

UNIVERSIDAD AUTÓNOMA



PRÁCTICAS DE SISTEMAS INFORMÁTICOS I

PRÁCTICA 2

Memoria

Autores:

Adrián FERNÁNDEZ
Santiago GONZÁLEZ-CARVAJAL

Pareja 11
Grupo 1401

29 de octubre de 2018

Índice

1. Descripción de los archivos	2
2. Instrucciones de uso	3
3. Detalles de implementación	5
4. Resultados obtenidos	6
5. Conclusiones	21

1. Descripción de los archivos

webflix.py

Fichero que contiene toda la lógica de la aplicación, está implementado en **Python 2** utilizando el módulo **Flask**. Se encarga de procesar la información de la aplicación para mostrar el sitio web de forma dinámica.

webflix.wsgi

Conjunto de instrucciones que haciendo uso del módulo **wsgi**, lanzan la aplicación en el servidor **Apache 2**.

base.html

Fichero **html** base, que contiene las barras superiores, lateral e inferior. El resto de ficheros **html** heredan de él, de manera que solo se encargan de definir la zona de contenido.

details.html

Muestra los detalles de una película y permite añadir la misma al carrito; en caso de que ya lo esté, permite eliminarla del mismo.

history.html

Muestra el historial de compras de un usuario. El historial es una lista de fechas desplegables. De tal forma que al hacer “click” sobre cada fecha, aparecen todas las películas compradas ese día. También incluye un formulario para incrementar el saldo del usuario.

index.html

Muestra la página principal con todas las películas, permitiendo buscar por nombre de película o por categoría. El botón con el logo **WebFlix** nos redirige a esta página estemos en la página que estemos.

login.html

Permite a un usuario iniciar sesión en la aplicación web.

register.html

Permite a un usuario registrarse en la aplicación web.

shoppingCart.html

Muestra el carrito de compra de la sesión, permitiendo eliminar películas del carro y realizar la compra tanto de una sola película como de todas al mismo tiempo (añadidas previamente al carro). Solo permitirá comprar los artículos si se ha iniciado sesión y el usuario tiene saldo suficiente.

styles.css

Contiene el formato de los elementos **html** utilizados en el diseño de la página web.

passwordCheck.js

Script **JQuery** encargado de ir verificando dinámicamente la fortaleza de la contraseña que se va introduciendo en el formulario de registro.

reloadNumberOfUsers.js

Script **AJAX** que muestra el número actual de usuarios conectados a la aplicación web (el número mostrado es aleatorio).

validateRegistration.js

Script **JavaScript** que comprueba si los campos introducidos al registrarse son válidos.

dropDown.js:

Script **JQuery** usado para mostrar los menús desplegables del historial de compras.

2. Instrucciones de uso

Para correr la aplicación en localhost, se deberá configurar **Apache 2** de manera que tenga los permisos necesarios en la carpeta **public_html**. Así mismo, deben darse a Apache 2 todos los permisos en la carpeta de usuarios, además de los permisos de lectura y escritura en todos los documentos de dicha carpeta. Importante tener instalado el módulo **wsgi**, ya que Apache 2 ha de ejecutar el fichero **webflix.wsgi**. Por último, debe añadirse un **virtualenv** de nombre **s1pyenv**.

Para acceder a la página principal se deberá introducir la siguiente **URL** en la barra del navegador: <http://localhost/~username/webflix.wsgi/>.

Registrar un usuario

Para registrar un usuario nuevo, se debe acceder a la página principal de la aplicación y hacer "click" en el botón *Register*, el cual nos llevará a la página de registro. Una vez ahí debemos llenar los campos mostrados con datos válidos, si no lo son, el navegador nos mostrará una ayuda para rellenarlos de forma correcta. Si

introducimos los datos correctamente y pulsamos *Confirm* se registrará un usuario en la aplicación.

Acceder con un usuario registrado

Para acceder con un usuario, se debe acceder a la página principal y hacer “click” en el botón *Sign in*, el cual nos llevará a la página de acceso. Una vez ahí debemos introducir nuestro nombre de usuario y contraseña. Si no son correctos, el navegador mostrará un mensaje de error; si son correctos, accederemos a la aplicación como un usuario registrado y en la barra superior se mostrará el dicho usuario.

Página principal

Desde la página principal tenemos acceso a todas las películas y podemos buscar o bien por nombre de película, o bien por categoría.

Detalles de película

Al hacer “click” sobre la imagen de una película desde la página principal accedemos a los detalles de dicha película. Desde la página de detalles podemos agregar una película al carrito de la compra o borrarla del mismo si ya está en él.

Carrito de la compra

Accediendo desde el botón ”My cart” tendremos una lista de las películas que hemos añadido al carrito, pudiendo comprar cada película (botón con monedas) o quitarla del carrito (botón papelera). Además, disponemos de un botón que nos permite comprar todas las películas del carrito a la vez.

Finalizar compra

Al intentar finalizar una compra, el sistema comprueba, en primer lugar, si hemos iniciado sesión, y si no lo hemos hecho nos redirige a la página de inicio de sesión; en segundo lugar, comprueba si nuestro saldo es suficiente para realizar la compra que estamos tratando de hacer, y en el caso de que no lo sea nos devuelve un mensaje de error. Si ambas comprobaciones han sido positivas, el usuario compra la película.

Mostrar historial de un usuario

A esta funcionalidad solo pueden acceder los usuarios que han iniciado sesión. El usuario dispone de una lista de fechas en las que ha comprado alguna película, y al hacer “click” en cualquiera de esas fechas (mediante JQuery) se despliega un menú con la lista de películas y precios correspondiente a dicha fecha.

Número de usuarios conectados

Empleando AJAX, se actualiza cada 3 segundos el número de usuarios conectados a la aplicación (que aparece en la barra superior con el formato Users: N).

Medidor de fortaleza de contraseña

En el formulario de registro, y concretamente en el campo de introducir contraseña, se ha implementado la funcionalidad de, mediante JQuery, ir midiendo la fortaleza de la contraseña introducida dinámicamente. Los niveles son: 'Too short', 'Weak', 'Good' y 'Strong'.

3. Detalles de implementación

Se puede dividir la implementación de la práctica en tres partes principales:

Templates y CSS

En esta parte hemos decidido crear un fichero base del que heredaran todos los demás (de la manera comentada anteriormente). En cuanto a los elementos 'html', hemos seguido los consejos proporcionados en el enunciado de la práctica. Nótese que todas los 'htMLS' son plantillas ('templates') generados dinámicamente. Por el uso de Apache 2 ha sido necesario usar la función `url_for()`. En esta parte, siguiendo la documentación de Flask, también hemos creado una variable, `SCRIPT_ROOT`, en la que guardábamos la ruta hasta el directorio raíz, de manera que esta fuera accesible desde los scripts en JavaScripts (ya que en ellos no se puede usar la función `url_for()`).

Flask

En esta parte, hemos proporcionado todas las funciones que generan el contenido dinámico para 'rellenar' las plantillas 'html'. La funcionalidad pedida era bastante 'estricta', por lo que la implementación de las funciones no daba lugar a demasiadas decisiones de diseño. Cabe destacar, quizás, la necesidad de obtener en tiempo de ejecución el directorio raíz en el que se está ejecutando la aplicación. Esto se debe a que la aplicación está corriendo en Apache 2.

Por otro lado, para el uso de 'cookies' y 'sesiones' hemos seguido la información obtenida en Internet (veremos las referencias más adelante).

También, cabe destacar cómo hemos implementado que al borrar una película, la aplicación redirija a la página desde la que se ha borrado la película (sin importar si ha sido borrada desde la página de detalles o la del carrito). Para conseguir esta funcionalidad hemos hecho uso de `request.referrer`.

También comentar que hemos ahorrado mucha comprobación de errores en Python gracias a la interfaz de usuario dinámica. Por ejemplo, a un usuario que no ha iniciado sesión no le aparece el botón para acceder a su historial de compras.

JavaScript

Ha sido necesario implementar 4 funciones: la primera para validar los elementos del formulario donde cabe destacar que hemos decidido que solo se admitan tarjetas VISA (mediante JavaScript); la segunda, para medir la fortaleza de una contraseña mientras el usuario la va escribiendo, dependiendo de la longitud de la misma, que mezcle mayúsculas y minúsculas y que contenga números (mediante JQuery); la

tercera, para actualizar, utilizando AJAX, el número de usuarios conectados a la aplicación; por último, la cuarta, para mostrar o esconder los menús desplegables cuando se hace 'click' sobre una fecha (mediante JQuery).

