Generación de un mapa interactivo a partir de la extracción e indexación de información del videojuego Temtem

Recuperación de la Información y Web Semántica
Máster Universitario en Ingeniería Informática
2021-2022

Barrientos Lema, Iván
ivan.barrientos.lema@udc.es

Ramil López, Diego
diego.ramil.lopez@udc.es

10 de diciembre de 2021

Índice general

Índice de figuras

Capítulo 1

Desarrollo del proyecto

Para este proyecto, se eligió recuperar la información del videojuego Temtem, un videojuego multijugador basado en el combate y la colección de múltiples criaturas repartidas en un mundo virtual. Concretamente, se escogió la wiki creada y mantenida por su comunidad de jugadores. En ella, se recoge todo el contenido que ofrece el juego. Aunque de toda esta información, nosotros únicamente nos hemos centrado en extraer los aspectos más importantes relativos a las criaturas y su ubicación dentro de este metaverso.

Con el fin de extraer toda esta información se ha procedido de la siguiente manera:

- En primer lugar, se parte de la lista de criaturas que contiene el juego.
- Para cada criatura de esta lista que se pueda capturar, se extrae la siguiente información:
  - número: identificador único de la criatura.
  - nombre: nombre de la criatura.
  - tipos: elementos que tiene asociada la criatura. Cada criatura puede tener entre uno y dos tipos diferentes. Para extraer esta información se accede a una nueva URL donde se extrae la siguiente información:
    - nombre: nombre del elemento. Ej. agua, fuego, etc.
    - icono: imagen asociada al elemento.
  - retrato: imagen asociada a la criatura.
  - proporción de género: porcentaje de que la criatura tenga género masculino.
  - tasa de captura: variable que afecta a la probabilidad de que la criatura se capture.

- TV: entrenamiento ganado al derrotar la criatura por cada una de las características de combate. Cada criatura puede otorgar esfuerzo en una o dos características diferentes:
  - HP: entrenamiento ganado en vida.
  - STA: entrenamiento ganado en estamina.
  - SPD: entrenamiento ganado en velocidad.
  - ATK: entrenamiento ganado en ataque.
  - DEF: entrenamiento ganado en defensa.
  - SPATK: entrenamiento ganado en ataque especial.
  - SPDEF: entrenamiento ganado en defensa especial.
- localización: ubicación de la criatura en el universo del videojuego. Para extraer esta información se accede a una nueva URL donde se extrae la siguiente información:
  - isla: nombre de la isla donde aparece.
  - ruta: nombre de la ruta dentro de la isla donde aparece.
  - área: nombre del área dentro de la ruta donde aparece.
  - frecuencia: porcentaje de apariciones en dicha área.
  - nivel mínimo: nivel mínimo de aparición en dicha área.
  - nivel máximo: nivel máximo de aparición en dicha área.

En todo este proceso de recuperación de información, no se empleo el CrawlSpider de Scrapy debido a la estructura que presenta la wiki. En ella, todos los recursos cuelgan del elemento raíz. Independientemente de si el recurso es una criatura, un tipo o una localización. Por lo tanto, no es posible definir una regla que identifique a cada recurso por separado. Lo que obliga a tener que seguir el proceso descrito anteriormente. En la Figura 1.1 se muestra un esquema de cómo se realiza este proceso.

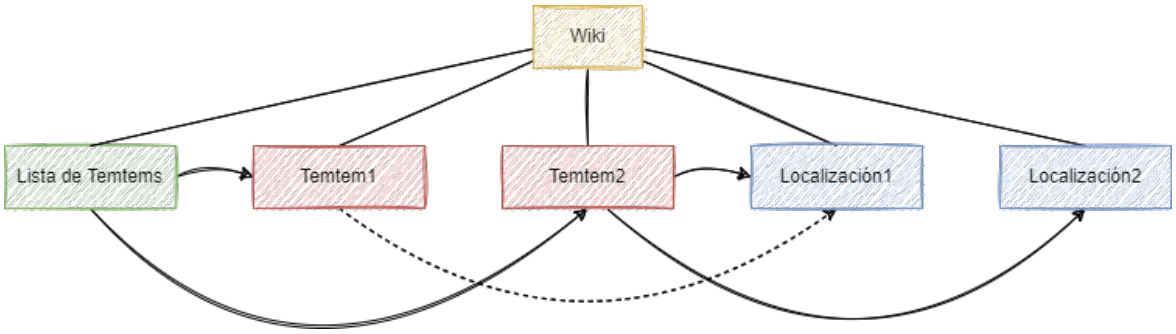

Figura 1.1: Esquema de recuperación de la información a partir de la wiki.

