READING

Adrián Cantón Fernández

Daniel Carpio Camacho

Rafael Fresno Aranda

Contenido

1. Introducción

El proyecto realizado consiste en una aplicación web usando el framework de Django en el cual los usuarios puntúan libros en función de sus gustos y la aplicación le recomienda libros que les puedan gustar.

Hay que destacar que la aplicación es de uso educativo, con lo cual el número de libros que se almacenan es muy pequeño. También hay que comentar que el sistema de recomendación está implementado de modo que sea fácil de ver a simple vista que son libros parecidos (de igual categoría), pero para uso comercial, sería necesario realizar algunas modificaciones para garantizar una mejor recomendación (por ejemplo utilizar el resumen de los libros para su recomendación).

2. Descripción técnica

Reading es una aplicación web escrita en lenguaje Python, con la ayuda del framework Django. Este framework define una arquitectura similar a la conocida MVC (modelo-vista-controlador), pero la denomina MTV (modelo-plantilla-vista), aunque su funcionamiento es prácticamente el mismo.

Para el almacenamiento de datos, se utiliza la base de datos relacional SQLite. Este sistema ofrece bases de datos ligeras, fáciles de utilizar y compatibles con todos los sistemas operativos y lenguajes. Su uso se basa en el acceso a un fichero que almacena los datos correspondientes; en este caso, dicho archivo se llama “db.sqlite3”.

Para poblar la base de datos de la aplicación, se utiliza “scrapping”, es decir, se obtienen los datos de otra página web rastreando su código HTML y extrayendo la información necesaria. Esto se realiza mediante el paquete “beautifulsoup4” de Python, que permite de forma fácil llevar a cabo el scrapping. Nuestra aplicación consume la información de la web de la Casa del Libro, una famosa librería, que ofrece todos los detalles necesarios de cada libro. El scrapping se realiza de la siguiente manera:

1. Para cada categoría de libros:
   1.1. Obtener todas las páginas
   1.2. Para cada página:
      1.2.1. Obtener la lista de libros
      1.2.2. Para cada libro:
         1.2.2.1. Guardar la información básica
         1.2.2.2. Acceder a la página que contiene información detallada
         1.2.2.3. Guardar la información detallada
2. Mostrar el tiempo total que se ha tardado en hacer el scrapping

```
def scrap(url, category):
        total = 0
        f = urllib.request.urlopen(url)
        html = f.read()
        f.close()
        soup = BeautifulSoup(html, 'lxml')
        pages = soup.find('span', class_='fnt09im').text.split(" ")[3]
        for i in range(scrapping_page_number):
                f = urllib.request.urlopen(url + '/p' + str(i + 1))
                html = f.read()
                f.close()
                soup = BeautifulSoup(html, 'lxml')
                books = soup.find_all('div', class_='mod-list-item')
                for book in books:
```

```
title = book.find('a', class_='title-link').string
bookURL = "https://www.casadellibro.com" + book.find('a', class_='title-link')['href']
author = book.find('div', class_='mod-libros-author').text
imageURL = book.find('img', class_='img-shadow')['data-src']
f = urllib.request.urlopen(bookURL)
html2 = f.read()
f.close()
detailssoup = BeautifulSoup(html2, 'lxml')
details = detailssoup.find('div', class_='book-description')
detailsli = []
if details is not None:
        bookdata = details.find('ul', class_='list07')
        if bookdata is not None:
                detailsli = bookdata.find_all('li')
npages = 0
editorial = ""
lengua = ""
binding = ""
for detail in detailsli:
        if detail.text[0:15] == "Nº de páginas: ":
                npages = int(detail.text[15:18])
        if detail.text[0:11] == "Editorial: ":
                editorial = detail.text[11:]
        if detail.text[0:8] == "Lengua: ":
                lengua = detail.text[8:]
        if detail.text[0:16] == "Encuadernación: ":
                binding = detail.text[16:]
if details.find('div', class_='col02') != None:
        resumen = details.find('div', class_='col02').find('p')

synopsis = ""
if resumen != None:
        synopsis = resumen.text
print(title)
if Book.objects.filter(title=title).first() is None and synopsis != "":
        Book.objects.create(title=title, bookURL=bookURL, author=author,
                            coverURL=imageURL, npages=npages,
                            editorial=editorial, language=lengua, category=category,
                            binding=binding, synopsis=synopsis)
```

Este método guarda cerca de 300 libros, que son suficientes para realizar pruebas sobre las diferentes herramientas que proporciona la aplicación.

