



RIWS

Máster Universitario en Ingeniería Informática - MUEI

2022/2023

Memoria RIWS

Autores: Jorge Berbel Carballal

Marcos Mosquera Miranda

Pablo Rodríguez Miñambres

A Coruña, Mayo de 2022.

Índice general

1	Introducción	1
2	Tecnologías utilizadas	3
2.1	Scrapy	3
2.2	Elasticsearch	4
2.3	Angular	4
3	Desarrollo del proyecto	5
3.1	Estudio del dominio y análisis de información	5
3.2	Crawling y procesamiento de datos	8
3.3	Interfaz Web de usuario y búsqueda de documentos	10
4	Entregables	15
4.1	Contenidos entregados	15
4.2	Ejecución	15

Índice de figuras

3.1	Información partidos Primera División	6
3.2	Estadísticas de jugadores de Primera División	7
3.3	Spider jugadores	9
3.4	Items de partidos y jugadores	10
3.5	Obtener resultados de los partidos por liga y jornada	11
3.6	Jornadas disponibles por liga	11
3.7	Obtener las ligas	12
3.8	Obtener detalles del partido de una jornada.	12
3.9	Web detalles partido concreto	12
3.10	Web listado de partidos de una jornada para un equipo	13
3.11	Obtener los campos de jugador	13
3.12	Obtener ligas de jugador	14
3.13	Web listado estadísticas jugadores de una liga	14

Capítulo 1

Introducción

EN el mundo del deporte cada vez esta más de moda el uso de "Big Data" y se le da gran importancia a todo lo que gira entorno a las estadísticas, resultados de los partidos, porcentajes de aciertos, probabilidades de anotación, etc. Por ello, hemos decidido enfocar nuestra práctica de recuperación de información en el dominio del fútbol y sus estadísticas.

Nos centraremos en obtener los datos de diferentes ligas y jugadores de la actual temporada 2022/2023 para poder realizar diferente filtrados sobre ellos como por ejemplo, resultados de los partidos, equipos con más goles anotados, jugadores con más goles, etc.

Para lograrlo se han utilizado técnicas de Information Retrieval para la Web. Se obtendrán los datos de un sitio web con información sobre el dominio utilizando técnicas de scraping web. Estos datos los parsearemos para enviarlos a un servidor de Elasticsearch y posteriormente podrán ser consultados por el usuario mediante diferentes métodos de búsqueda a través de una aplicación desarrollada en Angular.

Tecnologías utilizadas

Para la realización de la práctica se ha escogido la utilización de Scrapy para el web crawling y el scraping, Elasticsearch para el motor de búsqueda y Angular para el desarrollo de la aplicación Web. En las secciones de este capítulo se describirán en detalle las tecnologías usadas.

2.1 Scrapy

Scrapy es un framework de scraping y crawling, escrito en Python, para extracción de datos de sitios web. Con el obtendremos los diferentes datos sobre nuestro dominio.

Scrapy, a través de una "Spider" realiza el crawling de las webs necesarias y a continuación su parseo de datos. Este framework nos permite crear una estructura de proyecto, con todas las clases necesarias, de una manera sencilla simplemente con la ejecución del comando `scrapystartproject project_name`. También cabe destacar el comando `scrapy shell` que nos permite conocer de una manera eficaz si una página web es scrapeable.

2.2 Elasticsearch

Como motor de búsqueda se ha escogido Elasticsearch frente a Apache Solr, puesto que usa un lenguaje de consultas basado en JSON y sobretodo por que presenta una amplia y accesible documentación. Además, aunque no se ha utilizado en esta práctica, Elasticsearch cuenta con la herramienta Kibana como panel visualizador de datos, que se podría implementar para mostrar de una manera gráfica los datos indexados.

Cabe destacar, su fácil integración con docker que nos ha permitido crear un contenedor Docker donde hemos podido levantar fácilmente el servidor de Elasticsearch.

2.3 Angular

Angular es un framework para aplicaciones web desarrollado en TypeScript. Se utilizará para la creación de la página web que permita al usuario acceder a las búsquedas de datos. Se ha escogido por la experiencia que se tiene con este software y por la existencia de una librería para su integración con Elasticsearch.