Capítulo 2

Motivación

La motivación detrás de este proyecto es hacer que los jugadores de Temtem tengan información más intuitiva y visual sobre las criaturas del videojuego. Para ello, estas criaturas se representan en un mapa interactivo junto a diversos filtros que permiten limitar los resultados obtenidos, haciendo así que esta tarea sea menos tediosa para los jugadores.

Además, estas características no solo hacen más simple la vida de los jugadores asiduos, quienes normalmente simplemente consultarán información básica del juego. Si no que también resultará de utilidad para aquellos que buscan maximizar ciertos aspectos del juego, como el beneficio obtenido por capturar una criatura.

Otro punto importante a tener en cuenta es que este videojuego es una propuesta de un estudio español inspirado en la saga Pokémon. Nació a partir de un *crowdfunding*[1] que llegó a batir el récord de mayor recaudación histórica en Kickstarter de un proyecto español. Llegando en determinados momentos tras su lanzamiento en acceso anticipado a contar con más de 30.000 jugadores simultáneos. Si además tenemos en cuenta el hecho de que está en desarrollo para múltiples plataformas y que cuenta con *cossplay*[2], el potencial que muestra es muy elevado en contraposición con la información que hay sobre él.

[1]Financiación colectiva, normalmente online, que a través de donaciones económicas o de otro tipo consiguen financiar un determinado proyecto.

[2]Servicio para los jugadores donde, comprando el mismo juego para diferentes plataformas, comparten servidores en las partidas multijugador.

Capítulo 3

Funcionalidades implementadas

Como idea principal para representar los datos recopilados, se optó por exponer la posición de las criaturas en un mapa. Estas se mostrarán como resultado de aplicar alguno de los criterios de búsqueda que se muestran en el lado izquierdo de la aplicación. Estos filtros pueden habilitarse o deshabilitarse a través de un *checkbox*. En la Figura 3.1 se puede observar la vista general de la aplicación y en la Figura 3.2 la vista en detalle de una criatura.

