10 DE FEBRERO DE 2023

MANUAL DE LA APLICACIÓN

PRÁCTICA ESPUBLICO

MANUEL C. LEÓN RIVERA
ESPUBLICO

Contenido

Instalación

1. Tecnologías requeridas para el despliegue

Java 1.8.

➔ Enlace: https://www.oracle.com/es/java/technologies/javase/javase8-archive-downloads.html#license-lightbox

2. Configuraciones previas

La práctica se ha hecho sobre la API de Spring-boot, desplegando una aplicación web con un contenedor de Servlets TOMCAT embebido en el .JAR (si se prefiere, la aplicación puede exportarse en formato .WAR para desplegarse en otro servidor).

La ruta a la aplicación por defecto es: **http://localhost:8080/**

Antes de generar el .jar de la aplicación, se deben configurar los ficheros de propiedades ubicados en:

➔ **<RUTA_APLICACIÓN>**\src\main\resources**spring-properties.properties**

En este fichero hay que modificar los siguientes parámetros:

➔ datasource.username=[PONER AQUÍ EL **USER** DE LA BASE DE DATOS]
- Por ejemplo: test

➔ datasource.password=[PONER AQUÍ LA **PASS** A LA BASE DE DATOS]
- Por ejemplo: 1234

➔ datasource.driverClassName=[PONER AQUÍ EL **DRIVER** DE LA BASE DE DATOS]
- Por ejemplo: com.mysql.cj.jdbc.Driver

➔ datasource.url=[PONER AQUÍ LA **URL DE LA BASE** DE DATOS]:
- Estructura: URL + PUERTO + ESQUEMA + ?serverTimezoneUTC
  - **Nota**: es importante poner serverTimezoneUTC para que en la lectura de los atributos date del CSV sean compatibles con la BD.
- Por ejemplo: jdbc:mysql://localhost:3306/ESPUBLICO?serverTimezone=UTC

➔ datasource.showSql=[SE PUEDE DEJAR POR DEFECTO]

- ➔ datasource.auto=[POR DEFECTO ESTÁ **CREATE**, PERO SE PUEDE PONER UPDATE PARA QUE NO PISE LAS TABLAS CADA VEZ QUE INICIAMOS LA APLICACIÓN]
- ➔ datasource.lote=[EN LAS INSERCIONES A BD, AL HACER COMMIT SE HARÁN INSERCIONES POR PAQUETES SEGÚN ESTE PARÁMETRO, POR DEFECTO **100**]

- ➔ ruta_base: [**RUTA** BASE DONDE SE ENCUENTRAN LOS DOCUMENTOS **CSV**]
  - Por ejemplo: ruta.base=C:\\Users\pepito\\Practica_esPublico

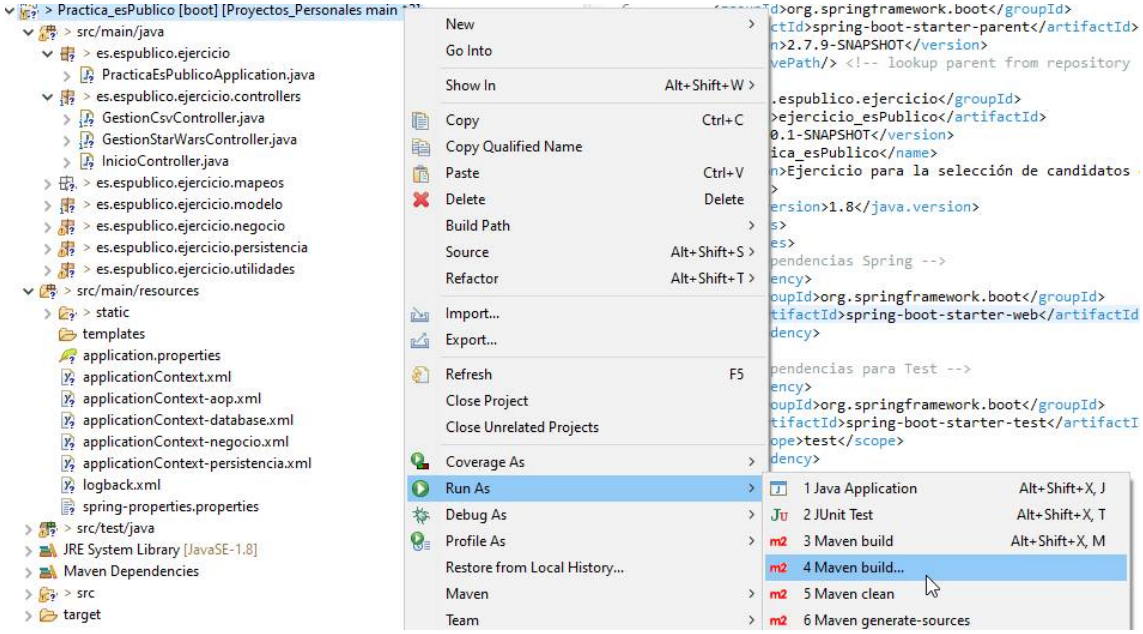
- ➔ fichero.rutaLecturaCSV: [RUTA FINAL AL DOCUMENTO **CSV CON SU NOMBRE**]
  - Para estos dos parámetros, un ejemplo sería:
    - fichero.rutaLecturaCSV=${ruta.base}\\RegistroVentas2.csv