La primera de estas herramientas es un buscador de libros, implementado mediante un sistema de recuperación de información. Esto se lleva a cabo gracias al paquete “Whoosh”, que permite indexar elementos y recuperarlos de forma rápida y eficiente. Es necesario definir un esquema, que es la estructura básica de los elementos que se indexarán, y posteriormente se realiza el propio indexado. El esquema que se utiliza en Reading es uno muy simple donde los campos a tener en cuenta son la ID, el título, el autor, la editorial y el resumen del libro. De estos campos, solamente se almacena el ID, es decir, que aunque podemos buscar en cualquiera de los campos indicados anteriormente, solo podremos recuperar el ID de los libros que obtengamos como resultado de la búsqueda. Esto es intencionado y no es un problema, ya que a partir del ID de un libro podemos sacar todos sus detalles desde la base de datos. Los campos ID y editorial son de tipo “id”, el autor de tipo “keyword”, y el título y la sinopsis de tipo “text”. El indexado de los elementos se realiza tras haber hecho scrapping y haber guardado todos los libros en la base de datos. Hacer ambas cosas simultáneamente provocaba un error de Python debido a que se excedía el límite de recursividad permitido, por lo que se optó por hacerlo por separado. Para realizar búsquedas sobre el sistema, simplemente hay que indicar la consulta y los campos donde se quiere aplicar dicha consulta, y se obtendrán los resultados

correspondientes. Se puede encontrar más información sobre cómo realizar búsquedas en Reading en el manual de usuario.

```
def createSchema():
        booksSchema = Schema(id=ID(stored=True), title=TEXT, author=KEYWORD,
                             editorial=ID, synopsis=TEXT)
        if not os.path.exists("booksIndex"):
            os.mkdir("booksIndex")
            create_in("booksIndex", schema=booksSchema)

def indexBooks():
        books = Book.objects.all()
        ix = open_dir("booksIndex")
        writer = ix.writer()
        for b in books:
            writer.add_document(id=str(b.id), title=b.title, author=b.author,
                                synopsis=b.synopsis)
            writer.commit()
```

Otra herramienta de la aplicación es un sistema de recomendación. Mediante un algoritmo desarrollado para este sistema, se pueden obtener libros recomendados para un usuario en función de los libros que este mismo usuario ha puntuado previamente. Reading permite que los usuarios registrados y con sesión iniciada puedan puntuar los libros que no haya puntuado con anterioridad. Dado que el número de usuarios y puntuaciones del sistema es considerablemente bajo, utilizar un sistema de recomendación colaborativo, tal y como se usa en páginas reales, no resultaría efectivo dado que no habría suficientes datos para generar buenas recomendaciones. Por ello, Reading utiliza un sistema basado en contenido, donde al obtener recomendaciones para un usuario solamente se tienen en cuenta las puntuaciones anteriores del mismo usuario. El algoritmo que calcula la la puntuación o potencial interés de un usuario por un libro es el siguiente:

1. Si el usuario no ha puntuado el libro X:
    1.1. total = 0
    1.2. cuenta = 0
    1.3. Para cada puntuación existente del usuario para un libro Y:
        1.3.1. actual = 0
        1.3.2. Si la categoría de X e Y coincide, actual += 3
        1.3.3. Si el autor de X e Y coincide, actual += 2
        1.3.4. Si la similaridad del título de X e Y es mayor que 0.75, actual += 1
        1.3.5. actual *= puntuación dada a Y
        1.3.6. total += actual
        1.3.7. cuenta ++
    1.4. total = total / cuenta

```
def load_rs(request):
        start = time.time()
        shelf = shelve.open('dataRS.dat')
        users = models.Rating.objects.values('user').distinct()
        userrecom = {}
        for user in users:
            idusu = user['user']
            user = get_object_or_404(User, id=idusu)
            ratings = models.Rating.objects.filter(user=user)
            books = models.Book.objects.all()
            userrecom.setdefault(idusu, [])
            bookrecom = {}
            for book in books:
                if ratings.filter(book=book.id).first() is None:
```

```
                        total = 0.0
                        count = 0
                        for rating in ratings:
                                actual = 0
                                if rating.book.category == book.category:
                                        actual += 3
                                if rating.book.author == book.author:
                                        actual += 2
                                similarity = SequenceMatcher(None, rating.book.title,
                                                             book.title).ratio()
                                if similarity > 0.75:
                                        actual += 1
                                actual *= rating.rating
                                total += actual
                                count += 1
                        total = total / count
                        bookrecom[book.id] = total
                sortedrecom = sorted(bookrecom.items(), key=lambda x: x[1], reverse=True)[:10]
                userrecom[idusu] = sortedrecom
        shelf['userrecom'] = userrecom
        shelf.close()
        stop = time.time()
        return render(request, 'finished.html', {'time': stop-start, 'process': 'carga del sistema de recomendación'})
```