Referencias importantes

A continuación incluimos una lista de las referencias más importante para el desarrollo de nuestra práctica (además de las especificadas en el enunciado de la práctica):

1. <http://blog.luisrei.com/articles/flaskrest.html>
2. <https://flask-login.readthedocs.io/en/latest/#how-it-works>
3. <https://www.programcreek.com/python/example/76929/flask.request.referrer>
4. <https://www.formget.com/password-strength-checker-in-jquery/>
5. <https://stackoverflow.com/questions/5052543/how-to-fire-ajax-request-periodically>
6. <http://flask.pocoo.org/docs/1.0/patterns/jquery/#where-is-my-site>
7. <https://www.emenia.es/creando-un-menu-vertical-desplegable-con-jquery/>
8. <https://www.w3resource.com/javascript/form/credit-card-validation.php>

Además de todas las referencias anteriores, también hemos usado como referencia las diapositivas de teoría y algunos otros sitios web para algunas consultas no destacables.

4. Resultados obtenidos

A continuación vamos mostrando los resultados obtenidos para cada página de la aplicación:

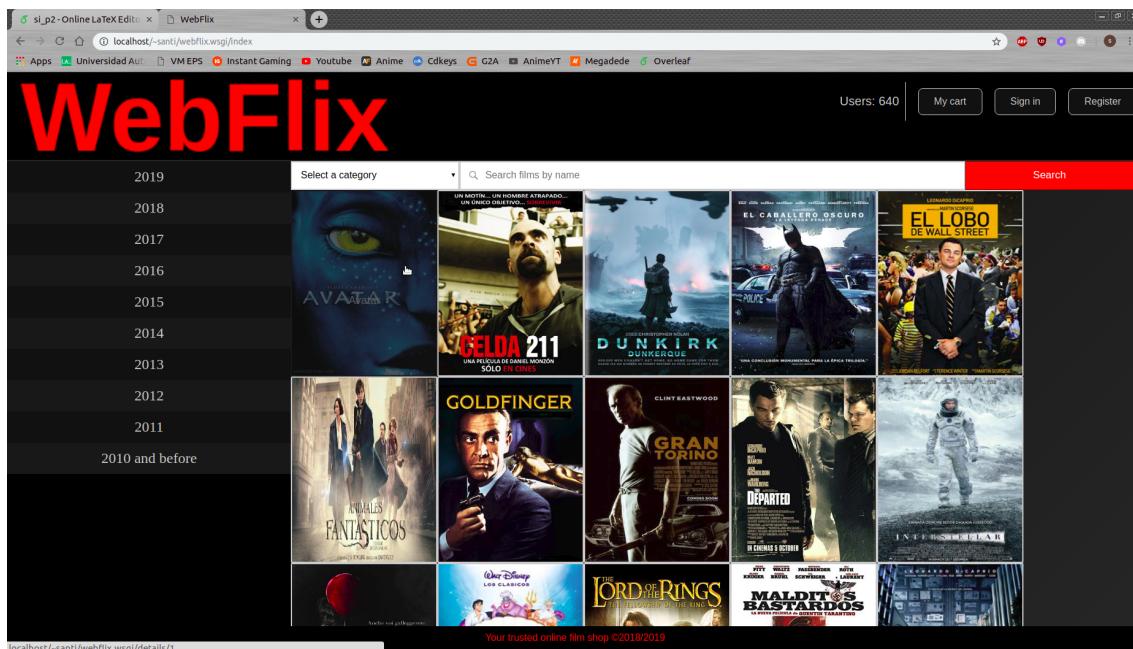


Figura 1: Página principal por defecto.

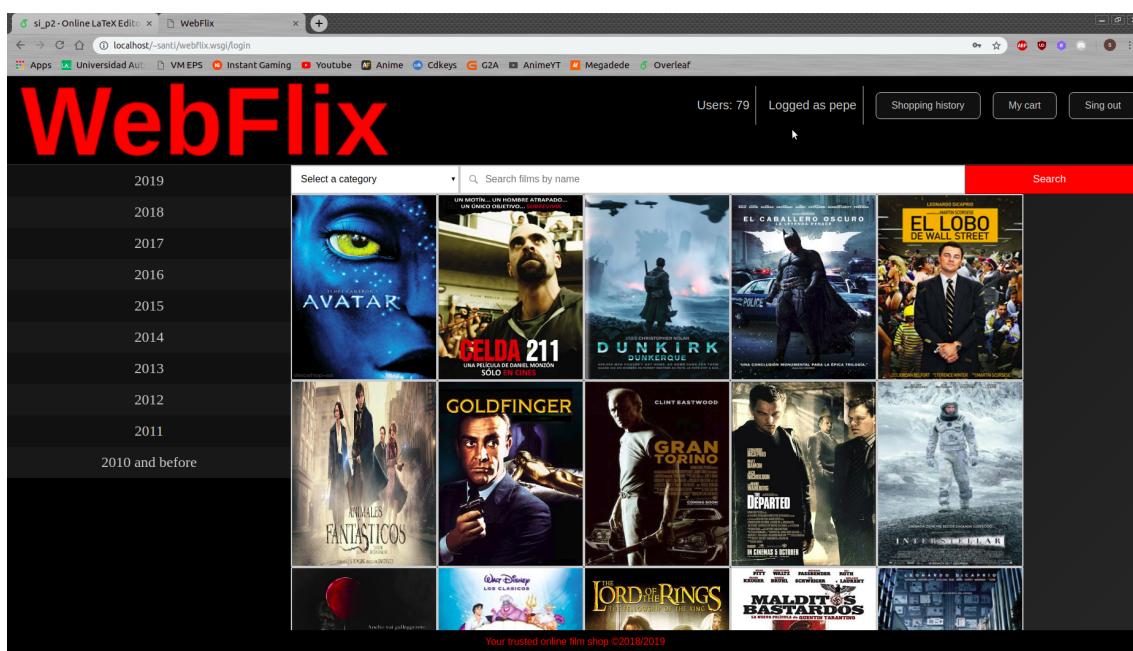


Figura 2: Página principal para usuario loggeado.

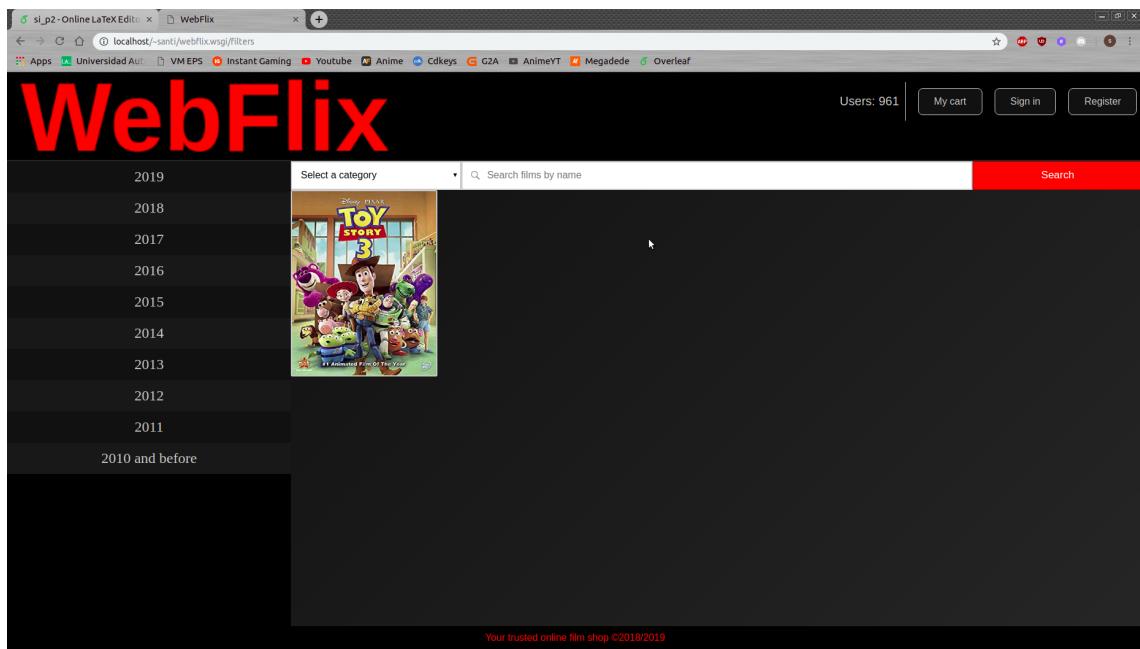


Figura 3: Filtrado por nombre 'toy'.

