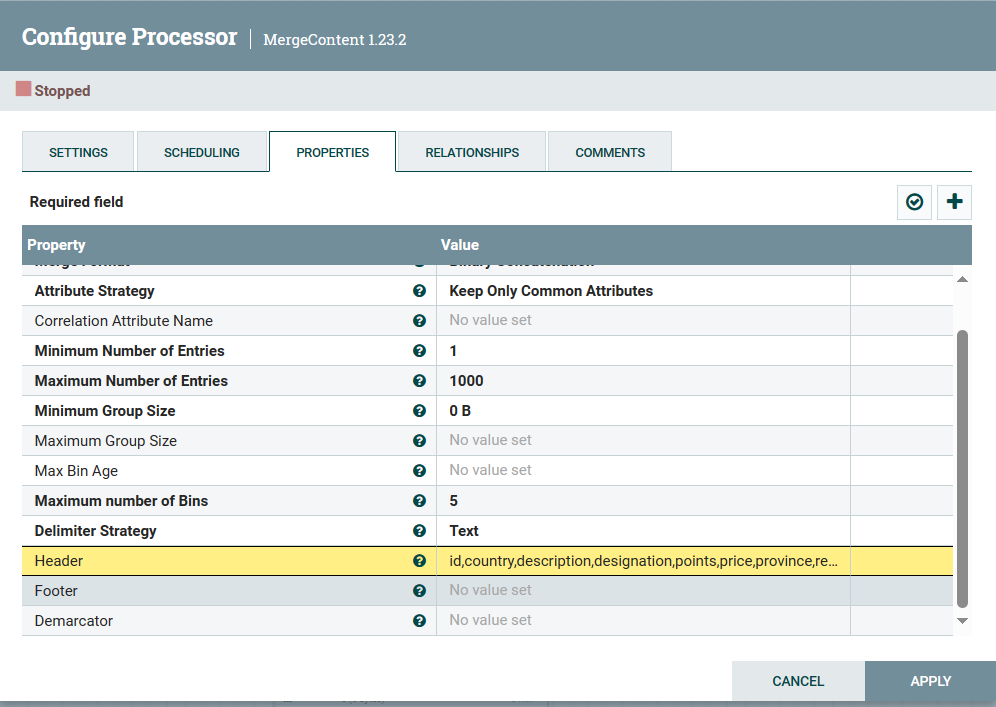


Рисунок 2 – Настройки MergeContent

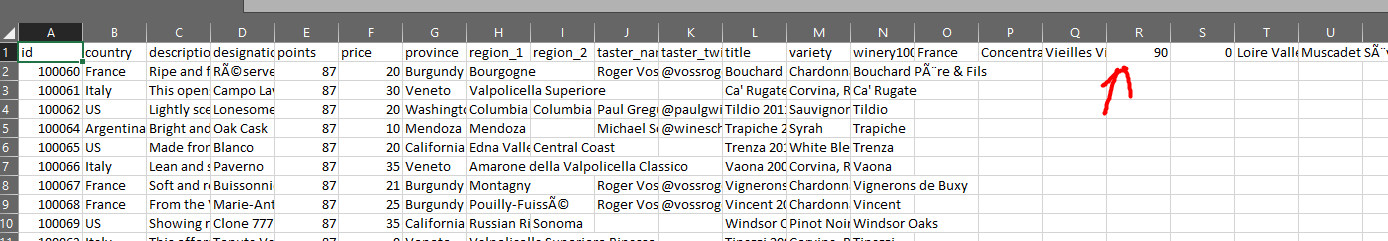


Рисунок 3 – Отсутствие новой строки при соединении шапки и строк файла

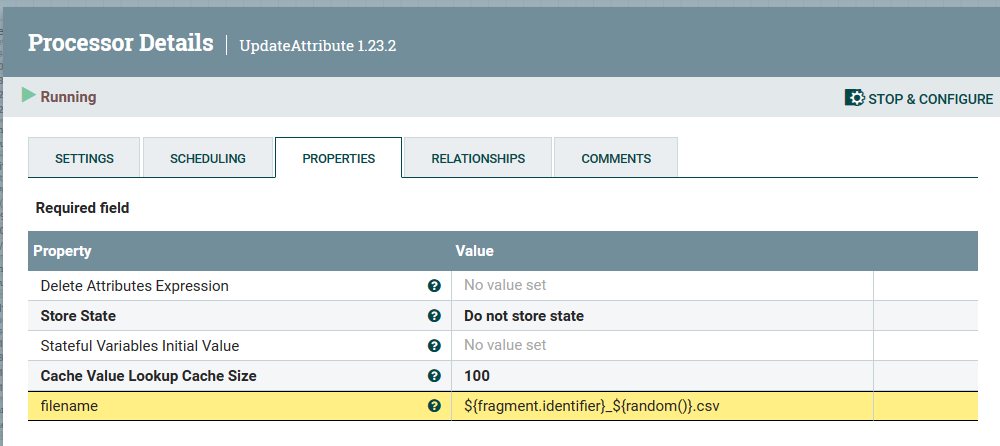