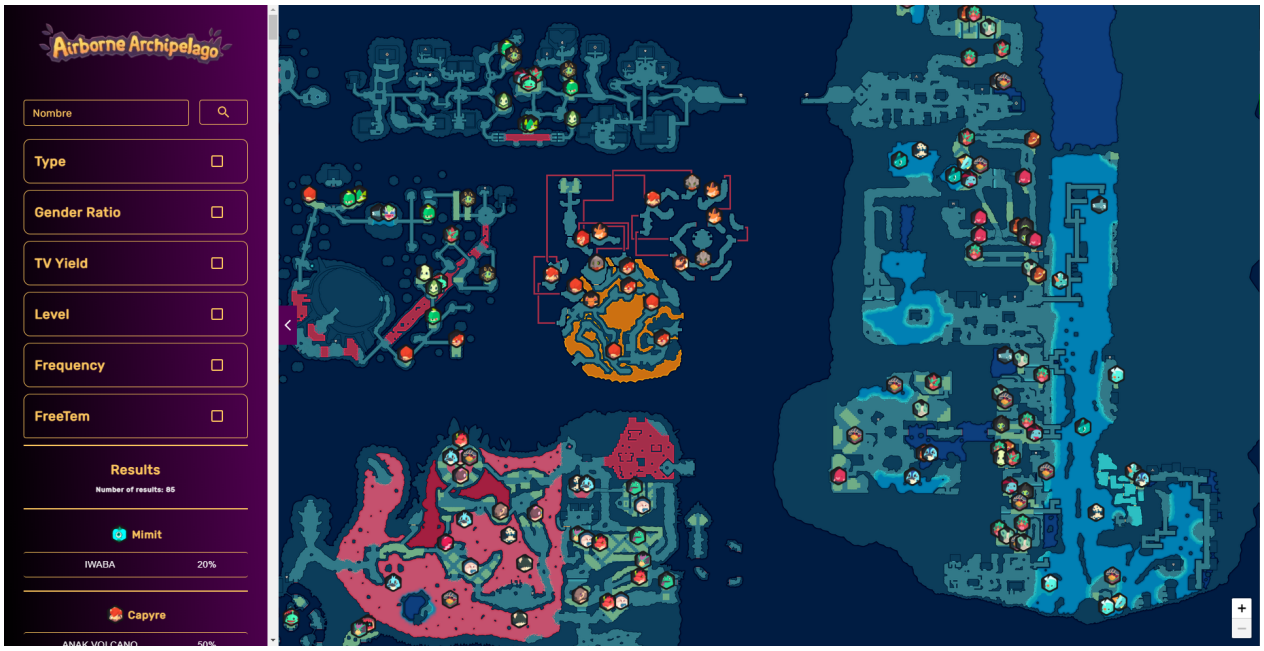

Figura 3.1: Pantalla inicial de la aplicación.

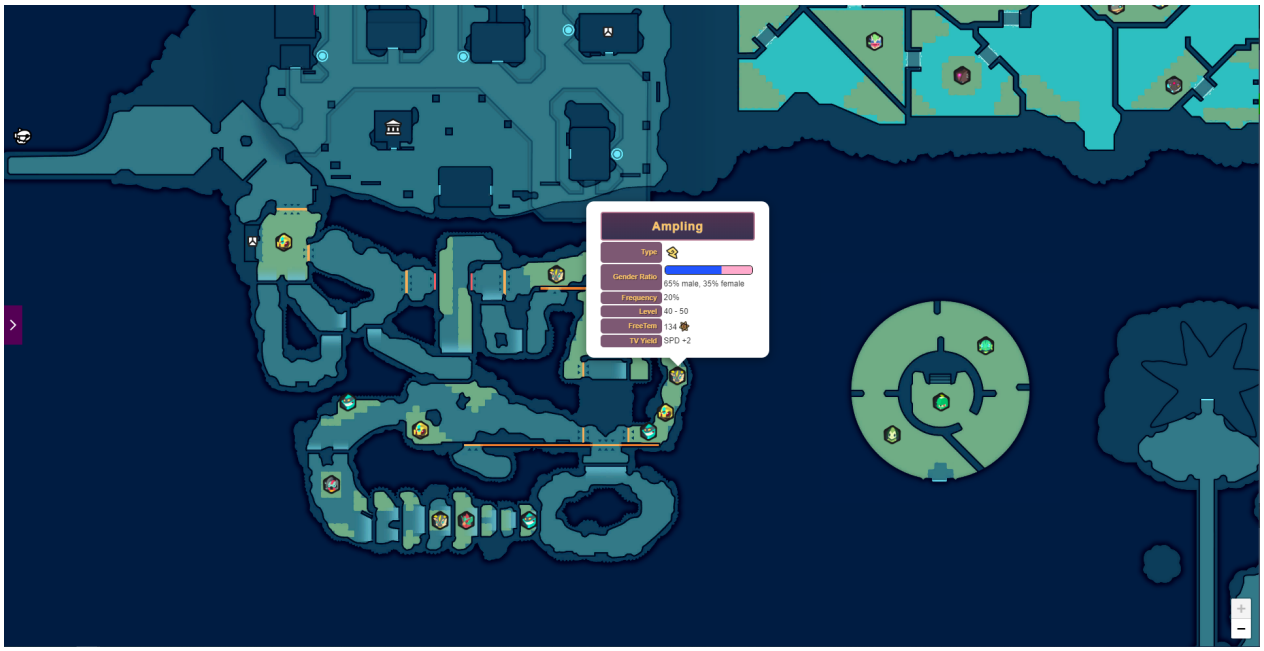

Figura 3.2: Pantalla de detalle de una criatura.

Los tipos de filtros que admite la aplicación son los siguientes:

- Filtro por nombre: este campo acepta una cierta variación en el valor que se le pase con respecto al nombre real de la criatura, permitiendo así ciertos errores de sintaxis por parte del usuario sin afectar a los resultados (Fig. 3.3).
- Filtro por tipo: el filtro limita a dos el número de tipos que se pueden seleccionar a la vez, ya que es el número máximo que una criatura puede llegar a tener (Fig. 3.3).
- Filtro por género: este filtro es un *slider* que permite seleccionar la proporción de género de una criatura. Para que el filtro solo permita valores relevantes, el rango de opciones se ha limitado a las utilizadas por el videojuego. Siendo este rango el que se muestra a continuación: [Macho-Hembra, 0-100, 5-95, 10-90, 20-80, 25-75, 35-65, 50-50, 60-40, 65-35, 70-30, 75-25, 85-15, 100-0] (Fig. 3.3).