La aplicación también cuenta con un fichero LOG que se puede configurar en el fichero:

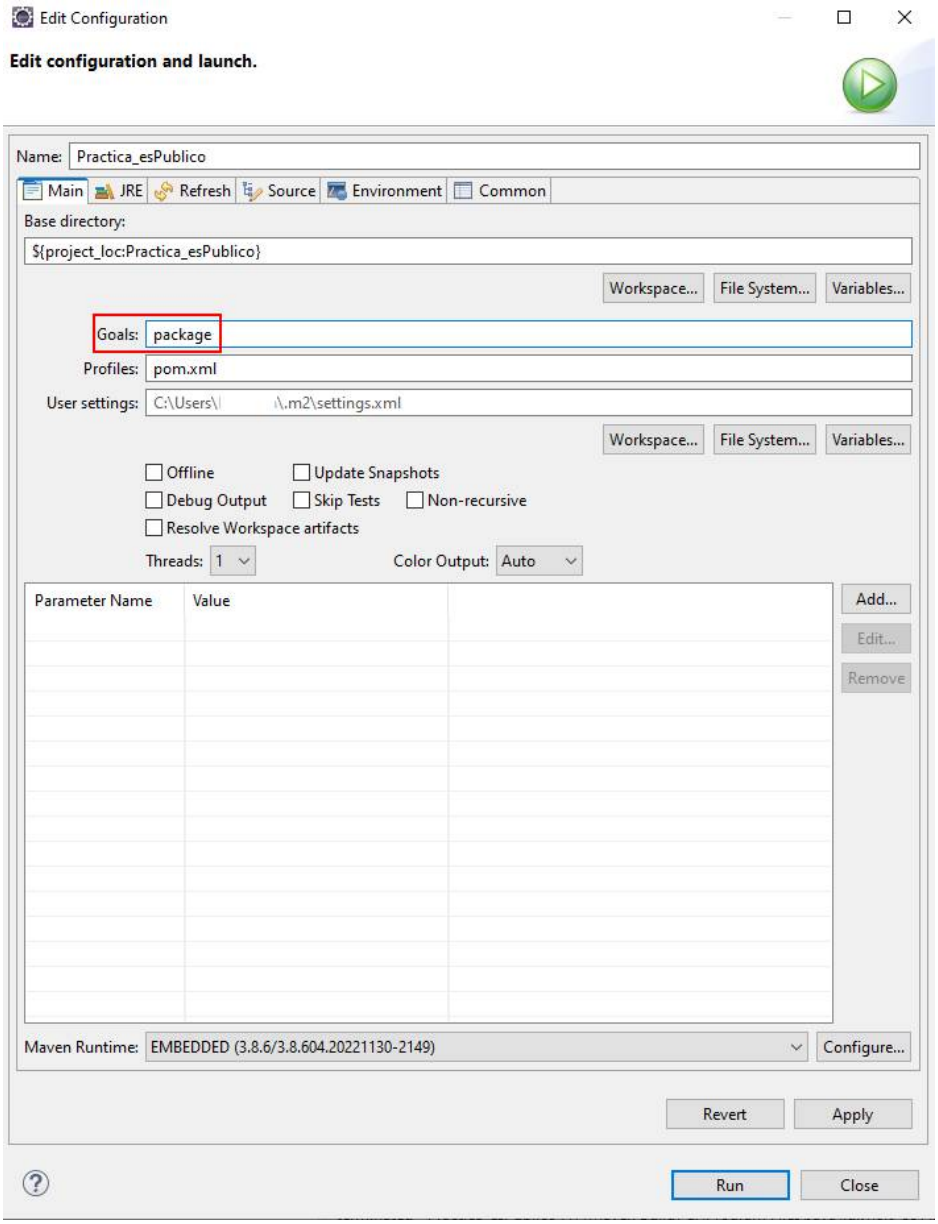
- ➔ **<RUTA_APLICACIÓN>**\src\main\resources**logback.xml**
  - Aquí, simplemente, basta con configurar el siguiente parámetro:
    - <property name=”LOG_PATH”>
    - Por ejemplo:
      <property name=”LOG_PATH”
      value=”C:\\Users\\pepito\\Practica_esPublico\\esPublico.log”/>