Рисунок 4 – Установка нового имени файла для сохранения

**Nibana elastic-search**

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

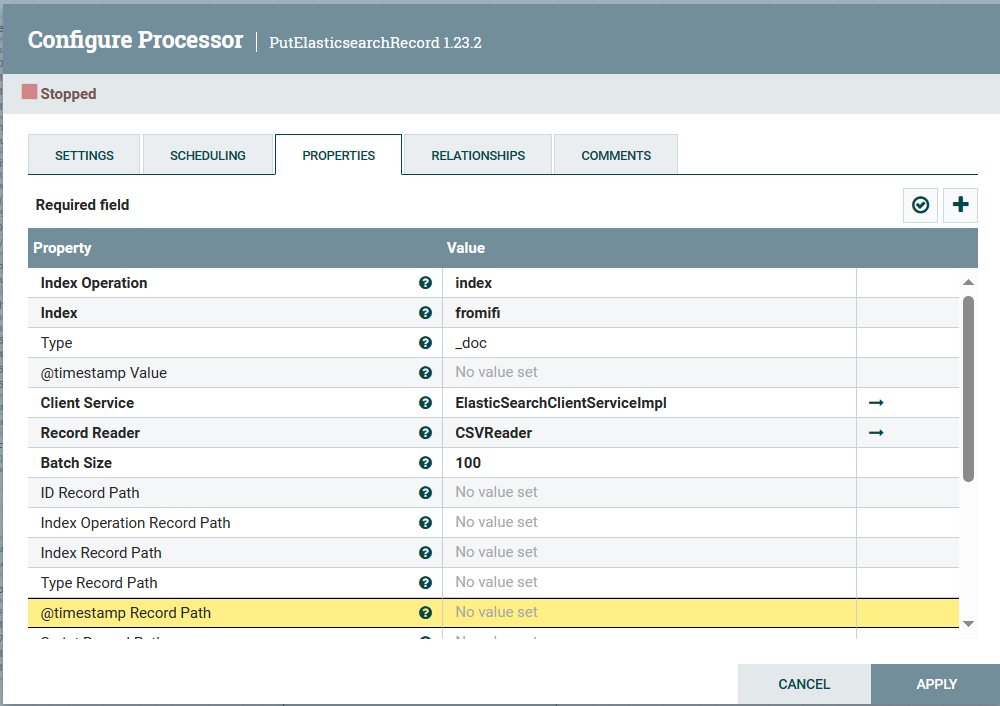


Рисунок 5 – Создание PutElasticsearchRecord

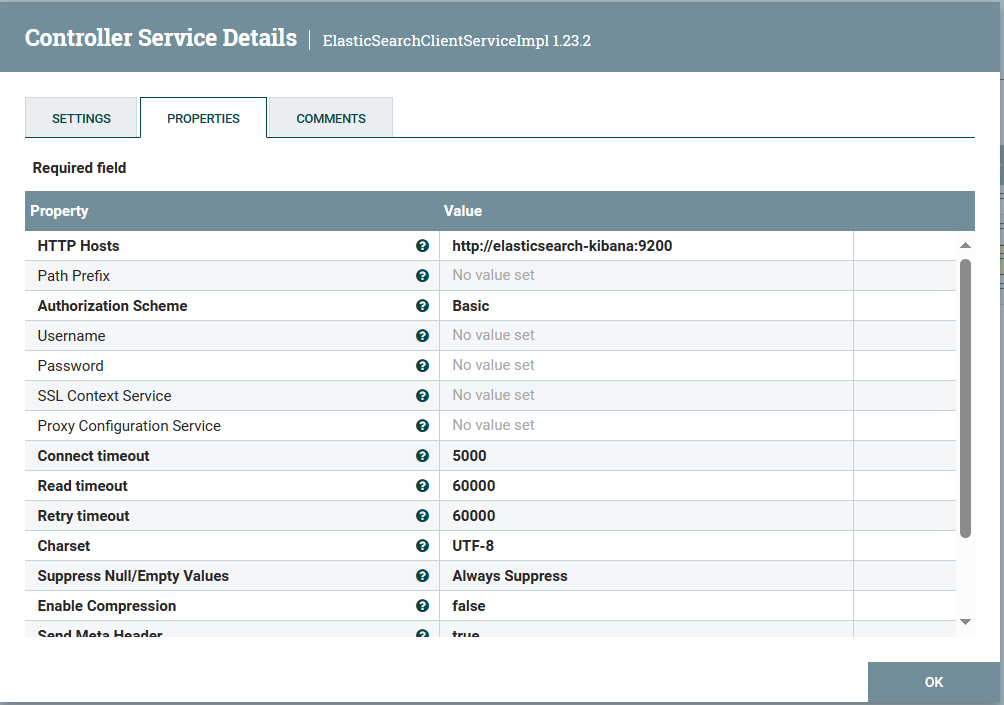