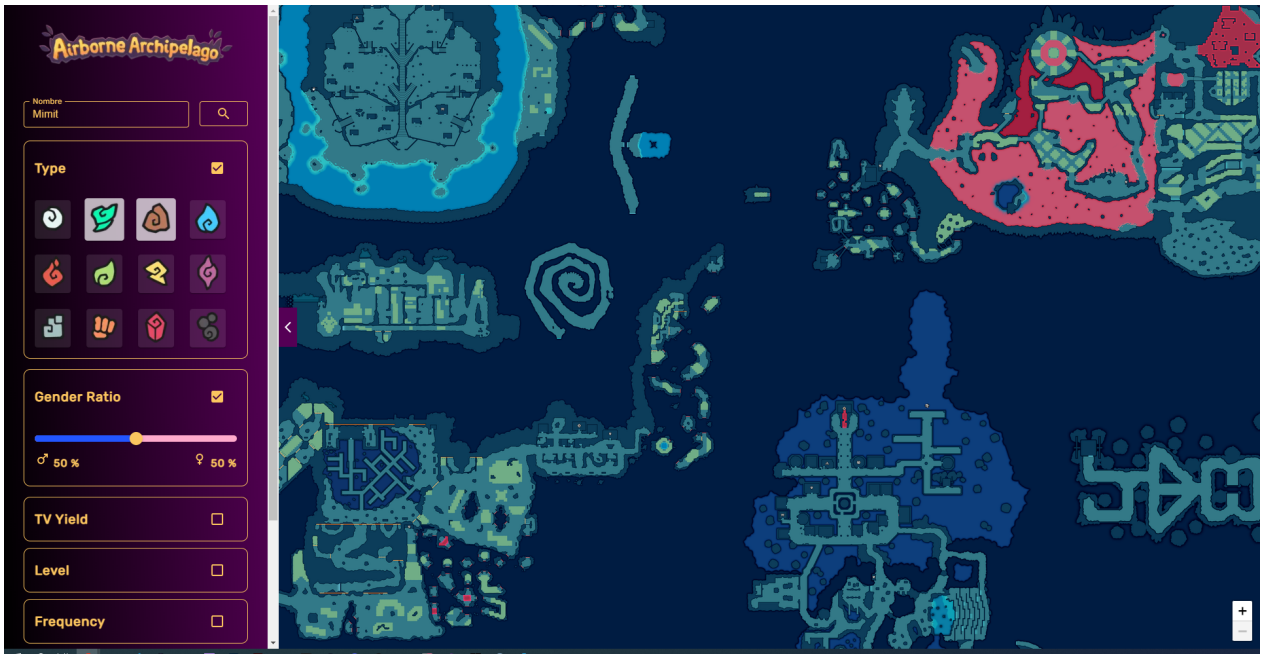

Figura 3.3: Pantalla con filtros por nombre, tipo y género.

- Filtro por valores de entrenamiento: en este videojuego cada criatura puede otorgar esfuerzo en un máximo de dos características. Es por eso que este filtro tiene limitada su selección a un máximo de dos (Fig. 3.4).
- Filtro por nivel: este filtro consta de un doble *slider*. El más bajo se corresponde con el nivel mínimo al que puede aparecer una criatura y el más alto al nivel máximo con el que puede aparecer una criatura (Fig. 3.4).
- Filtro por frecuencia: se corresponde al porcentaje de apariciones de una criatura. Este filtro es un *slider* con límites entre el 0 y el 100 (Fig. 3.4).
- Filtro por liberación: este filtro, al igual que los anteriores es un *slider*. Para establecer sus límites se realiza una consulta contra Elasticsearch para conocer la cota superior, ya que esta puede variar en función de las criaturas almacenadas (Fig. 3.4).