3. Despliegue

Se puede exportar la aplicación ejecutándola con **Maven build**:

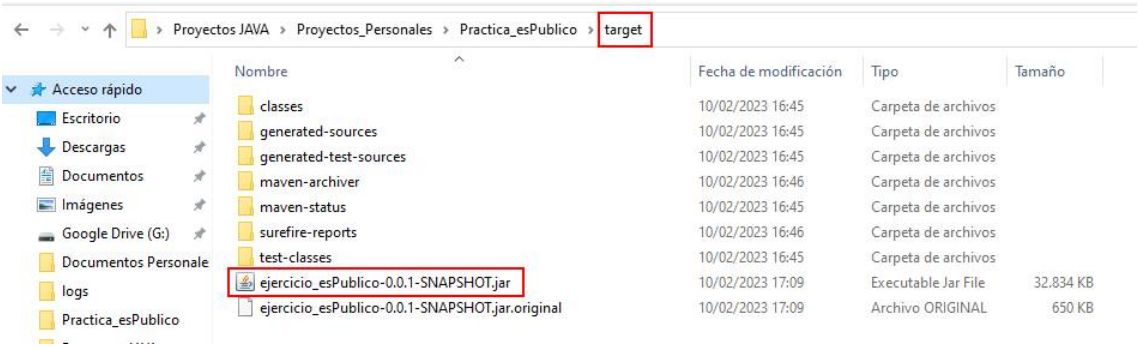


En “**goals**” escribimos “**package**” y le damos a **Run**.



La aplicación comenzará a construir el proyecto y pasará por los tests unitarios.

Encontraremos el .jar en la carpeta “target” en la raíz del proyecto:



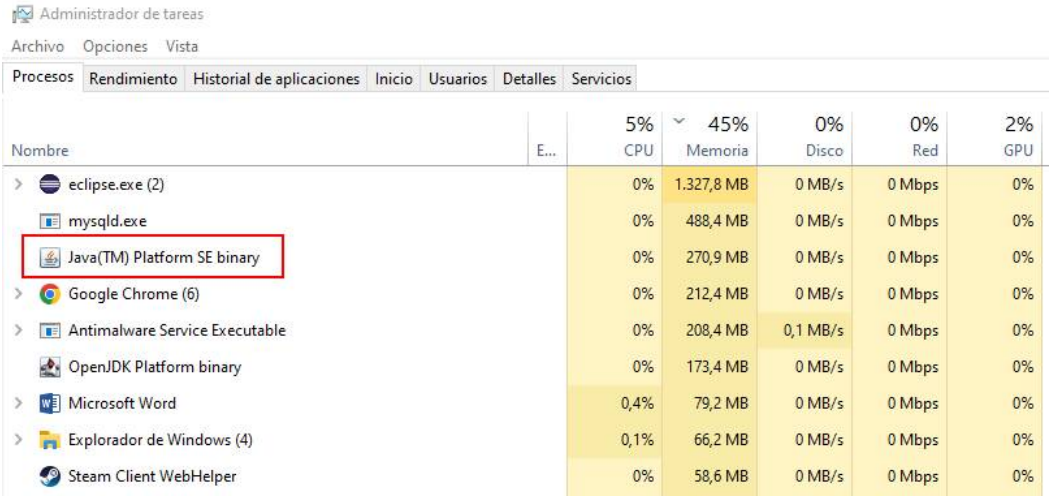
El fichero se llama “**ejercicio_esPublico-0.0.1-SNAPSHOT.jar**”