Рисунок 6 – Настройка сервиса для отправки

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

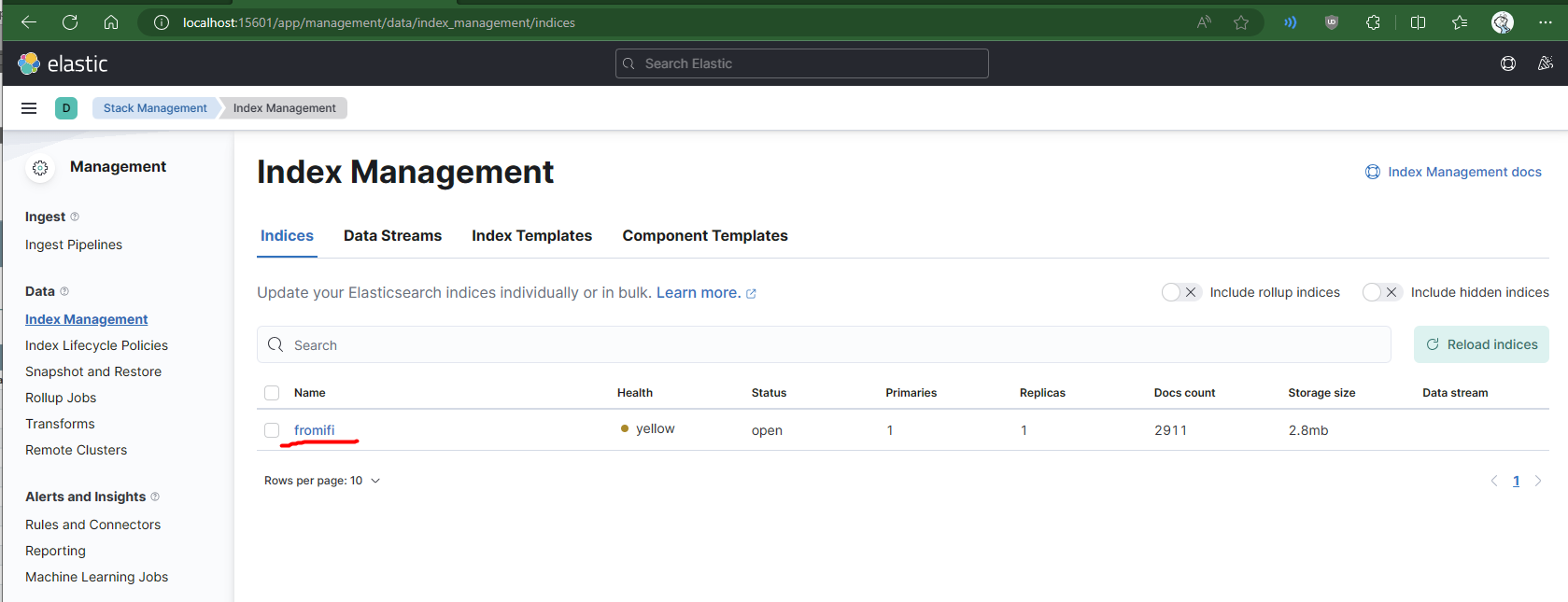


Рисунок 7 – Присланные данные из NiFi

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).