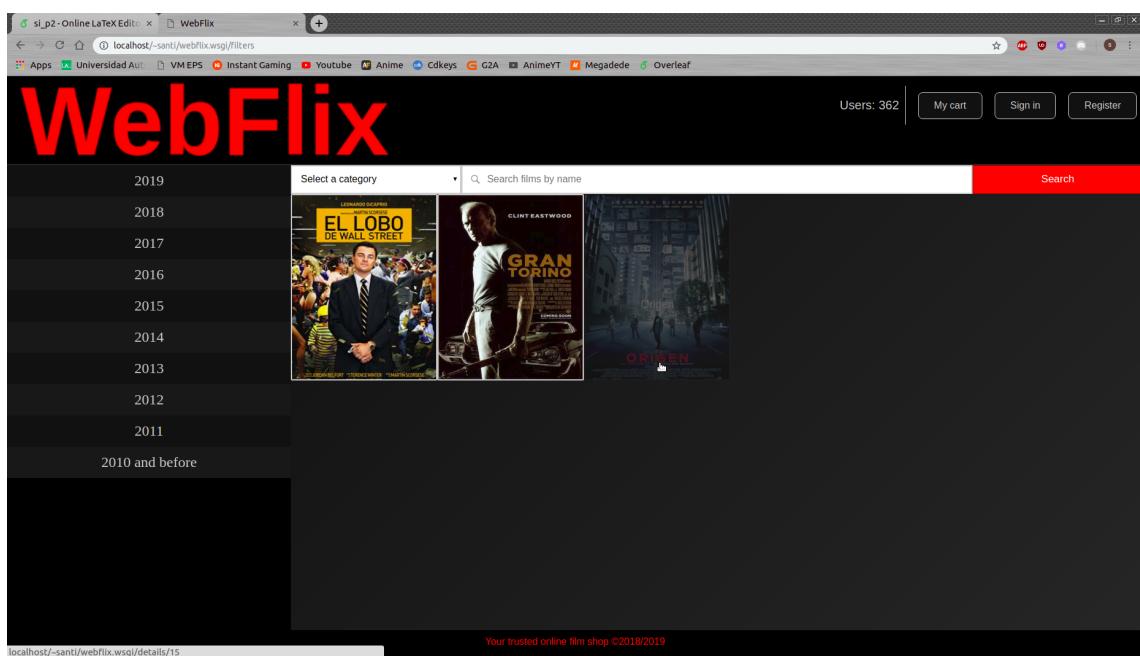


Figura 4: Filtrado por categoría 'drama'.

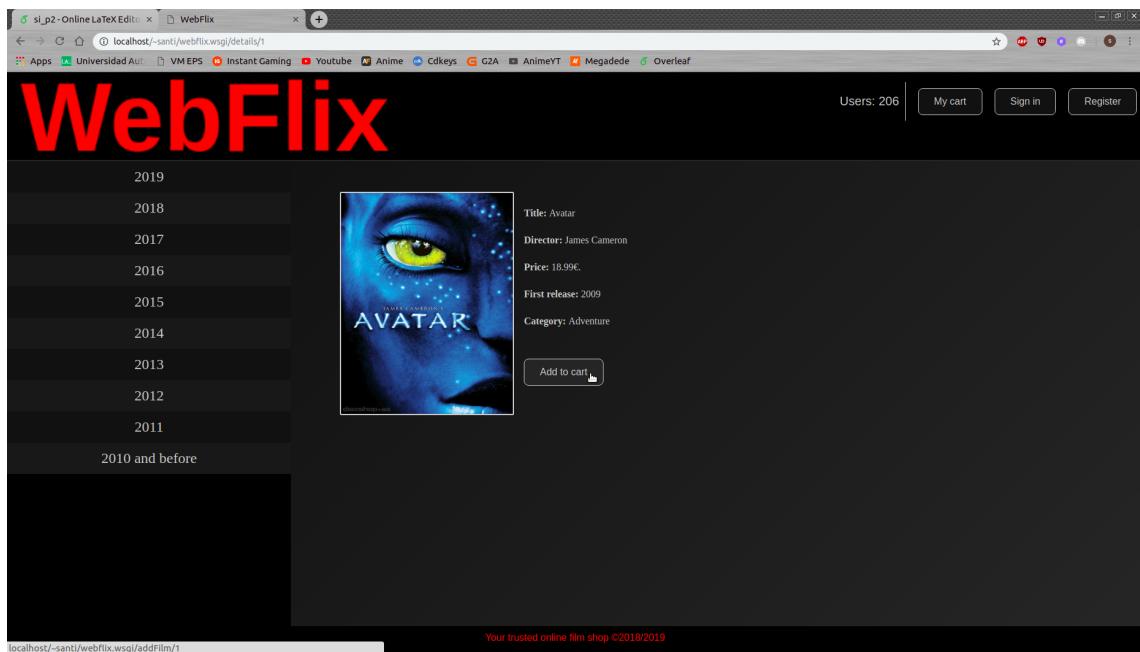


Figura 5: Detalles de película que no está en el carrito.

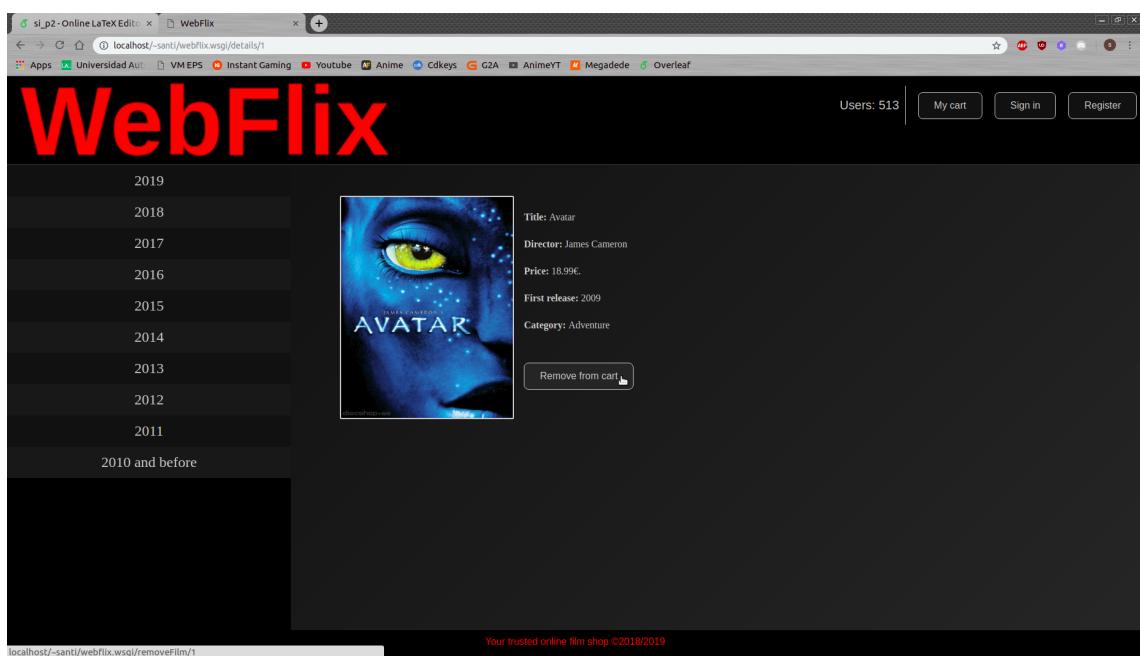


Figura 6: Detalles de película que está en el carrito.

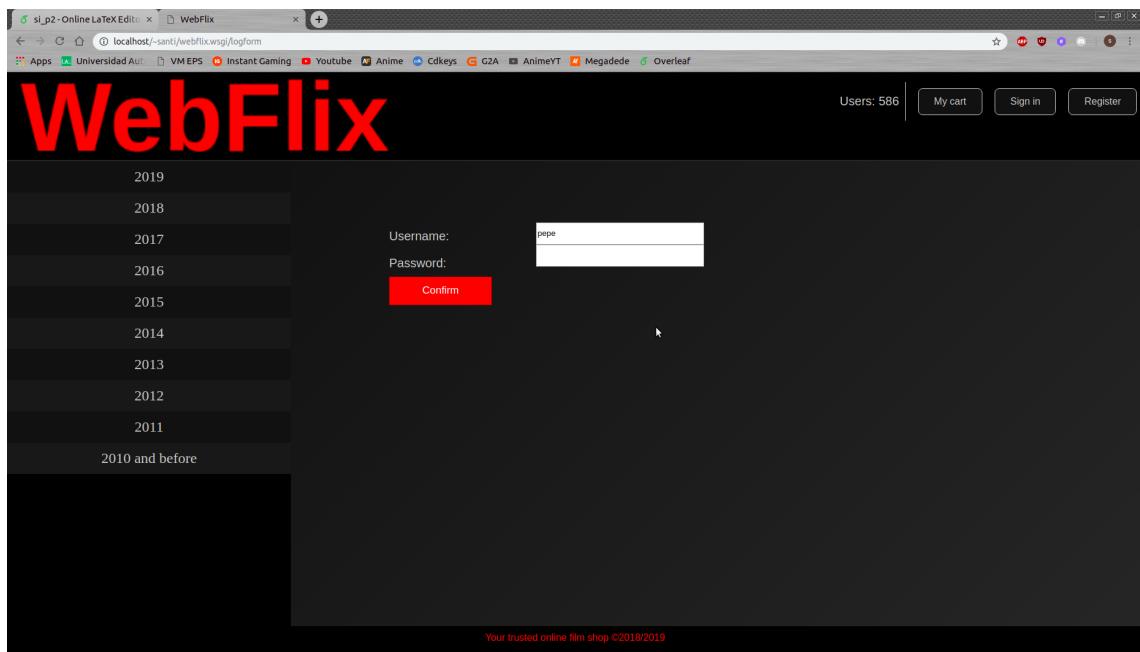


Figura 7: Log in con cookie.