En este algoritmo, “total” es el valor final que se asocia a cada libro no puntuado por un usuario. Posteriormente, una vez calculadas todas las puntuaciones, se ordenan de mayor a menor, se obtienen las diez mejores, y se almacenan en un fichero, llamado “dataRS.dat”, con la ayuda del paquete “shelve”. No es muy eficiente que estos cálculos se lancen cada vez que un usuario puntúe un libro, sino han de ejecutarse manualmente de forma regular para que las nuevas puntuaciones se tengan en cuenta. A la hora de recuperar las recomendaciones para un usuario, simplemente habrá que extraer del fichero donde se almacenaron las puntuaciones la lista de libros correspondiente

3. Manual de instalación

Para la instalación de este software, es necesario tener instalado Python 3, además de los siguientes paquetes: Django, beautifulsoup4, lxml y Whoosh.

3.1. Instalación de Python

La instalación de Python puede depender de su sistema operativo.

Instalación en Windows.

Para la instalación de Python en cualquier versión de Windows, es necesario que vaya a su página oficial: https://www.python.org/. En esta página podrá encontrar el instalador de dicho software.

Instalación para distribuciones Linux.

Es muy común que Python venga ya instalado en su sistema operativo si usa una distribución Linux. Si desea comprobarlo, escriba en su terminal:

$ python3 --version

Esto le indicará qué versión de Python tiene instalado.

En el caso de que no tenga instalado Python, deberá instalando usando alguno de estos comandos:

- Debian o Ubuntu:

$ sudo yum install python3

- Arch Linux:

$ sudo pacman -S python3

- Fedora

$ sudo dnf instalar python3

En versiones anteriores de Fedora, puede que salga un error de que no se encuentra el comando 'dnf'. En ese caso usar 'yum' en su lugar.

- openSuse:

$ sudo zypper install python3

Tras finalizar la instalación puede comprobar si todo ha salido bien lanzando el comando:

$ python3 –versión

3.2. Paquetes

- Django: Framework que nos permite desarrollar la web.
- Beautiful Soup: Librería de Python que nos permite analizar y extraer documentos de una página HTML.
- Lxml: Parser para Beautiful Soup que nos permitirá extraer de una forma más rápida los datos.
- Whoosh: Librería que nos permitirá indexar los datos para realizar búsquedas rápidamente.

3.3. Instalación de paquetes

Para instalar los paquetes, se usará el comando 'pip', que viene junto con la instalación de Python.

NOTA: Se puede requerir permisos de superusuario para la instalación de los paquetes.

Para la instalación de los paquetes, nos podemos ayudar del archivo 'requirements.txt' que viene junto al proyecto. Para ello, nos movemos a la carpeta del proyecto con la terminal y ejecutamos:

$ pip install -r requirements.txt

No obstante, la instalación de los paquetes también se puede hacer manualmente de la siguiente forma:

$ pip install Django beautifulsoup4 lxml Whoosh

3.4. Ejecución del software

Una vez tengamos todos los paquetes instalados, podemos ejecutar el software. Para ello, nos movemos con una terminal a la carpeta del proyecto y ejecutamos estos tres comandos:

$ python3 manage.py makemigrations

$ python3 manage.py migrate

$ python3 manage.py runserver

Con esto, el software se estará ejecutando en el puerto 8000. Se puede acceder mediante http://localhost:8000.