Si las configuraciones descritas en el punto 2 de este manual son correctas, cuando ejecutemos el fichero “**ejercicio_esPublico-0.0.1-SNAPSHOT.jar**” debería funcionar con normalidad. Para ello abrimos el navegador y escribimos la siguiente dirección:

http://localhost:8080

Nos debería aparecer la siguiente página:

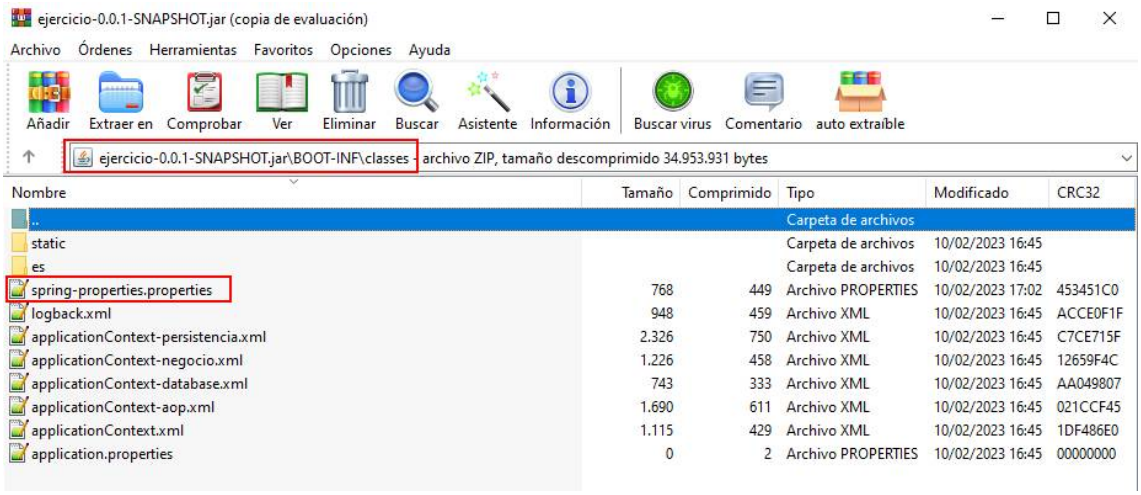
Para matar el proceso, CTRL+ALT+SUPR y suprimimos el proceso JAVA:



4. (OPCIONAL) Ejecución de la aplicación modificando directamente el archivo "ejercicio_esPublico-0.0.1-SNAPSHOT.jar" que se ha entregado con la práctica.

Podemos optar por modificar directamente la configuración sobre el fichero "**ejercicio_esPublico-0.0.1-SNAPSHOT.jar**" del ejercicio. Para ello lo abrimos con cualquier programa descompresor y nos vamos a la carpeta:

<RUTA_EJERCICIO>\BOOT-INF\classes



Aquí podemos modificar los ficheros:

- ➔ spring-properties.properties
- ➔ logback.xml

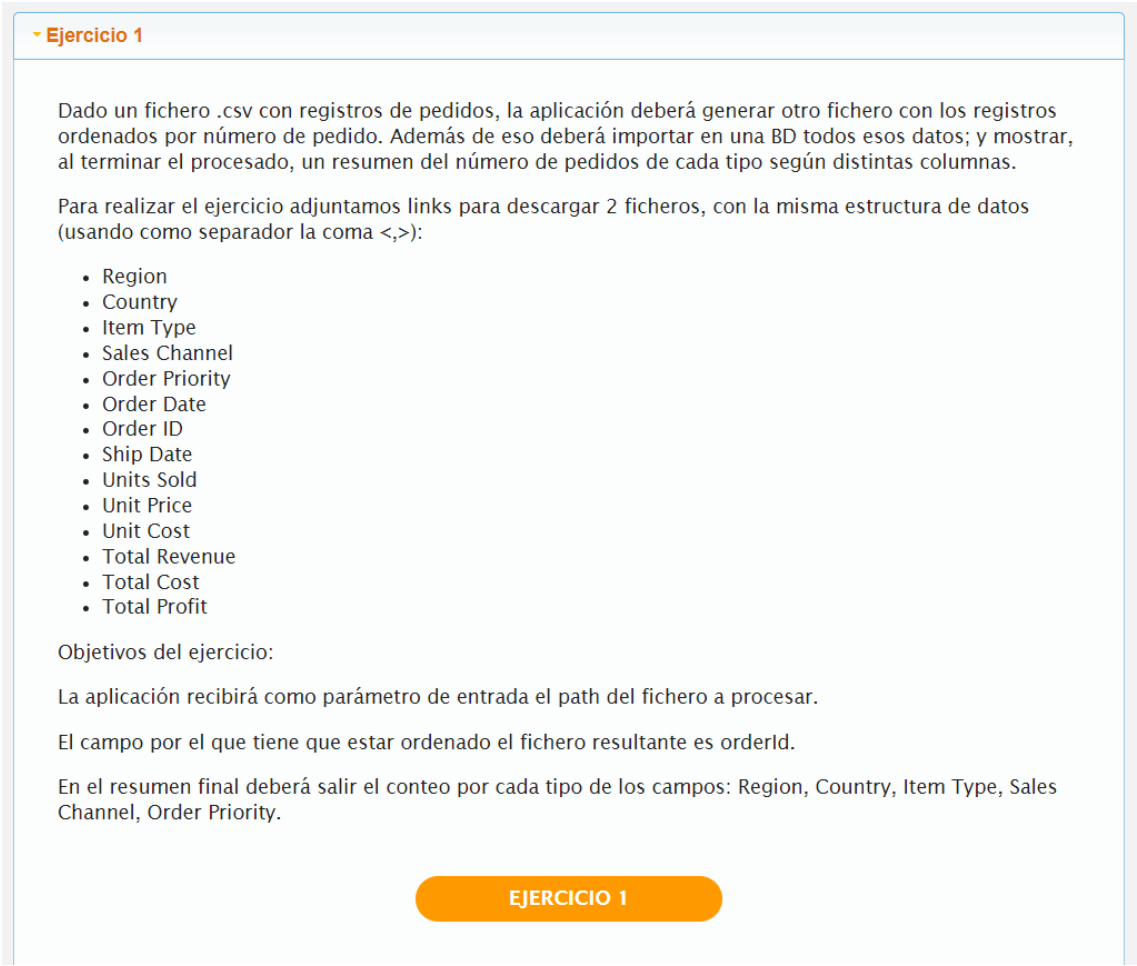
Cuando guardemos, el programa compresor nos preguntará si queremos actualizar el paquete, le diremos que sí (no pasa nada ya que no hay que recompilar la aplicación, son simples ficheros de configuración).

Ya tendríamos la aplicación lista para ser ejecutada por JAVA y vista por el navegador en la url: **http://localhost:8080/**

Tecnologías utilizadas en la aplicación

- SPRING-BOOT
  - SPRING AOP
  - SPRING CORE
  - SPRING MVC
- HIBERNATE
- MAVEN
- JUNIT
- JQUERY
  - Datatables
  - Jquery-UI
  - MultiSelect
  - jDialog
  - AJAX
- OPENCSV

Ejercicio 1



Al clicar sobre el Ejercicio 1, se va a efectuar una llamada AJAX que va a interceptar la aplicación realizando 3 procesos:

1) Lectura del fichero “RegistroVentas3.csv” mediante opencsv. Se van a cargar en memoria todos los objetos de la clase VENTA.