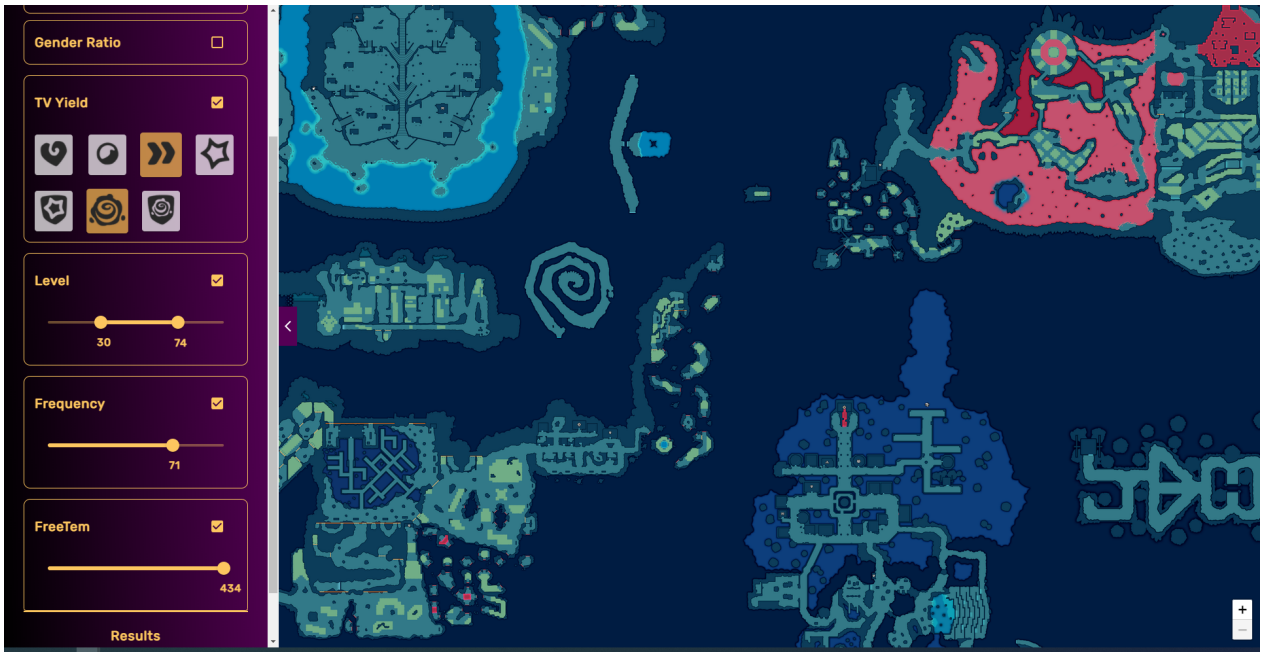

Figura 3.4: Pantalla con filtros por valores de entrenamiento, nivel, frecuencia de aparición y liberación.

Debajo de los filtros está la lista de resultados. Indicando en primer lugar el número de coincidencias y luego los lugares donde se pueden encontrar, agrupados siempre por la criatura. Pero, además de la ubicación se muestra la frecuencia de aparición, valor por el cual se ordenan los resultados mostrados. Cada elemento de esta lista es a su vez un botón que, si se hace clic en él, mueve el mapa a la zona de aparición de la criatura. Mostrando sus detalles (Fig. 3.5).