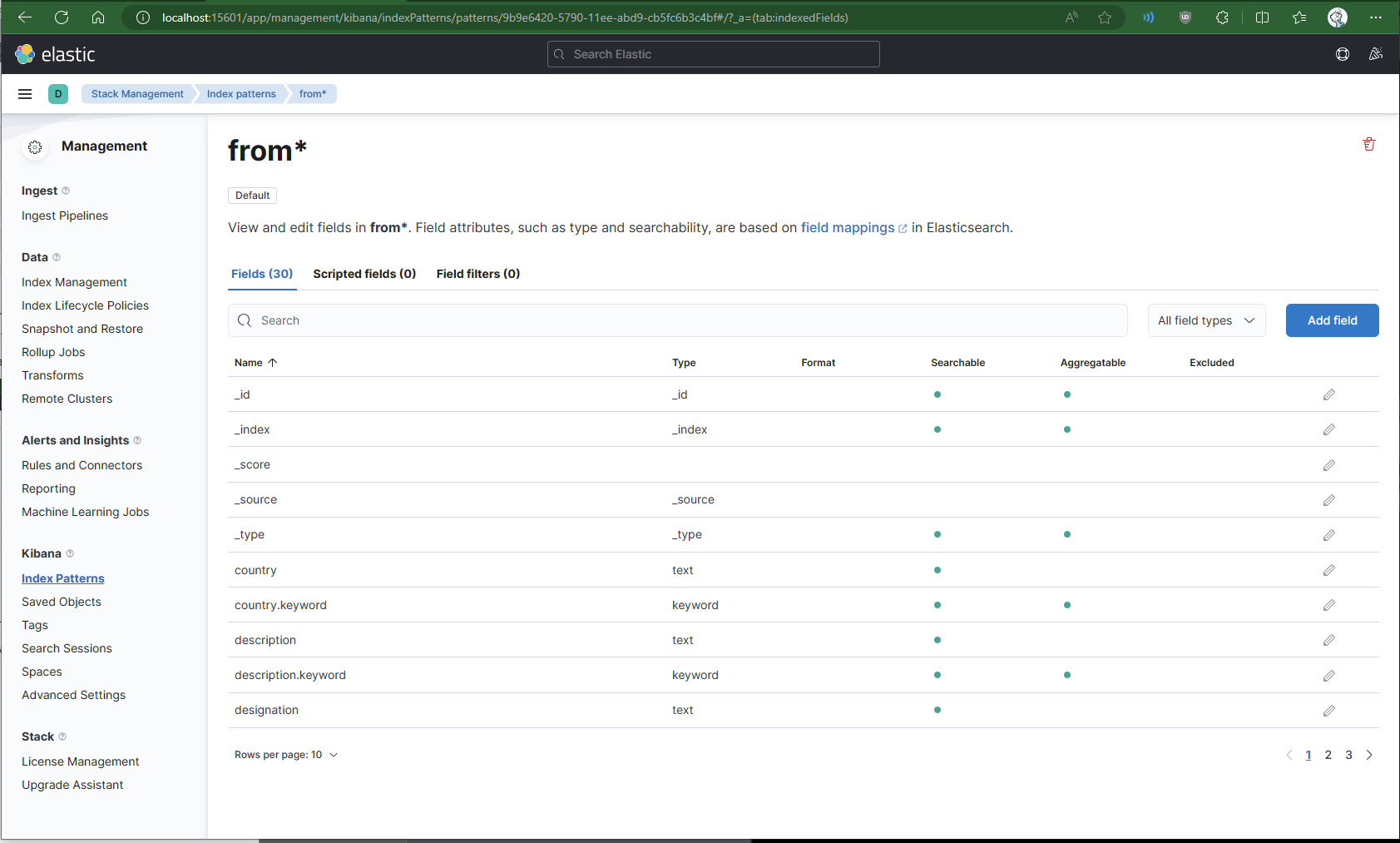


Рисунок 8 – Индекс паттерн для присланных данных

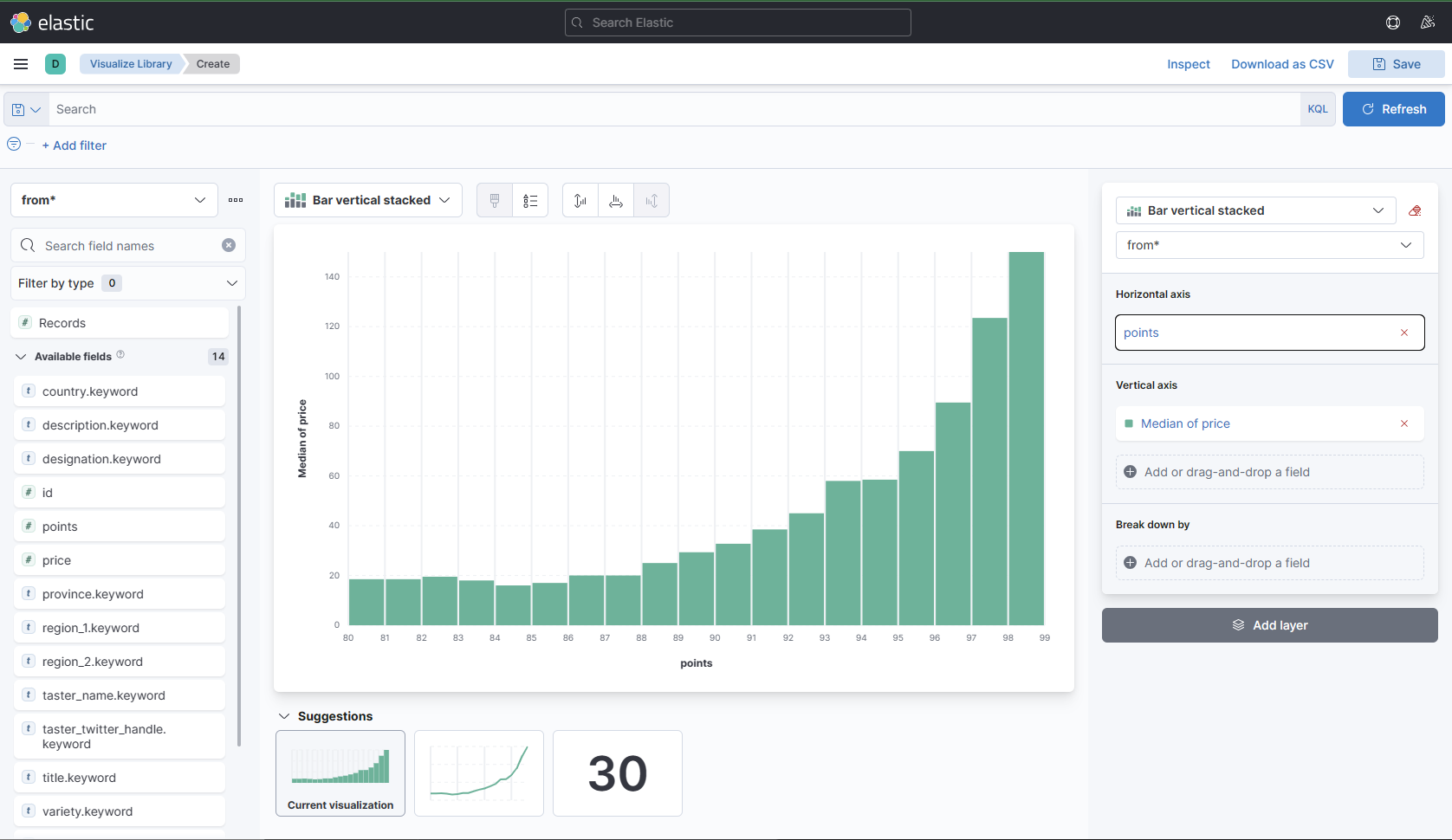


Рисунок 9 – Гистограмма зависимости цены от очков

**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\_\_,
6. )
7. **def** data\_clean\_pipeline():
8. filename\_path\_save **=** '/opt/airflow/data/lab1\_output/lab1\_output.csv'
9. load\_data\_folder **=** '/opt/airflow/data/lab1'
11. @task
12. **def** load\_data(path\_folder: str):
13. csv\_files **=** glob.glob(os.path.join(path\_folder, '\*.csv'))
14. print(f'Found csv files: {len(csv\_files)}')
16. **return** pd.concat(
17. (
18. pd.read\_csv(single\_file)
19. **for** single\_file **in** csv\_files
20. ),
21. ignore\_index**=**True
22. )
24. @task
25. **def** preprocessing(df: pd.DataFrame):
26. df **=** df[df['designation'].notnull()]
27. df **=** df[df['region\_1'].notnull()]
28. df.loc[df["price"].isnull(), "price"] **=** 0.0
30. **return** df
32. @task
33. **def** save\_data(df: pd.DataFrame, filename\_path\_save: str):
34. df.to\_csv(filename\_path\_save)
36. @task
37. **def** save\_data\_to\_elastic(df: pd.DataFrame):
38. **from** elasticsearch **import** Elasticsearch
39. **from** elasticsearch.helpers **import** bulk
40. es **=** Elasticsearch(['elasticsearch-kibana'], port**=**9200)
42. data **=** df.to\_json(orient**=**'records')
43. actions **=** [
44. {
45. '\_index': 'my\_index',
46. '\_type': 'my\_type',
47. '\_id': i,
48. '\_source': doc
49. }
50. **for** i, doc **in** enumerate(data)
51. ]
52. # bulk(es, actions)
53. es.bulk(actions)
55. df **=** load\_data(load\_data\_folder)
56. pd\_preprocessed **=** preprocessing(df)
57. save\_data(pd\_preprocessed, filename\_path\_save)
58. save\_data\_to\_elastic(pd\_preprocessed)