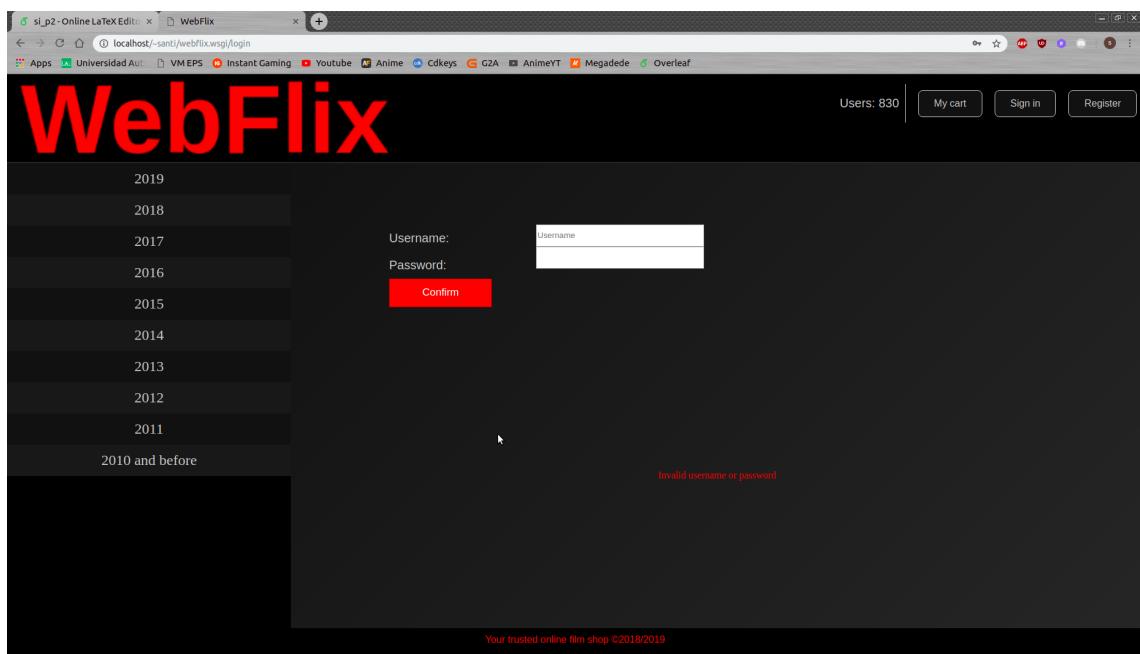


Figura 8: Log in con error.

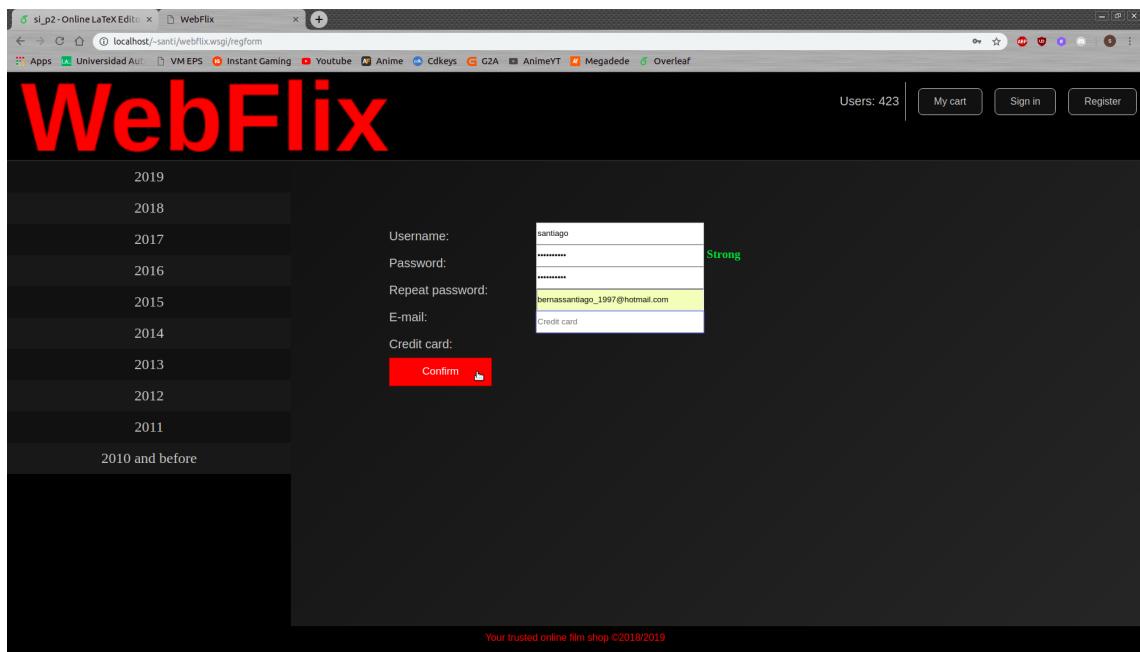


Figura 9: Strong password.

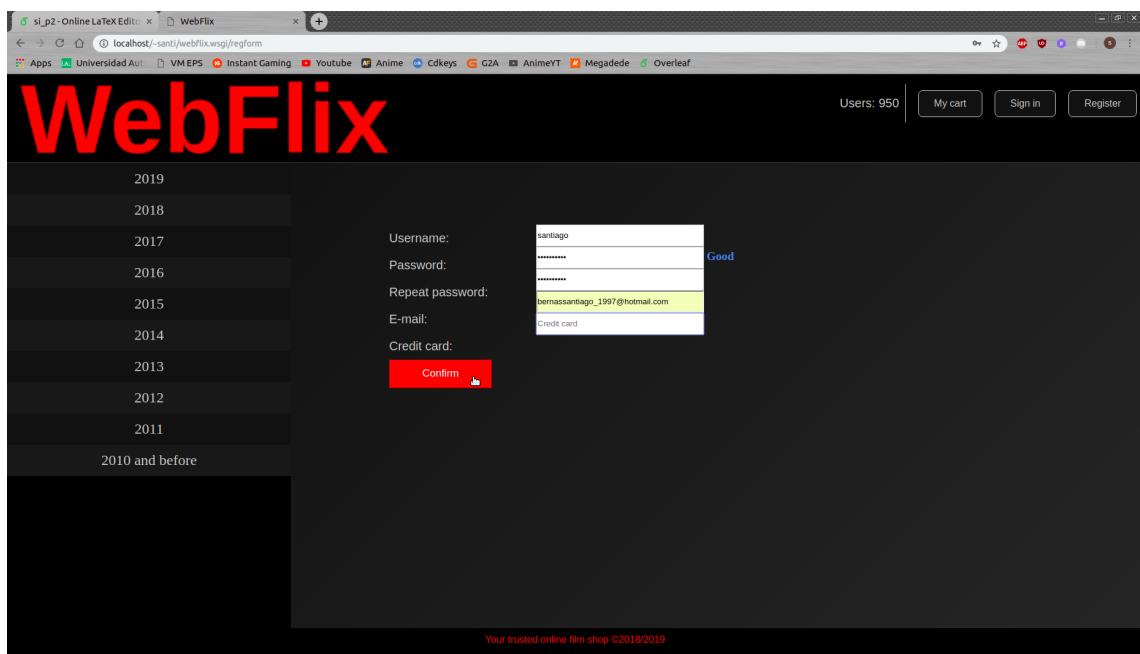


Figura 10: Good password.