Desarrollo del proyecto

EL desarrollo del proyecto consta de diferentes fases, que se explicarán a lo largo de este capítulo.

3.1 Estudio del dominio y análisis de información

Mediante el uso de la herramienta *scrapy shell* encontramos varias webs sobre estadísticas de fútbol que a priori eran scrapeables pero nos daban problemas para obtener los datos. Finalmente, nos decantamos por la web www.resultados-futbol.com en la que descubrimos multitud de datos sobre los partidos, jugadores y demás. Una vez escogida la web que vamos a scrapear estudiamos que campos concretos nos interesa obtener, por lo que primeramente establecemos las urls queremos recuperar. Partiremos de los siguientes links:

- <https://www.resultados-futbol.com/primera/grupo1/jornada1>:

URL que devuelve la información de los resultados de la jornada número uno de Primera División española. En ella podemos observar los nombres de los equipos, el resultado del partido, la clasificación actual del campeonato e información sobre estadísticas de jugadores. Utilizaremos el link de *Jornada Siguiente*

para movernos entre jornadas de la liga y el enlace presente en el resultado del encuentro para acceder a información más detallada de este.

- https://www.resultados-futbol.com/primera_division_rfef/grupo1/jornada1: Esta URL devuelve los mismos campos de información que la anterior pero para la liga de primera división RFEF.

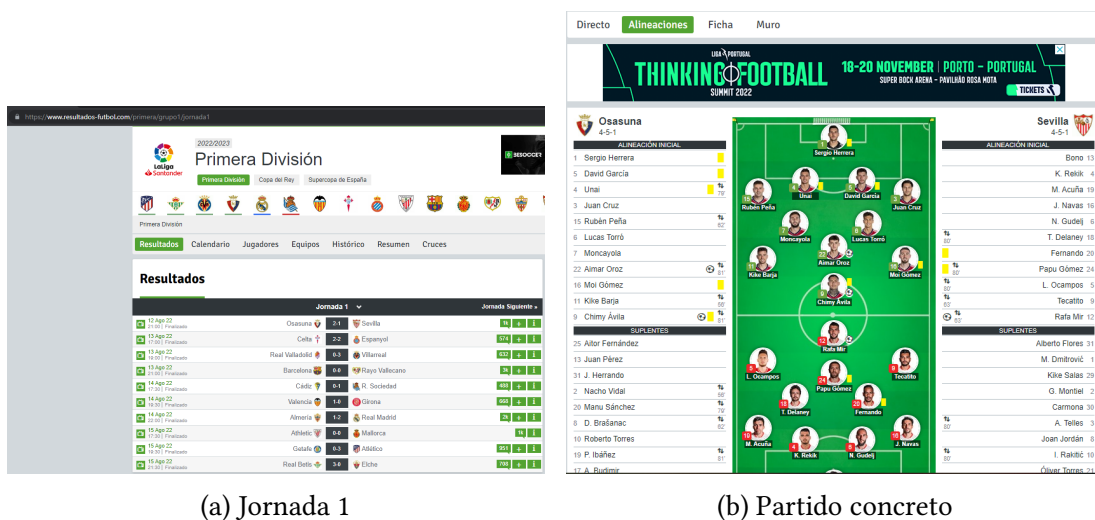


Figura 3.1: Información partidos Primera División

https://www.resultados-futbol.com/primera/grupo1/jornada1

Descenso: Elche, Getafe, Real Valladolid

JUGADORES DESTACADOS

GOLES		TARJETAS AMARILLAS		TARJETAS ROJAS		ASISTENCIAS	
RK	Nombre	Goles	Posición	Equipo			
1	R. Lewandowski	12	DEL	Barcelona			
2	Borja Iglesias	8	DEL	Real Betis			
3	V. Muriqi	8	DEL	Mallorca			
4	Iago Aspas	7	DEL	Celta			
5	Joselu	7	DEL	Espanyol			
6	Chimy Ávila	6	DEL	Osasuna			
7	Brais Méndez	6	MED	R. Sociedad			
8	F. Valverde	6	MED	Real Madrid			
9	Vinicius Jr.	6	DEL	Real Madrid			
10	K. Benzema	5	DEL	Real Madrid			
11	A. Griezmann	5	DEL	Atlético			
12	Á. Morata	5	DEL	Atlético			
13	Sergio León	5	DEL	Real Valladolid			
14	I. Williams	5	DEL	Athletic			
15	Gorka Guruzeta	5	DEL	Athletic			
16	E. Cavani	4	DEL	Valencia			
17	C. Stuani	4	DEL	Girona			
18	E. Únal	4	DEL	Getafe			
19	A. Serloth	4	DEL	R. Sociedad			
20	O. Dembélé	4	DEL	Barcelona			