4. Manual de usuario

4.1. Cargando los datos

Una vez tengamos el software ejecutándose, nos deberá salir esta ventana:

Imagen 1: Index

Lo primero que debemos hacer será cargar todos los datos necesarios. Para ello, hemos dispuesto de un conjunto de botones en el menú de navegación. Los botones que necesitamos se encuentran en el dropdown llamado 'Carga de datos'. Para cargar todos los datos, ejecutamos todas las opciones que nos ofrece. Estas opciones son: hacer el scrapping para obtener los datos, crear usuarios con distintas puntuaciones y cargar los datos del sistema de recomendación. Esto puede tardar más o menos tiempo en función del ordenador en el que se ejecute y de la velocidad de la conexión a internet del mismo.

Imagen 2: Carga de datos - Scrapping

Sabremos que se ha ejecutado sin problemas una vez que salga el tiempo que ha tardado en pantalla.

4.2. Búsqueda de libros

Cuando volvamos al index de nuestra web, ya nos aparecerán los datos correctamente. En esta ventana, nos mostrará los 10 libros mejor puntuados según los usuarios.

Imagen 3: Index con carga de datos realizada

Si accedemos a 'Todos los libros', nos mostrará todos los libros que se encuentran en la base de datos. Para facilitar la carga, estos vendrán paginados de 10 en 10.

Imagen 4: Listado de libros

También podemos buscar los libros. Esto se hace accediendo al buscador mediante el botón 'Buscar libro' en el menú de navegación.

Imagen 5: Buscador de libros

Si, por ejemplo, buscamos 'george' nos devolverá dos resultados. Debido a que la página donde obtenemos los datos puede incluir nuevos libros según pase el tiempo, los resultados pueden variar.

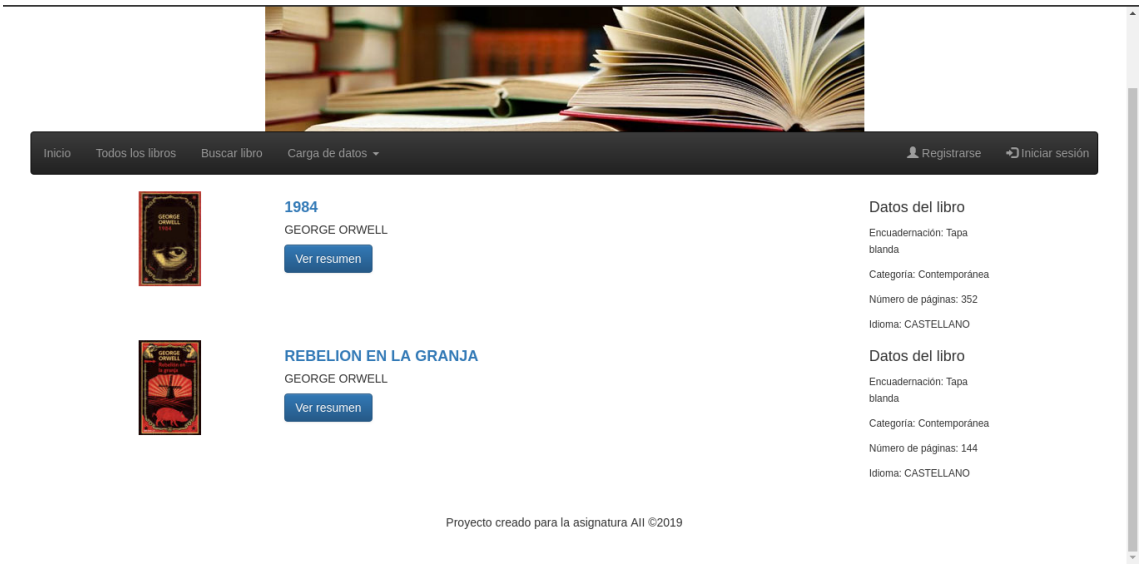

Imagen 6: Resultado de búsqueda

En todas estas páginas, si hacemos click en 'Ver resumen' que se encuentra debajo del título, nos desplegará un breve resumen del mismo.

Imagen 7: Resumen de un libro

Además, si hacemos click en el título del libro, nos llevará a una página web externa donde se puede comprar el libro.