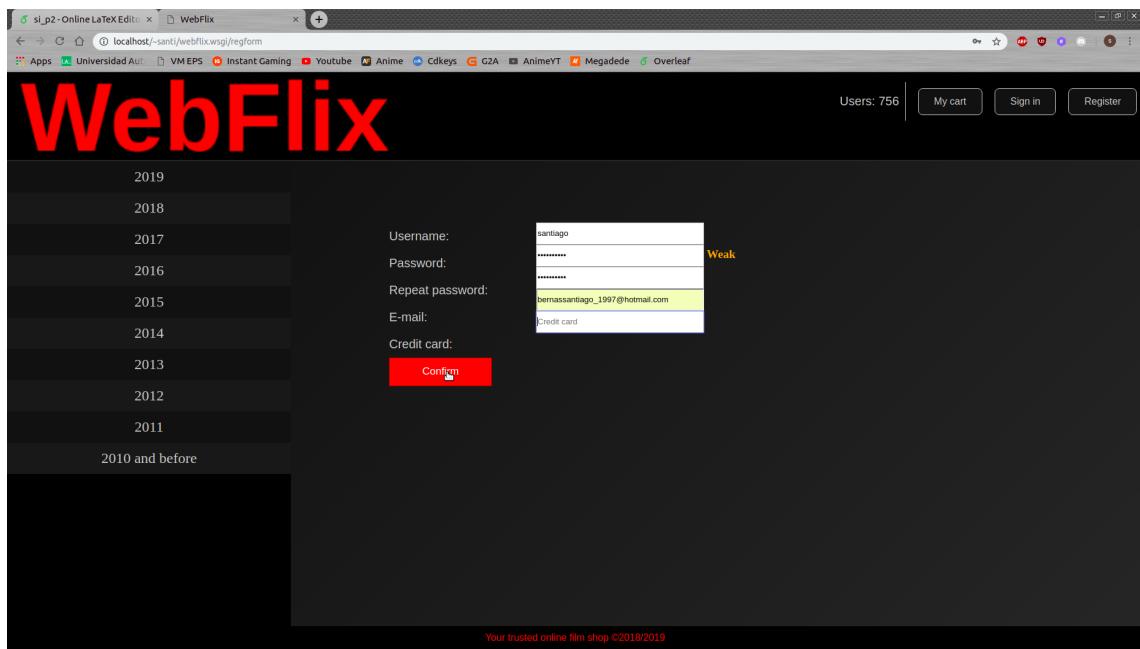


Figura 11: Weak password.

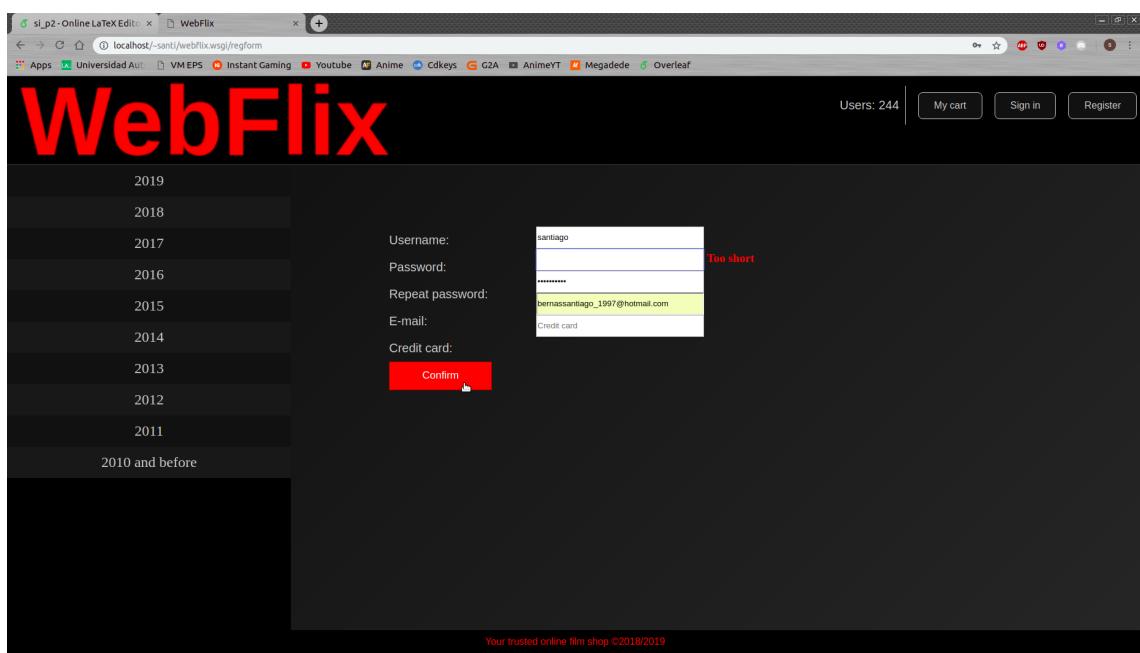


Figura 12: Too short password.

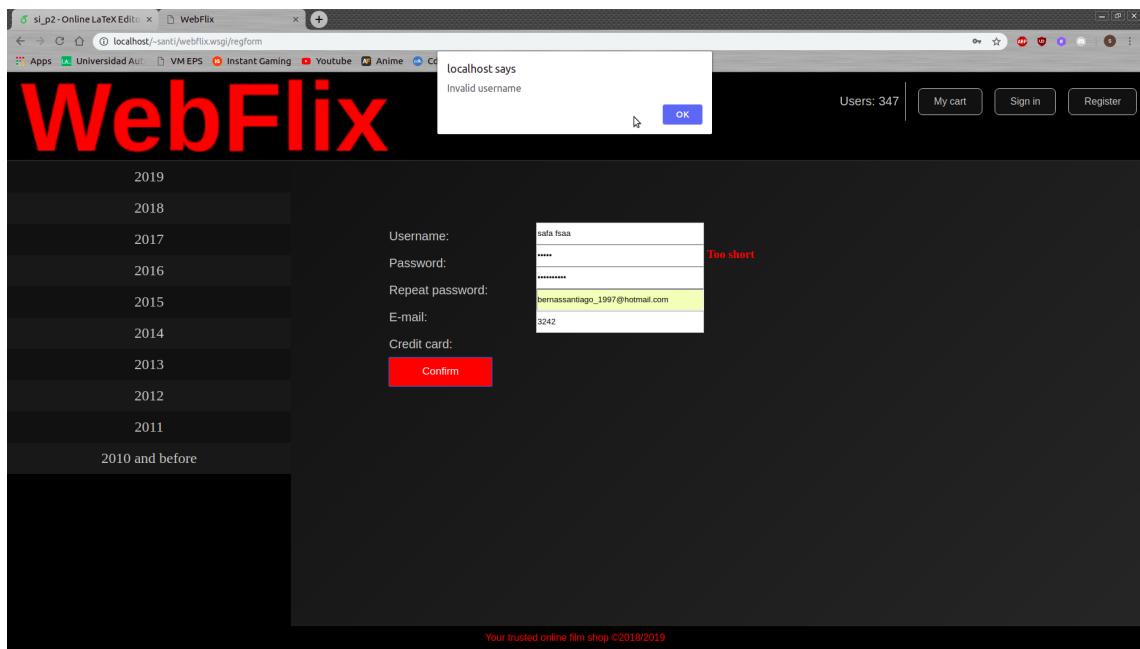


Figura 13: Invalid username.

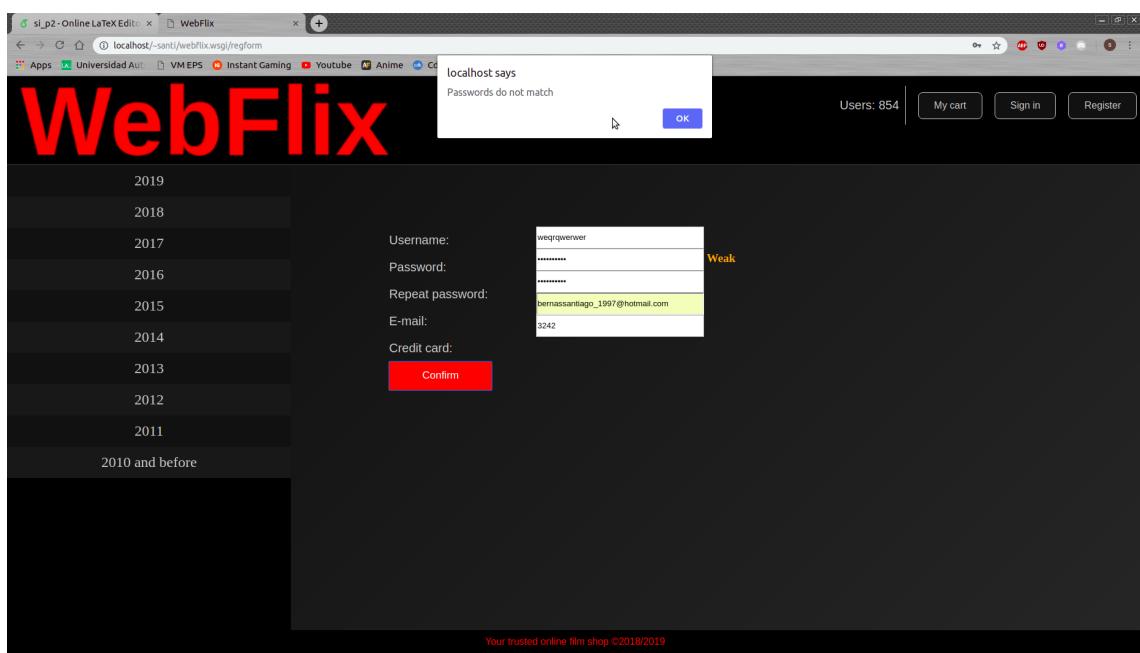


Figura 14: Passwords must match.