Figura 3.2: Estadísticas de jugadores de Primera División

La web es homogénea para todos los campeonatos ligeros disponibles por lo que se podría añadir más ligas y scrapear esta misma información para ellas. Como último punto de esta fase, debido a que queremos obtener por un lado datos de los partidos y por el otro datos de los jugadores, tendremos que realizar dos extracciones de datos distintas que implicarán la creación de dos índices en Elasticsearch. Uno de ellos para obtener los campos deseados sobre los partidos y otro sobre los jugadores. De este modo, se define la siguiente estructura que tendrán los documentos que se almacenarán:

- **Partido:**

- **Home team:** Nombre del equipo local.
- **Home score:** Goles del equipo local.
- **Home shield:** Escudo del equipo local.

- **Away team:** Nombre del equipo visitante.
 - **Away score:** Goles del equipo visitante.
 - **Away shield** Escudo del equipo visitante.
 - **Match day:** Fecha del partido.
 - **Match Stadium:** Estadio del partido.
 - **League:** Liga de los equipos.
 - **Journey:** Número de la jornada del campeonato.
 - **Referee:** Árbitro principal del partido.
- **Jugador:**
 - **Ranking:** Número en el ranking de máximos goleadores de la liga.
 - **Name:** Nombre del jugador.
 - **Goals:** Goles del jugador.
 - **Position:** Posición del jugador.
 - **Team name:** Nombre del equipo al que pertenece el jugador.
 - **League:** Liga a la que pertenece el jugador.

3.2 Crawling y procesamiento de datos

Para la implementación del Crawling y el Scrapping hemos usado el software Scrapy como se ha comentado anteriormente. Como se indica en la documentación de Scrapy, el procedimiento es implementar lo que se denomina un spider, en ella se realiza el proceso de crawling y el parseo de datos. El proyecto scrapy se encuentra estructurado de la siguiente manera:

- **Spiders:** Los Spiders son las clases que definen como será la extracción de datos, el proceso de crawling y la estructuración de las páginas en items. En la figura 3.3 se muestra el spider utilizado para la obtención de los datos de jugadores.

```
class PlayerSpider(scrapy.Spider):
    es_create_index_if_not_exists(es, 'matchplayer', mapping_player)
    name = 'players'
    start_urls = ['https://www.resultados-futbol.com/primera/grupo1/jornada1', 'https://www.resultados-futbol.com/primera_division_rfef/grupo1/jornada1']

    def parse(self, response):
        global cont
        league = response.css('div.clearfix div.titular-data h1::text').get()
        for player in response.css("div.gridPlayers tr.fila"):
            football_player = FootballPlayer()
            #other option
            doc = {
                'id': str(cont),
                'ranking' : player.css('td::text').get(),
                'name' : player.css('td a::text').get(),
                'goals' : player.css('td strong::text').get(),
                'position' : player.css('td.role::text').get(),
                'teamName' : player.css('td.esp a::text').get(),
                'league' : league
            }
            football_player['id'] = cont
            football_player['ranking'] = player.css('td::text').get()
            football_player['name'] = limpiar_acentos(player.css('td a::text').get())
            football_player['goals'] = player.css('td strong::text').get()
            football_player['position'] = player.css('td.role::text').get()
            football_player['teamName'] = limpiar_acentos(player.css('td.esp a::text').get())
            football_player['league'] = limpiar_acentos(league)
            cont=cont+1
```

Figura 3.3: Spider jugadores

- **Items:** Se define la estructura que seguirán los documentos. En ella se definen un item para partidos y otro para jugadores.

```
class FootballScoreItem(scrapy.Item):
    id = scrapy.Field()
    homeTeam = scrapy.Field()
    homeScore = scrapy.Field()
    homeShield = scrapy.Field()
    awayTeam = scrapy.Field()
    awayScore = scrapy.Field()
    awayShield = scrapy.Field()
    matchDay = scrapy.Field()
    matchStadium = scrapy.Field()
    matchResult = scrapy.Field()
    league = scrapy.Field()
    journey = scrapy.Field()
    referee = scrapy.Field()
```