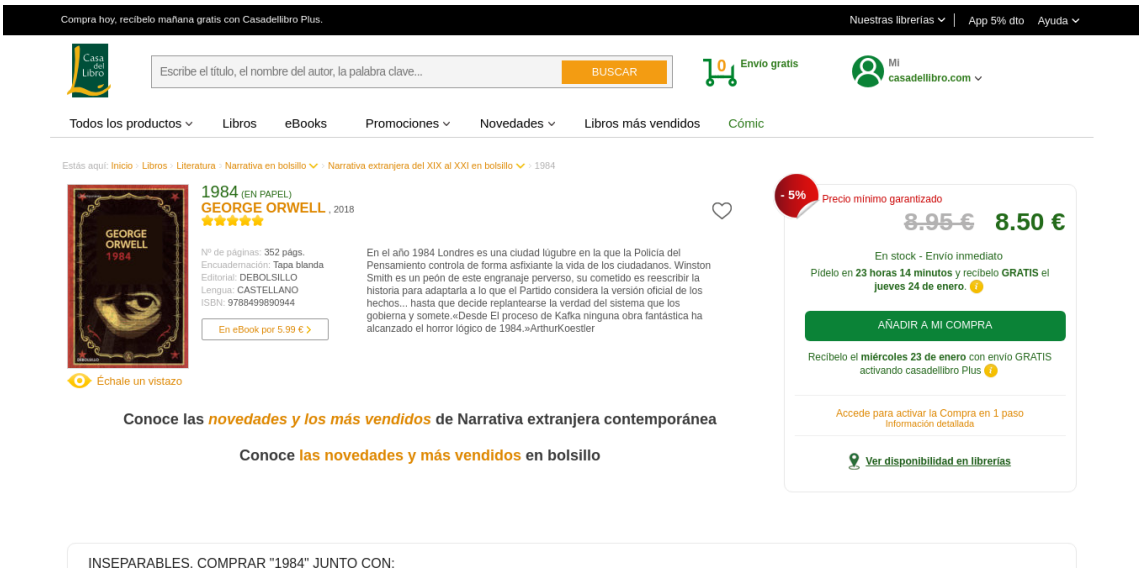

Imagen 8: Vista del libro en web externa

4.3. Usuarios

Nos podemos registrar mediante el botón 'Registrarse', que se encuentra en la parte derecha del menú de navegación.

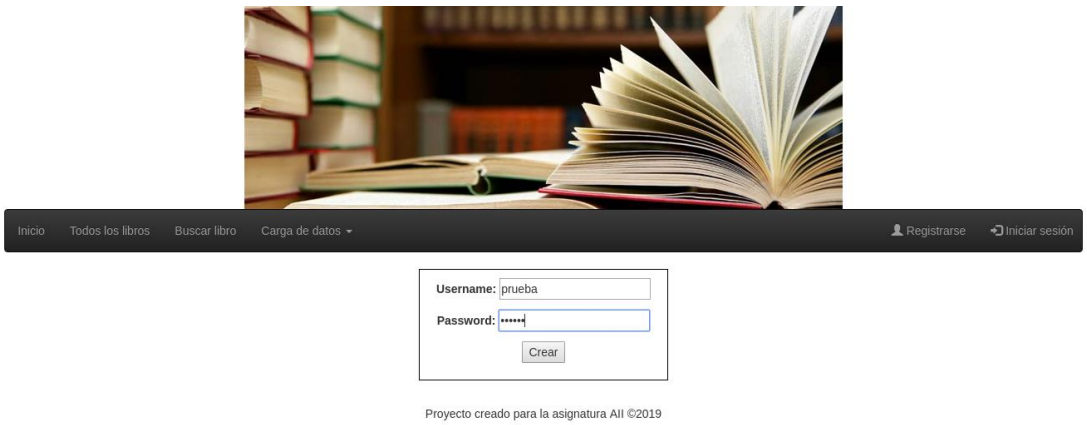

Imagen 9: Registro de usuarios

Una vez nos hayamos registrado, podemos hacer login mediante el botón 'Iniciar Sesión' que se encuentra a la derecha del botón de registrarse.