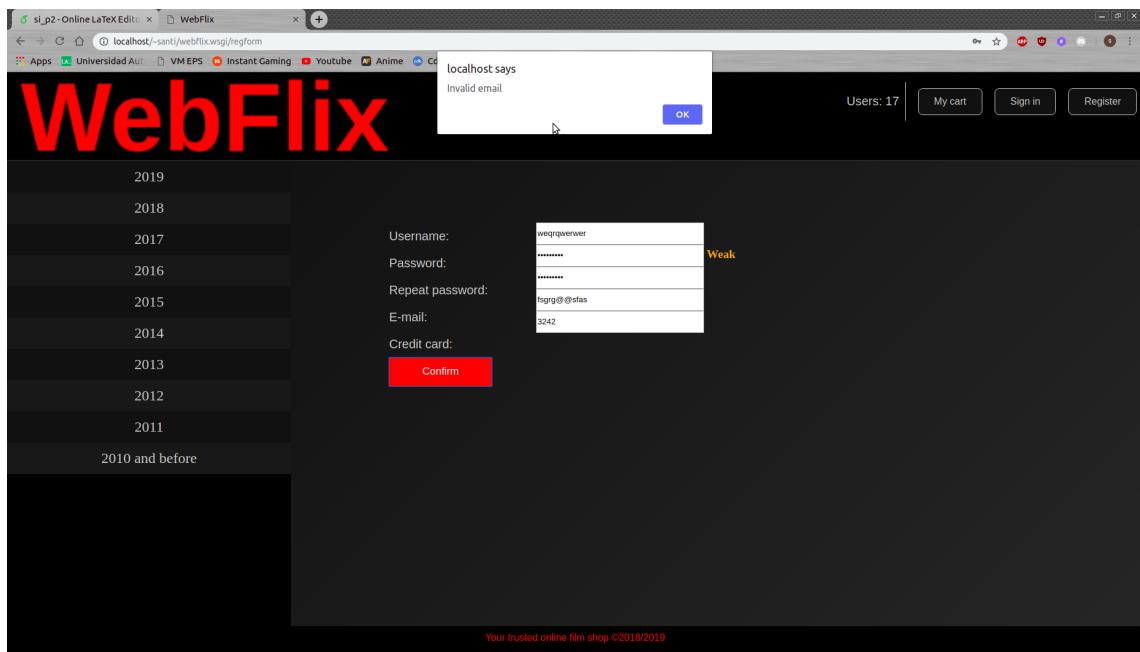


Figura 15: Invalid e-mail.

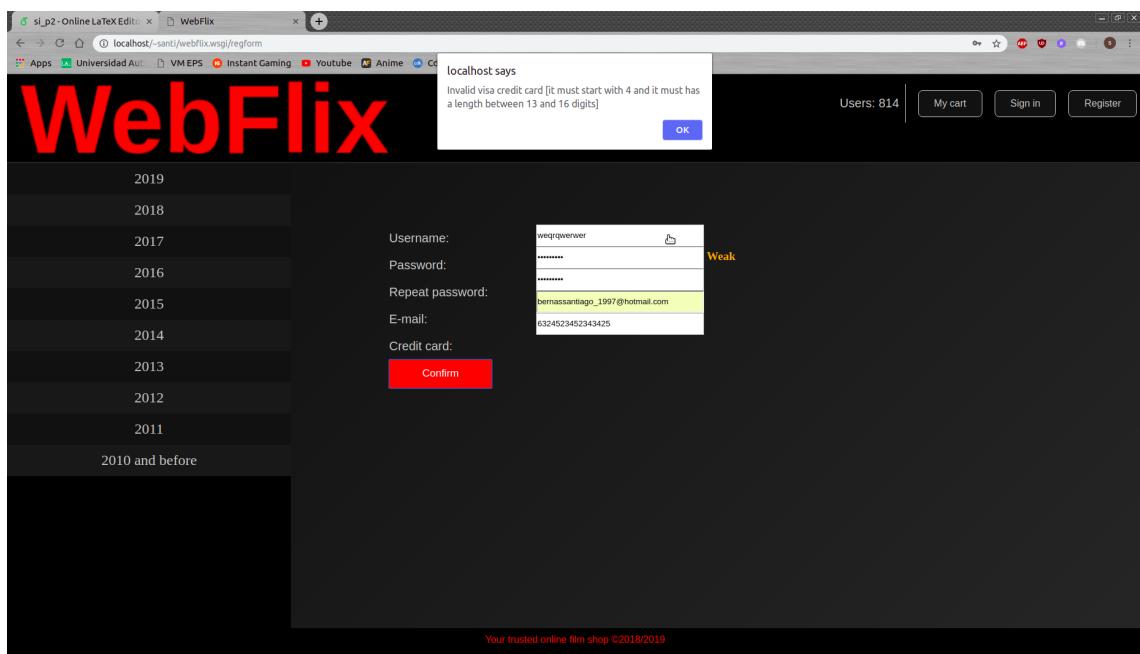


Figura 16: Invalid VISA credit card.

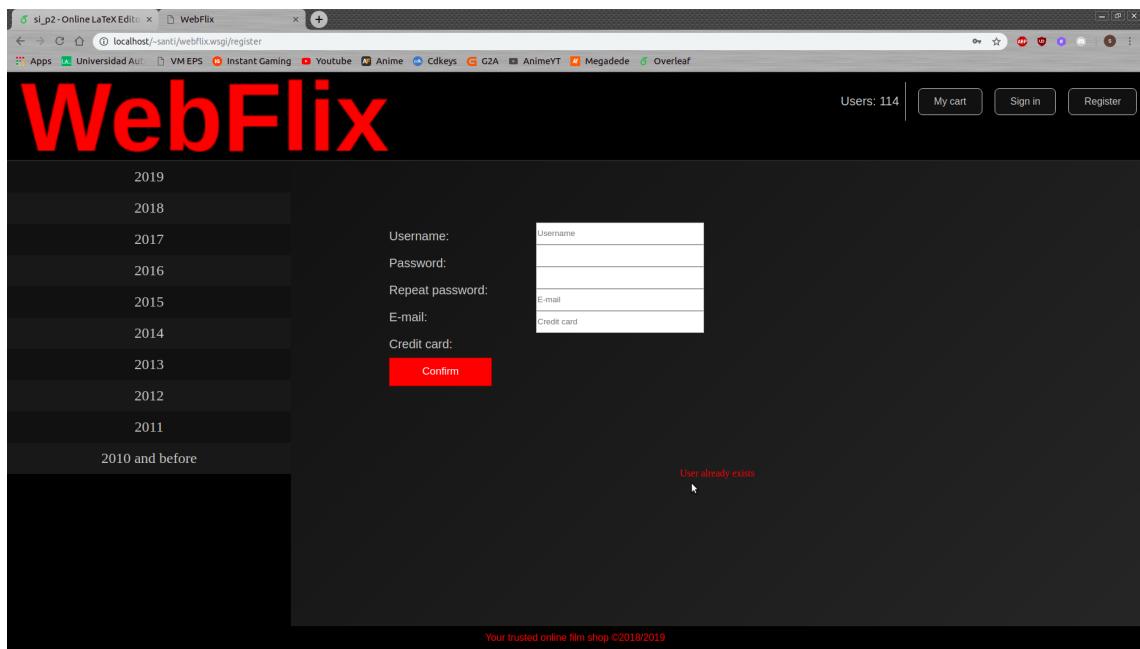


Figura 17: User already exists.