2) Una vez que tenemos la lectura completa, se van a insertar en base de datos cada uno de los objetos leídos. Este punto tarda entre 2 y 3 minutos. Se han seguido diversas estrategias para optimizar el tiempo de inserción:

   - Escritura con un buffer de registros en memoria: He intentado hacer ciclos de 100 en 100, incluso de 1000 en 1000 registros que, una vez listos a insertar, se realiza session.flush() y limpiamos la memoria, pero esto no ha mejorado el rendimiento, de hecho lo ha empeorado (hasta 12 minutos):

```
BufferLineas=1000 -> Tiempo de ejecución: 725722mseg. -> 12,09 minutos
```

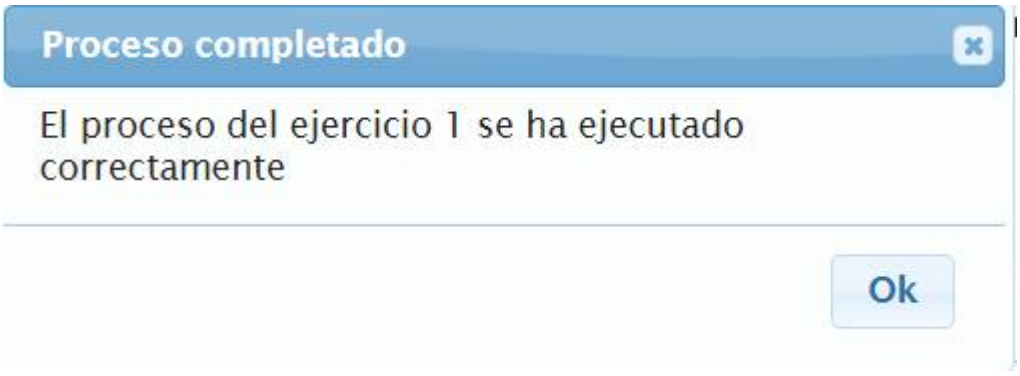
- He intentado utilizar HibernateTemplate para lanzar peticiones template.saveAll(List<Venta>), pero tampoco ha mejorado el rendimiento.

- Finalmente, he encontrado una alternativa modificando la propiedad hibernate.jdbc.batch.size de Hibernate, que configura la inserción por lotes, es decir, en lugar de insertar 1 a 1 cada registro, efectúa una inserción por paquetes según este parámetro, he aquí los resultados según dicha configuración:

```
datasource.lote = 200 -> Tiempo de ejecución: 200167mseg -> 3,33 minutos
datasource.lote = 500 -> Tiempo de ejecución: 201662mseg -> 3,36 minutos
datasource.lote = 10000 -> Tiempo de ejecución: 205474mseg -> 3,42 minutos
datasource.lote = 100   -> Tiempo de ejecución: 198201mseg -> 3,30 minutos
datasource.lote = 50    -> Tiempo de ejecución: 202100mseg -> 3,36 minutos
```

Finalmente, se ha optado por configurar un lote de 100 registros, que es el que parece más eficiente. No obstante, no parece que Hibernate sea la mejor opción para la inserción masiva de datos.

3) Si todo ha ido bien, los datos aún están en memoria. Los ordenamos por la columna OrderId y escribimos en el fichero de salida "VentasOrdenadasPorOrderId.csv" que estará en la ruta que hayamos puesto en el fichero de configuración de la aplicación. La pantalla indicará que el proceso ha finalizado.