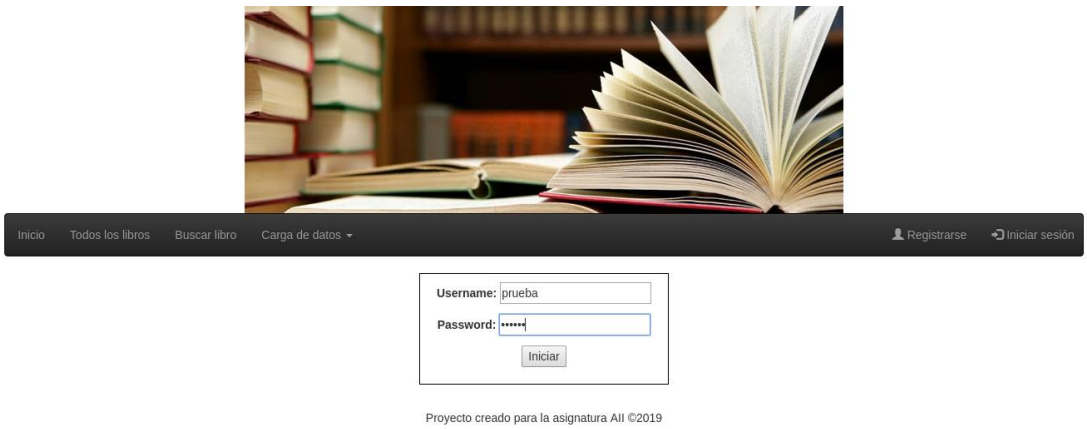

Imagen 10: Login

Una vez nos hayamos logueado, nos devolverá al index.

Imagen 11: Index con sesión iniciada

Además, tendremos la opción de cerrar la sesión haciendo click en 'Cerrar sesión' en la parte derecha del menú de navegación.

Al generar las puntuaciones también se han generado una serie de usuarios que tienen los siguientes nombres y contraseñas:

- user1/user1
- user2/user2
- user3/user3
- user4/user4
- user5/user5
- user6/user6
- user7/user7

4.4. Puntuaciones de libros

Una vez que estemos logueados, al lado de cada libro nos aparecerá un nuevo botón 'Puntuar'. Si hacemos click en él, nos llevará a otra ventana donde podremos puntuar dicho libro.

Imagen 12: Crear puntuación

En cambio, si ya lo habíamos puntuado antes, nos saldrá:

Imagen 13: Puntuar libro ya puntuado

Disponemos de una vista que nos permite ver todas las puntuaciones que hemos realizado. Se accede mediante el botón 'Puntuaciones':

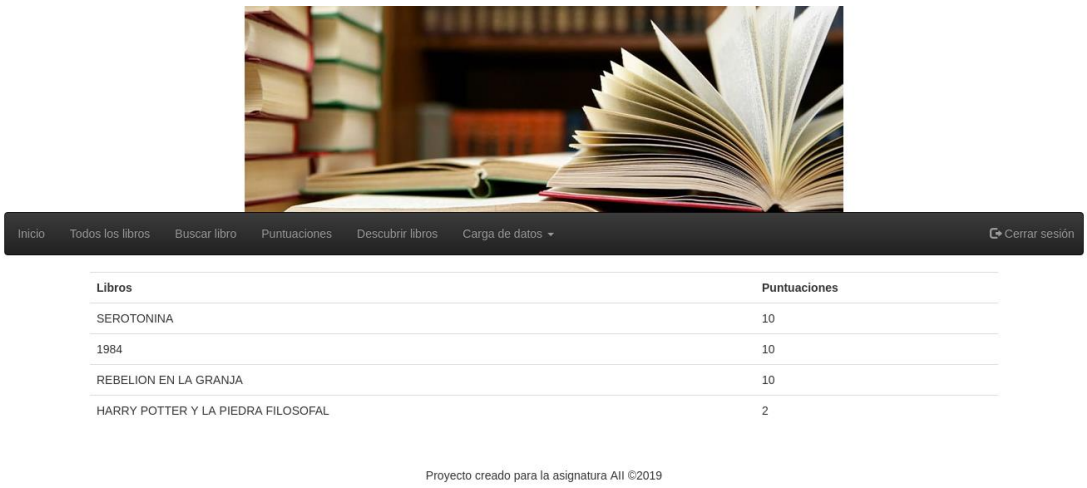

Imagen 14: Listado de puntuaciones realizadas

Mediante las valoraciones que hagamos, el sistema nos puede recomendar libros. Para ello, hacemos click en 'Recomendar libros'.

NOTA: Para esto es necesario que volvamos a cargar el sistema de recomendaciones. Esto se hace haciendo click en 'Carga de datos' > 'Cargar sistema de recomendación'.

Imagen 15: Libros recomendados