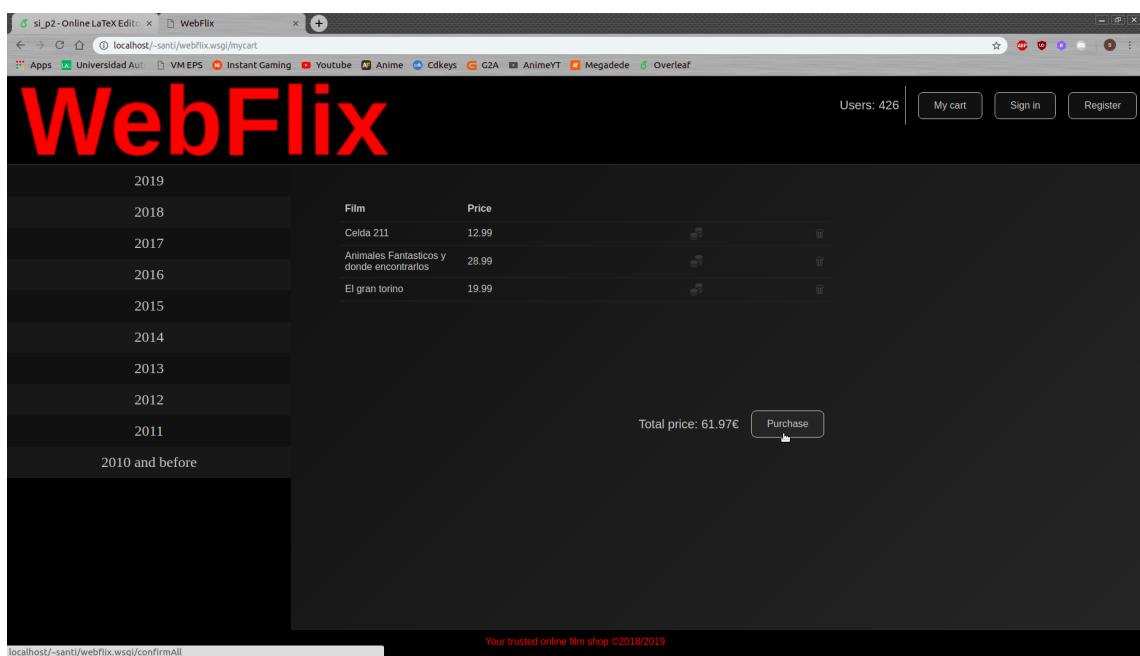


Figura 18: Carrito.

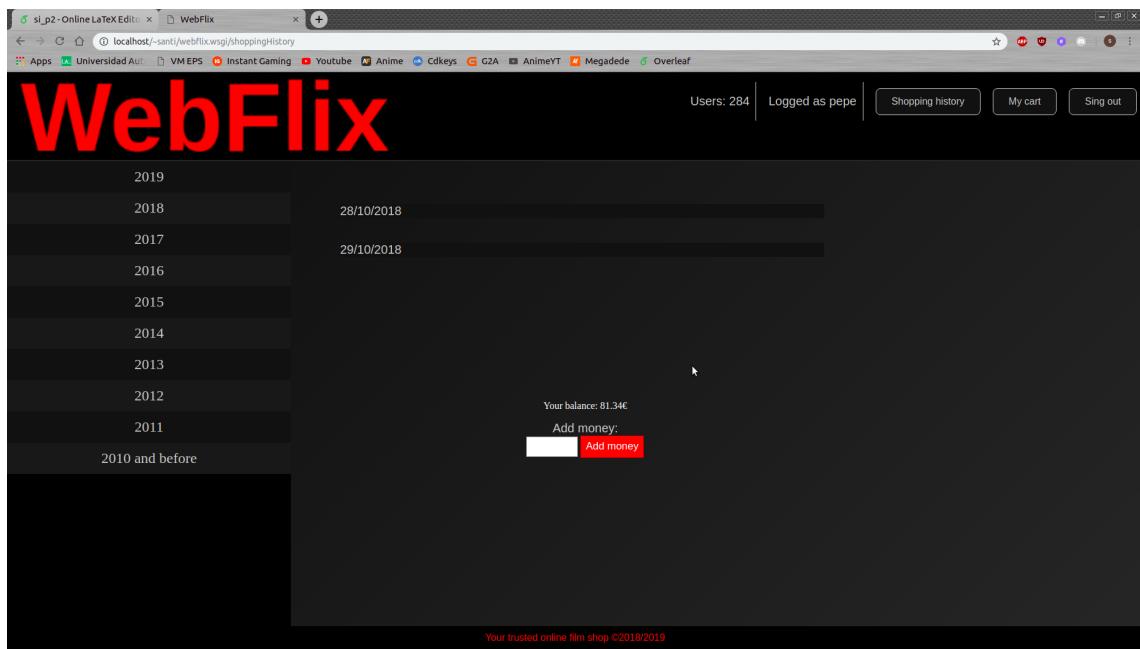


Figura 19: Historial de películas por defecto.

WebFlix	
2019	
2018	28/10/2018
2017	29/10/2018
2016	
2015	
2014	
2013	
2012	
2011	
2010 and before	
Film	
El caballero oscuro	14.39
Los increíbles 2	32.99
Celda 211	12.99
Dunkerque	24.99
Your balance: 81.34€	
Add money	

Figura 20: Desplegando una lista.

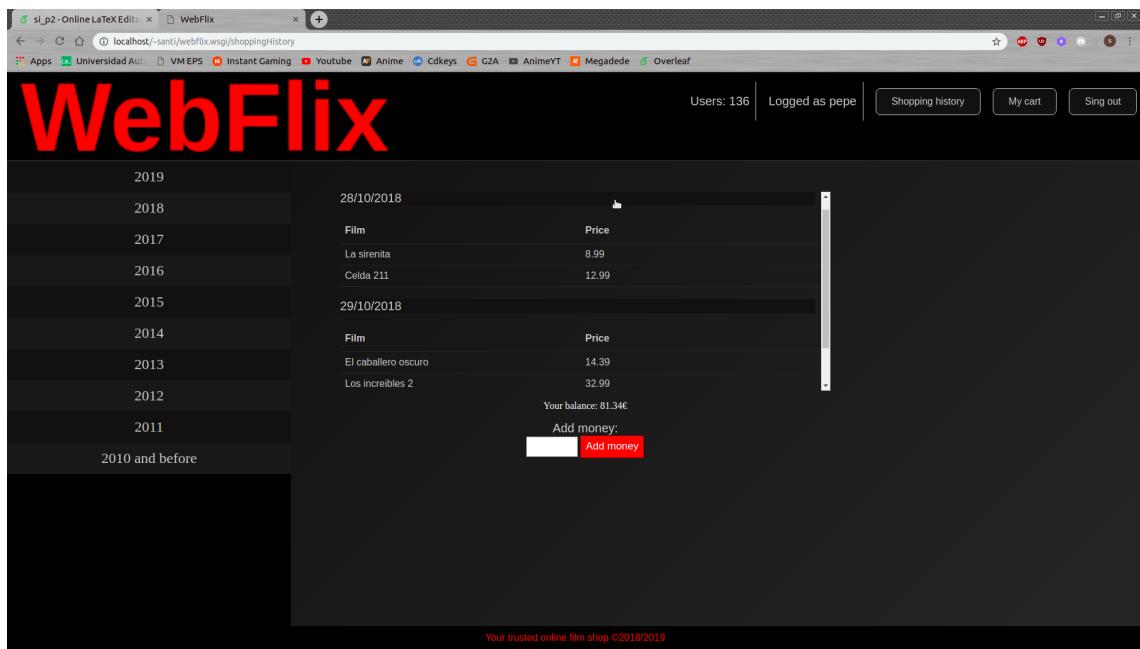


Figura 21: Desplegando las dos lista.

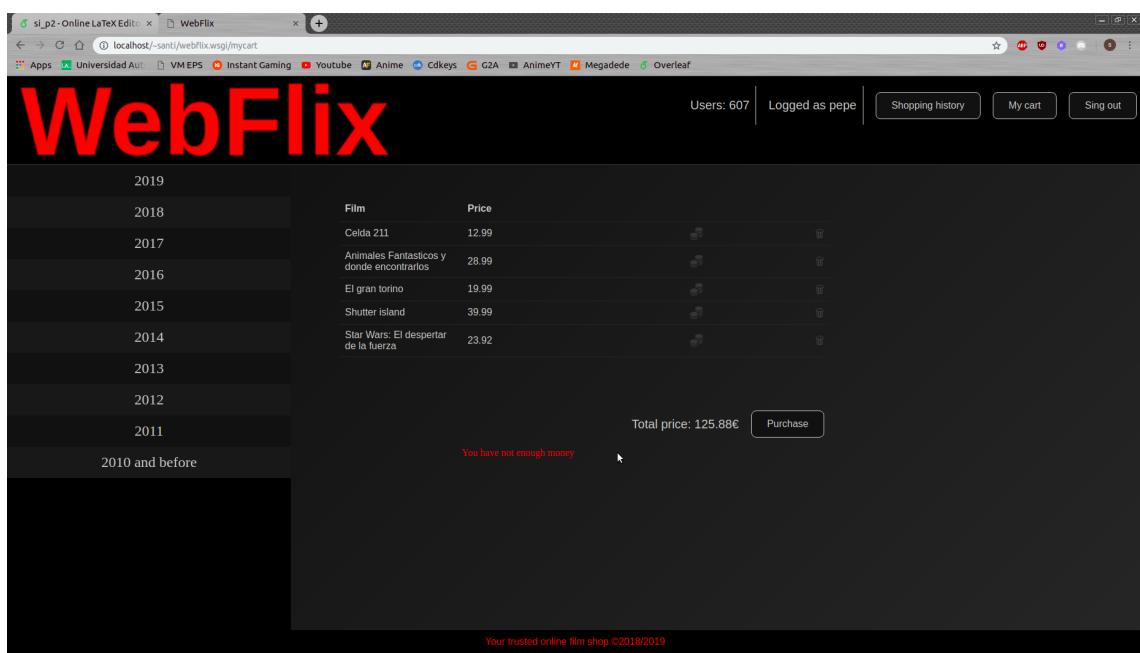


Figura 22: Dinero insuficiente.

