Steam Recommender

Objetivos

He desarrollado un sistema de recomendación de juegos de *Steam*, el objetivo de este proyecto es poder obtener una gran cantidad de datos de la web oficial de *Steam* y sugerir juegos que no han sido jugados por el usuario, basándose en usuarios que hayan jugado número de horas parecidas al usuario que estamos recomendando. Por lo tanto, he utilizado el filtrado colaborativo por usuario.

La aplicación consigue almacenar y obtener información de más de 5000 juegos, 200.000 acciones de usuario (horas jugadas y compra de juegos) y 22 categorías.

Descripción de las partes del proyecto

Para la obtención de datos aplicación se han utilizado distintas herramientas que ha hecho dotar a la aplicación de una mayor complejidad y de una mayor cantidad de datos. Primeramente, se obtienen los usuarios, los juegos y las acciones de los usuarios (compra de juegos y horas jugadas) de un csv que recopila a usuarios y que juegos han comprado y jugado, además almacena cuantas horas han jugado a ese título, lo cual podemos usarlo para el sistema de recomendación.

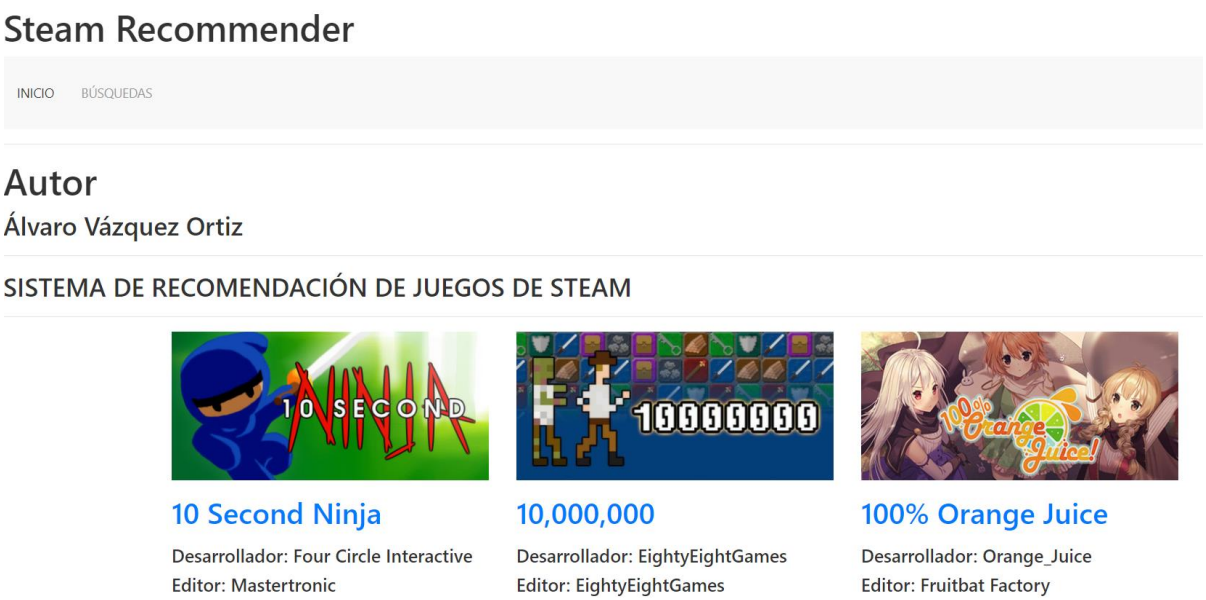
Como el csv recopila solo esa información, he decidido completar más la base de datos mediante el uso de scraping, para así almacenar los videojuegos con sus respectivas categorías y atribuir a los videojuegos un desarrollador y editor. Para realizar esto, necesitaba las urls de cada videojuego en la tienda de Steam para así poder obtener los datos. Para obtener cada url necesitaba el appId de cada videojuego, tras una investigación sobre cómo conseguirlo, conseguí acceder al api de la tienda de Steam y pude obtener un JSON que emparejaba el nombre del videojuego con su appId.