(a) Partidos

```
class FootballPlayer(scrapy.Item):
    id = scrapy.Field()
    ranking = scrapy.Field()
    name = scrapy.Field()
    goals = scrapy.Field()
    position = scrapy.Field()
    teamName = scrapy.Field()
    league = scrapy.Field()
```

(b) Jugadores

Figura 3.4: Items de partidos y jugadores

- **Middleware:** Se crea por defecto al crear el proyecto scrapy, en nuestro caso no la hemos modificado.
- **Pipelines:** Se puede utilizar para crear procesos que se aplican a los items que se van obteniendo durante el crawling de la web. En nuestro caso los hemos definido sobre el spider directamente.
- **Settings:** En esta clase se definen diferentes parámetros de configuración como puede ser deshabilitar el robots.txt: *ROBOTSTXT_OBEY = False*

3.3 Interfaz Web de usuario y búsqueda de documentos

Realizamos una interfaz Web en Angular para realizar las siguientes búsquedas:

- Partidos:
 - Obtener resultados de los partidos por liga y jornada.

```
private _getMatchBody(league: String, journey: String): any {  
  const requestBody =  
  {  
    "query": {  
      "bool": {  
        "must": [  
          {  
            "match": {  
              "league": league  
            }  
          },  
          {  
            "match": {  
              "journey": journey  
            }  
          }  
        ]  
      }  
    }  
  }  
  return requestBody;  
}
```

Figura 3.5: Obtener resultados de los partidos por liga y jornada

- Jornadas disponibles por liga.

```
private _getJourneyRequestBody(league: String): any {  
  const requestBody =  
  {  
    "_source": "journey",  
    "query": {  
      "match": {  
        "league": league  
      }  
    }  
  }  
  return requestBody;  
}
```

Figura 3.6: Jornadas disponibles por liga

- Obtener las ligas.

```
private _getMatchLeagueBody(): any {
  const requestBody =
  {
    "size": "0",
    "aggs": {
      "uniq_gender": {
        "terms": {
          "field": "league"
        }
      }
    }
  }
  return requestBody;
}
```

Figura 3.7: Obtener las ligas

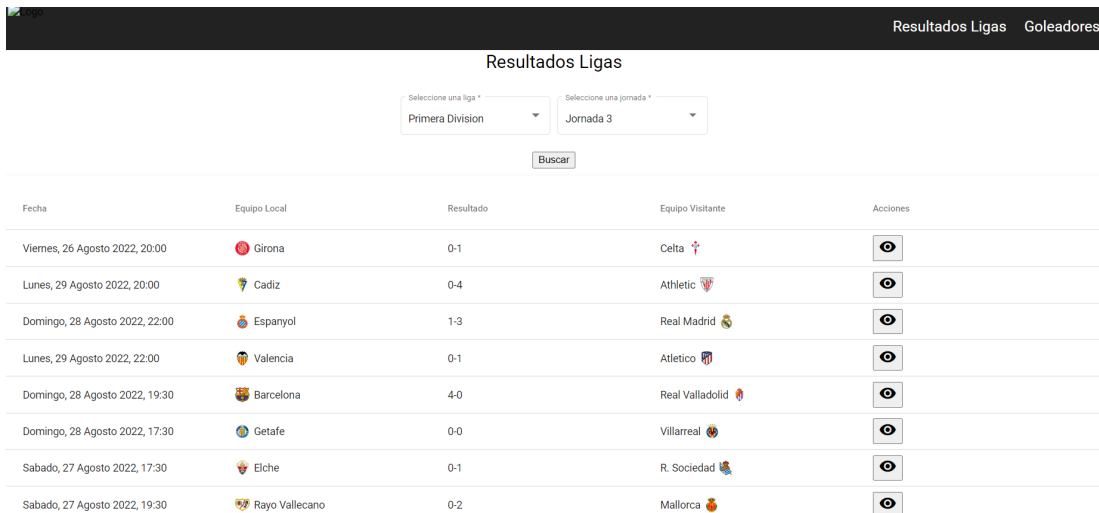
- Obtener detalles del partido de una jornada.

```
getMatchScoreById(id: number): Observable<Object> {
  const headers = new HttpHeaders({
    'Content-type': 'application/json'
  });
  return this._httpClient.get<Object>(environment.host + environment.matchIndexName + "/_doc/" + id);
}
```

Figura 3.8: Obtener detalles del partido de una jornada.