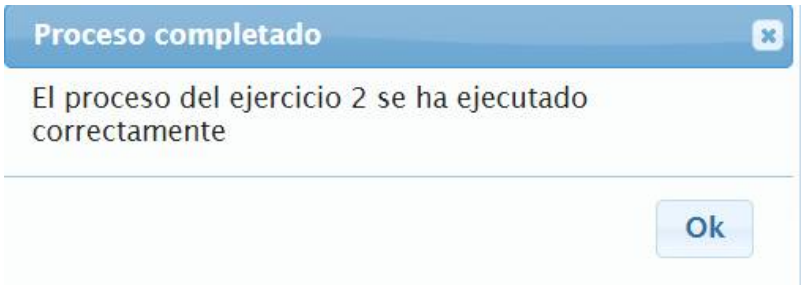
Ejercicio 2

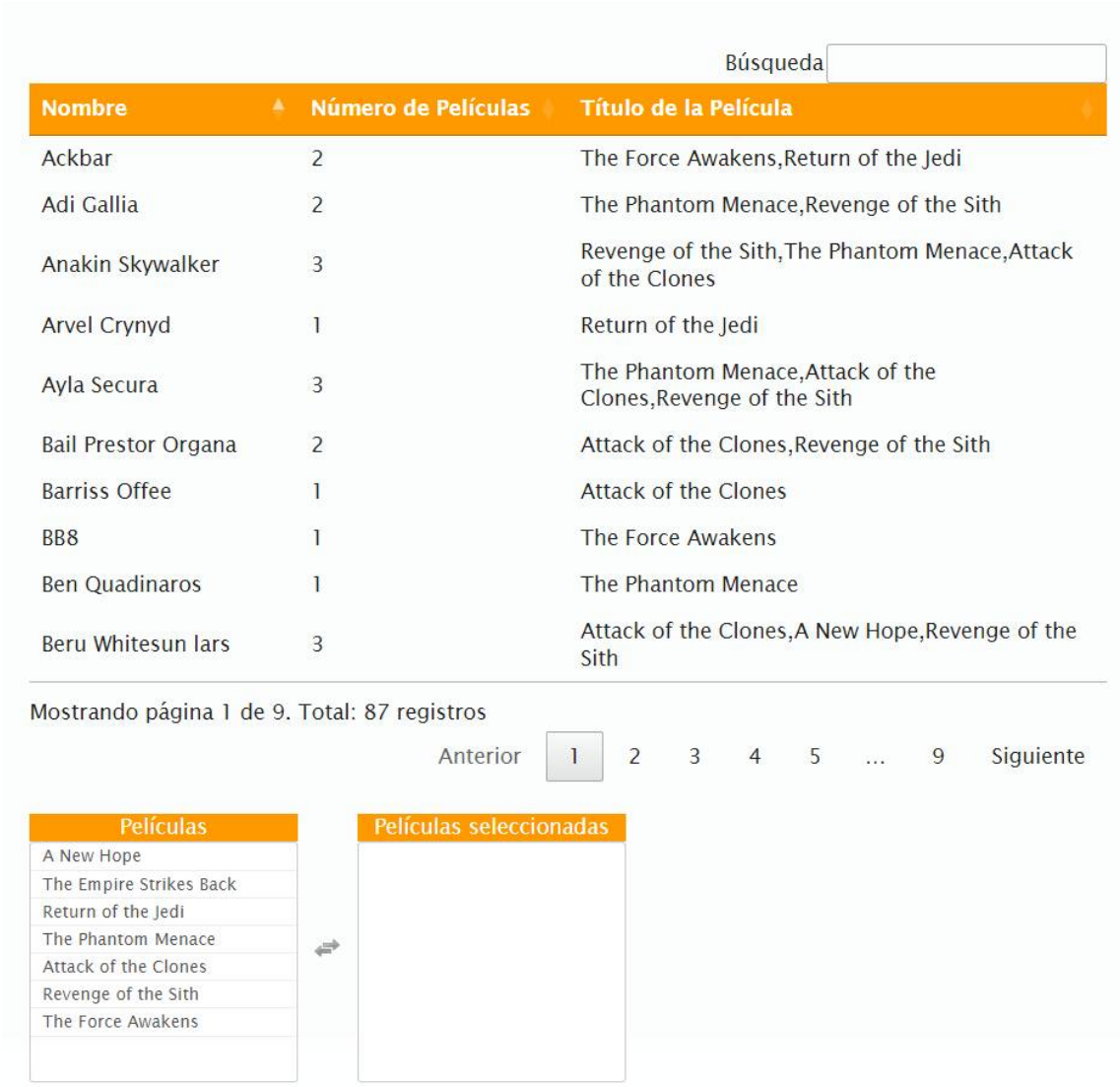
1) Al pinchar en el botón Ejercicio 2, se realiza una petición AJAX para que realice un proceso de obtención de datos de la API: https://swapi.py4e.com/api, que se traerá las entidades Film, People y Starships con sus relaciones. Cuando los datos estén en memoria, se realiza el proceso de relación entre entidades.