Al estar obteniendo información de un JSON en el que hay almacenado todos los juegos de Steam, 152362 registros el día que estoy escribiendo esto. Es muy ineficiente consultarlo cada vez que se requiera el appId de un videojuego, ya que JSON no tiene un selector directo que permita extraer un elemento que contenga un valor concreto. Es por ello por lo que decidí utilizar **Whoosh** para crear un esquema que contenga y empareje los videojuegos con sus respectivos appId. Además, se utiliza después este esquema para ir juego por juego de la Base de datos de Django (SQLite) y almacenar el appId. Tras ello se ha realizado el scraping comentado anteriormente con BeautifulSoup, componiendo las urls con las appId.

Como sistema de recomendación se ha utilizado el filtrado colaborativo basado en el usuario y se han implementado 3 funcionalidades. Por un lado, podemos ver las acciones tomadas, por un usuario, es decir, que juegos ha comprado y cuantas horas ha invertido en cada juego. También se puede obtener los videojuegos parecidos al que elijamos. Por último, los juegos que se recomiendan a un usuario concreto. Estas dos últimas funcionalidades del sistema de recomendación basan las recomendaciones en las horas jugadas de los usuarios, sin tener en cuenta si los usuarios han comprado el juego simplemente. He puesto el foco en las horas jugadas y además he hecho una traducción para simular que el usuario puntúa al juego en base a las horas jugadas (del 1 al 5).

Manual de uso

Al correr la aplicación, nos encontramos con la página de inicio, la cual tiene dos menús desplegables en las cuales están las funcionalidades. Debajo podemos ver ejemplos de videojuegos almacenados en nuestra base de datos.



Para la corrección de este proyecto, se recomienda comprobar las funcionalidades con la base de datos que esté en la entrega, pues para disponer de todos los datos, hay que realizar el scraping y tarda demasiado. Por ello, primero hablaré de las funcionalidades no relacionadas con el poblado de datos.

Primeramente, deberemos cargar el sistema de recomendación, para ello pondremos el ratón sobre el botón inicio, y se nos desplegará un menú, clicaremos sobre *Cargar el sistema de recomendación*. Una vez cargado, podremos utilizar todas las funcionalidades.

Si ponemos el ratón sobre *Búsquedas*, en el menú que se despliega podemos acceder a 3 funcionalidades:

- **Recomendar juegos a un usuario:** En esta sección hay un input para introducir la id de un usuario y un botón para buscar y que obtengamos 6 recomendaciones para un usuario. Dejo aquí algunas id de usuarios: 151603712, 92107940, 59945701, 11373749, 41883322, 227083521, 65229865, 68049243, 92393218, 71603645.
- **Mostrar juegos similares a uno:** En esta sección se puede introducir el appId de un videojuego y la aplicación mostrará 6 videojuegos similares basándose en los usuarios que juegan a este videojuego. AppIds de prueba: 10, 30, 40, 50, 60, 100, 220, 570, 2630, 3300.
- **Buscar usuario:** Esta sección permite introducir la id de un usuario y mostrará las acciones tomadas por ese usuario. Se muestran los juegos comprados, y los juegos jugados, si la acción es que el juego se ha jugado, se mostrarán las horas jugadas a ese juego. Se pueden utilizar las ids de la sección de recomendar juegos a un usuario.

Por último, el poblado de datos se encuentra en el menú de inicio, primero deberemos acceder a la sección *Poblar Base de datos*. Deberemos iniciar sesión como administrador de la aplicación para ello y poblará la base de datos con los juegos y acciones de usuario, pero todavía faltará las appId de los juegos, las imágenes de los juegos, los editores, etc. Para completar la información, deberemos ejecutar el scraping, accediendo a la sección *Cargar categorías y datos de videojuegos*, en el que también se requiere acceder como administrador. El proceso tardará demasiado tiempo, por eso recomiendo comprobar el resto de funcionalidades con la base de datos de la entrega.