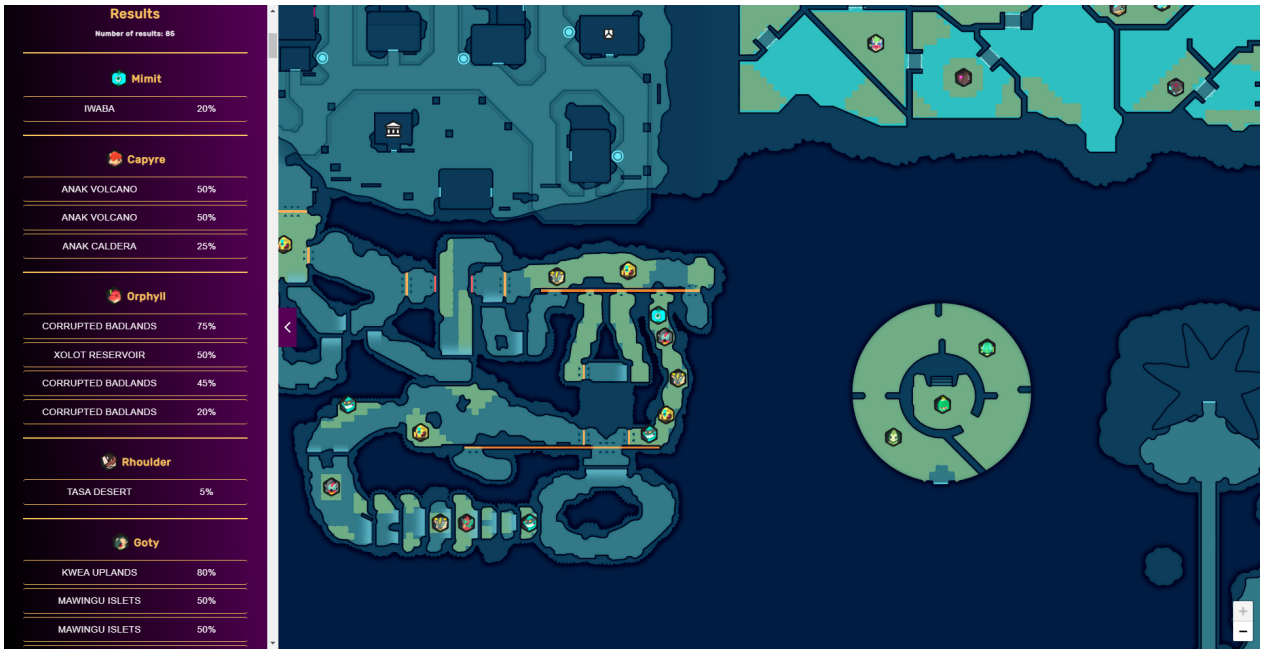

Figura 3.5: Pantalla con la lista de criaturas que coinciden con los criterios de búsqueda.

Dado que el número de marcadores resultante de una búsqueda puede tener un impacto negativo en el rendimiento de la aplicación, se decidió limitar el número de marcadores visibles a únicamente aquellos que se encontrasen dentro del campo de visión del usuario. Los demás se mostrarían únicamente si se realizase algún desplazamiento sobre el mapa y quedasen dentro del campo de visión.

Capítulo 4

Tecnologías empleadas

A continuación se enumeran las tecnologías empleadas en la realización de este proyecto:

1. Python es un lenguaje de programación interpretado de alto nivel y de propósito general. Además, es de código abierto y soporta diferentes paradigmas como programación orientada a objetos, programación imperativa y programación funcional.

2. Scrapy es un framework de *scraping* y *crawling* de código abierto, escrito en Python.

3. Elasticsearch es un servidor de búsqueda basado en Lucene. Provee un motor de búsqueda de texto completo, distribuido y con capacidad de multitenencia con una interfaz web RESTful y con documentos JSON.

4. React es una biblioteca Javascript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página.

5. Next.js es un marco de desarrollo de código abierto construido sobre Node.js que permite funcionalidades de aplicaciones web basadas en React.

6. React Leaflet es una biblioteca de JavaScript de código abierto que se utiliza para crear aplicaciones de mapas web. Concretamente se utiliza para este proyecto una adaptación de esta librería a React.

7. Tile cutter es una aplicación de escritorio para la generación de mapas a diferentes escalas basados en *tiles*.

Capítulo 5

Instalación y ejecución

Para poder hacer uso de esta aplicación es necesario tener instaladas las siguientes dependencias. Además, se acompañan las versiones en las que ha sido probado:

1. Python 3.8.8
    a) scrapy 2.5.1
    b) elasticsearch 7.15.1
    c) requests 2.26.0
2. Docker 20.10.7
    a) elasticsearch 7.15.1
3. Node 12.18.1
4. NPM 7.23.0

En primer lugar, es necesario tener levantado un contenedor con la imagen de ElasticSearch para poder indexar la información. Además, se configurará para que escuche los puertos 9200 y 9300. En este caso, se ejecuta activando los *CORS* y que permita el tráfico desde *localhost:3000*, que será el de nuestra aplicación.

```
docker run --name Elasticsearch -p 9200:9200 -p 9300:9300 -e "discovery.
    type=single-node" -e "http.cors.enabled=true" -e "http.cors.allow-
    origin=http://localhost:3000" docker.elastic.co/elasticsearch/
    elasticsearch:7.15.1
```

Seguidamente, desde la carpeta de *backend*, se instalarán las dependencias necesarias y se ejecutará Scrapy para que recupere la información de la wiki. El script *main.py* ya se encarga de hacer esto automáticamente, almacenado la información en ElasticSearch:

```
pip install -r requirements.txt
python main.py
```

Finalmente, se instalan las dependencias y se levanta la aplicación web desde la carpeta de *frontend*:

```
npm install
npm run dev
```

Después de seguir todos estos pasos se podrá acceder desde el navegador a la aplicación desde la dirección localhost:3000.