Detalles del partido	
Estadio	Árbitro
Estadio Ramon Sanchez-Pizjuan	De Burgos Bengoetxea, Ricardo

Figura 3.9: Web detalles partido concreto



Fecha	Equipo Local	Resultado	Equipo Visitante	Acciones
Viernes, 26 Agosto 2022, 20:00	Girona	0-1	Celta	
Lunes, 29 Agosto 2022, 20:00	Cadiz	0-4	Athletic	
Domingo, 28 Agosto 2022, 22:00	Espanyol	1-3	Real Madrid	
Lunes, 29 Agosto 2022, 22:00	Valencia	0-1	Atletico	
Domingo, 28 Agosto 2022, 19:30	Barcelona	4-0	Real Valladolid	
Domingo, 28 Agosto 2022, 17:30	Getafe	0-0	Villarreal	
Sabado, 27 Agosto 2022, 17:30	Elche	0-1	R. Sociedad	
Sabado, 27 Agosto 2022, 19:30	Rayo Vallecano	0-2	Mallorca	

Figura 3.10: Web listado de partidos de una jornada para un equipo

- Jugadores:

- Obtener los campos de jugador.

```
private _getPlayerBody(league: String): any {  
    const requestBody =  
        {  
            "query": {  
                "match": {  
                    "league": league  
                }  
            }  
        }  
    }  
  
    return requestBody;  
}
```

Figura 3.11: Obtener los campos de jugador

- Obtener la liga de un jugador.

```
private _getPlayerLeagueBody(): any {
  const compoSiteQuery= {
    "size": 250,
    "sources": [{
      "terms": {
        "field": "league"
      }
    }]
  }

  const requestBody =
  {
    "aggs": {
      "values": {
        "composite": compoSiteQuery
      }
    }
  }

  return requestBody;
}
```

Figura 3.12: Obtener ligas de jugador

Resultados Ligas Goleadores				
Goleadores				
Seleccione una liga *				
Primera Federacion				
Buscar				
Posición	Nombre	Nº Goles	Posición	Equipo
1	A. Romero	7	DEL	Algeciras
2	Willy	5	DEL	Cordoba
3	Manu Justo	5	DEL	Racing Ferrol
4	S. Arribas	5	MED	RM Castilla
5	D. Rodriguez	4	DEL	Rayo Majadahonda
6	J. Casado	4	DEF	Rayo Majadahonda
7	Nestor Albiach	4	MED	Rayo Majadahonda
8	Kike Marquez	4	DEL	Cordoba

Figura 3.13: Web listado estadísticas jugadores de una liga

Entregables

4.1 Contenidos entregados

EL código fuente esta almacenado en el siguiente directorio de github: <https://github.com/marmosqueramir/practicaRIWS> en el distinguimos:

- **Docker/**: Directorio que contiene el contenedor Docker que aloja el servidor de Elasticsearch.
- **football-scores-web/**: Contiene el código de la aplicación web.
- **footballCrawler/**: Contiene el código del proyecto Scrapy que obtiene los datos de los partidos y de los jugadores.

4.2 Ejecución

Para la ejecución de esta práctica es necesario tener instalado:

- **Docker**
- **Docker Compose**

- **Scrapy python**

Pasos para la ejecución de la práctica:

1. Arrancar el servicio de Elasticsearch con el comando *docker-compose up -d*.
2. Situarnos en el directorio /footballCrawler.
3. Ejecutar el comando *pip install -r requirements.txt*.
4. Ejecutar el crawler de partidos con el comando *scrapy crawl matchScores*.
5. Ejecutar el crawler de jugadores con el comando *scrapy crawl players*.
6. Situarnos en football-score-web.
7. Ejecutar el comando *npm install*.
8. Ejecutar el comando *ng build*.
9. Ejecutar el comando *ng serve -o*.