2) Una vez tenemos los datos en memoria, junto con sus relaciones, se vuelca todo a base de datos, persistiéndolos.

3) Si todo ha ido bien, se realiza una consulta que devolverá una relación de los personajes con el número de películas en las que aparecen y los títulos de las películas separados por “,”

4) Este es el resultado si todo ha ido bien:





Búsqueda

Nombre	Número de Películas	Título de la Película
Ackbar	2	The Force Awakens,Return of the Jedi
Adi Gallia	2	The Phantom Menace,Revenge of the Sith
Anakin Skywalker	3	Revenge of the Sith,The Phantom Menace,Attack of the Clones
Arvel Crynyd	1	Return of the Jedi
Ayla Secura	3	The Phantom Menace,Attack of the Clones,Revenge of the Sith
Bail Prestor Organa	2	Attack of the Clones,Revenge of the Sith
Barriss Offee	1	Attack of the Clones
BB8	1	The Force Awakens
Ben Quadinaros	1	The Phantom Menace
Beru Whitesun lars	3	Attack of the Clones,A New Hope,Revenge of the Sith

Mostrando página 1 de 9. Total: 87 registros

Anterior 1 2 3 4 5 ... 9 Siguiente

5) Se puede apreciar en la imagen anterior, que la tabla se ha rellenado automáticamente y que, además, se ha incluido un multiselect en la parte inferior con el nombre de todas las películas registradas. Esta información se la ha traído también al finalizar la primera consulta con otra llamada AJAX.

6) El multiselect permite seleccionar películas o quitarlas. En cada interacción, se realiza otra llamada AJAX que devuelve el actor que pilota la nave que más aparece en las películas seleccionadas, por ejemplo, si seleccionamos todas las películas, la petición devolvería:

7) La consulta realizada en este último caso sería la siguiente (formato SQL para que se entienda mejor):

```
SELECT p.name as pName, st.name as stName, COUNT(f.ID)as total
                        FROM people p JOIN people_starships ps ON (p.ID = ps.ID_PEOPLE)
                        JOIN starships st ON (ps.ID_STARSHIP = st.ID)
                        JOIN films_starships fs ON (fs.ID_STARSHIP = st.ID)
                        JOIN films f ON (fs.ID_FILM = f.ID)
                        WHERE f.ID IN("https://swapi.py4e.com/api/films/1/",
                                      "https://swapi.py4e.com/api/films/2/",
                                      "https://swapi.py4e.com/api/films/3/",
                                      "https://swapi.py4e.com/api/films/4/",
                                      "https://swapi.py4e.com/api/films/5/",
                                      "https://swapi.py4e.com/api/films/6/",
                                      "https://swapi.py4e.com/api/films/7/")
                        GROUP BY pName, stName ORDER BY total DESC, pName ASC LIMIT 1;
```

Resultado:

	pName	stName	total
▶	Chewbacca	Millennium Falcon	4

Si eliminamos el tope (1), nos devolvería el siguiente resultado:

pName	stName	total
Chewbacca	Millennium Falcon	4
Han Solo	Millennium Falcon	4
Lando Calrissian	Millennium Falcon	4
Nien Nunb	Millennium Falcon	4
Biggs Darklighter	X-wing	3
Jek Tono Porkins	X-wing	3
Luke Skywalker	X-wing	3
Wedge Antilles	X-wing	3
Anakin Skywalker	Naboo fighter	2
Boba Fett	Slave 1	2
Chewbacca	Imperial shuttle	2
Gregar Typho	Naboo fighter	2
Han Solo	Imperial shuttle	2
Luke Skywalker	Imperial shuttle	2
Obi-Wan Kenobi	Jedi starfighter	2
Padmé Amidala	Naboo fighter	2
Plo Koon	Jedi starfighter	2
Anakin Skywalker	Trade Federatio...	1
Anakin Skywalker	Jedi Interceptor	1
Arvel Crynyd	A-wing	1

Como realmente nos interesa el primero, sólo obtenemos este resultado y es el que se devuelve en la consulta.