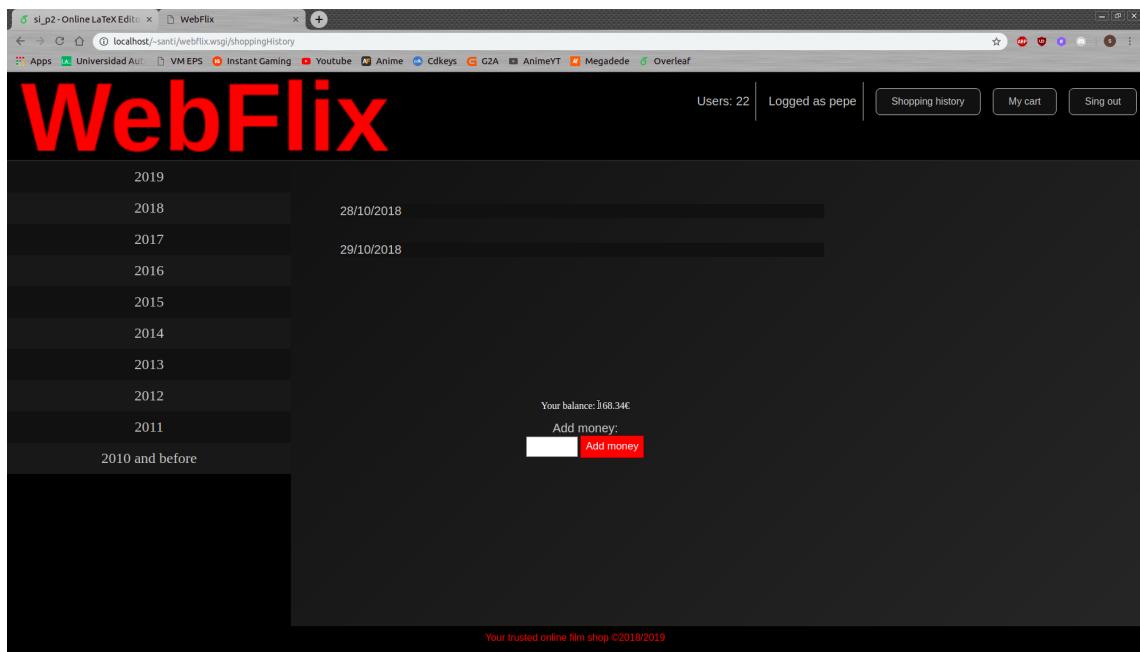


Figura 23: Añadiendo dinero.

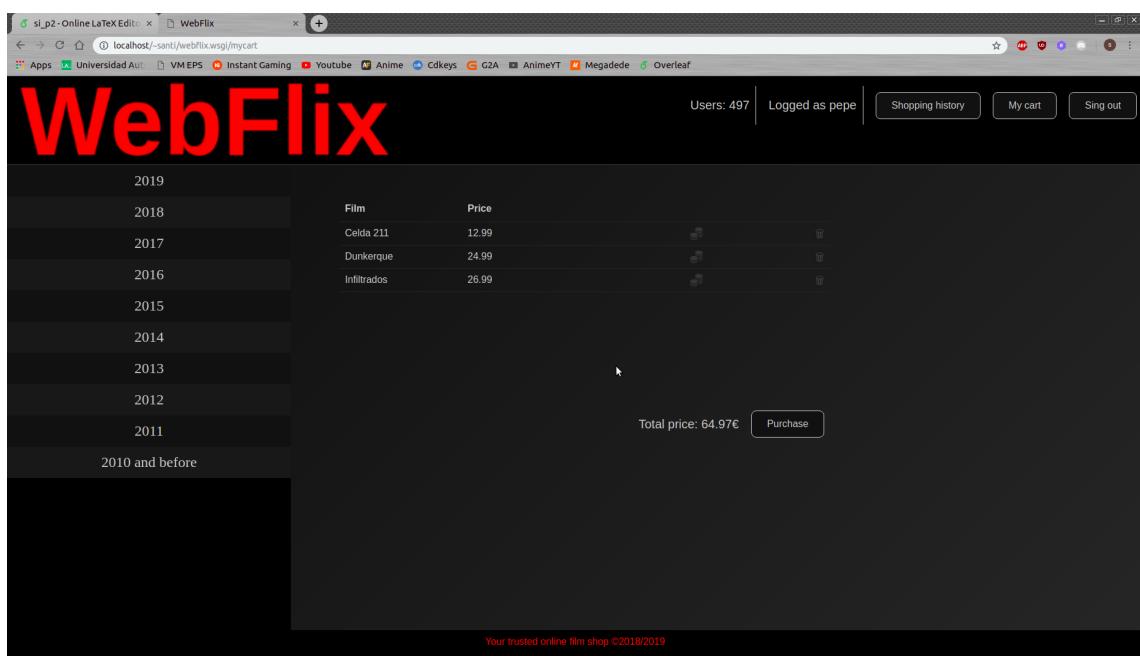


Figura 24: Carrito que vamos a comprar.

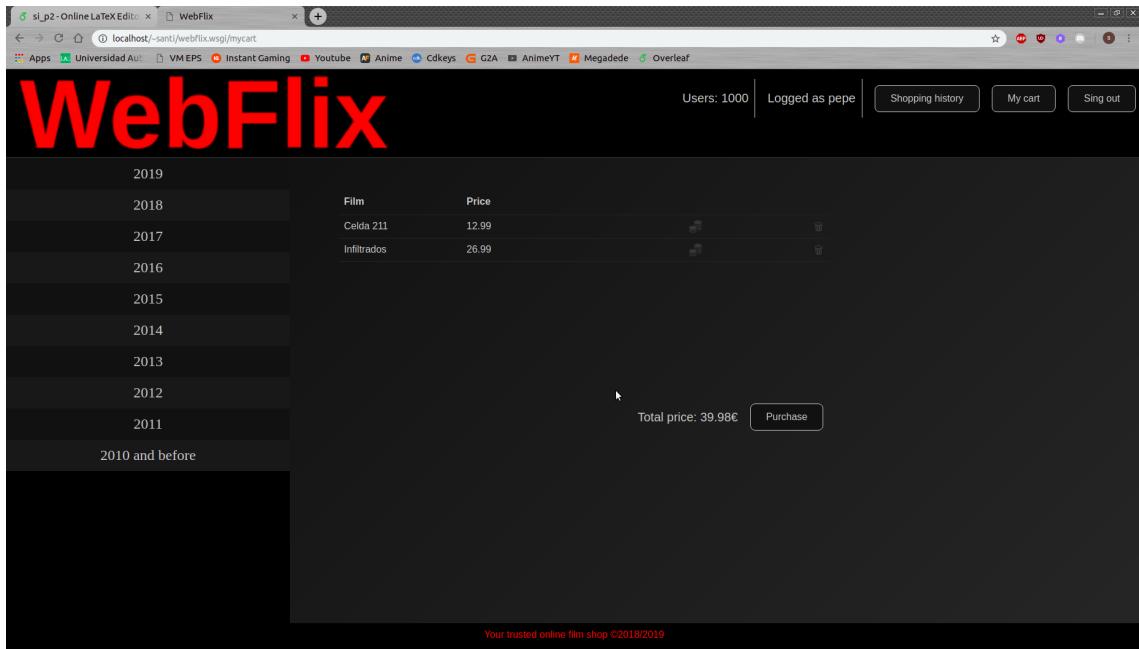


Figura 25: Compramos 1.

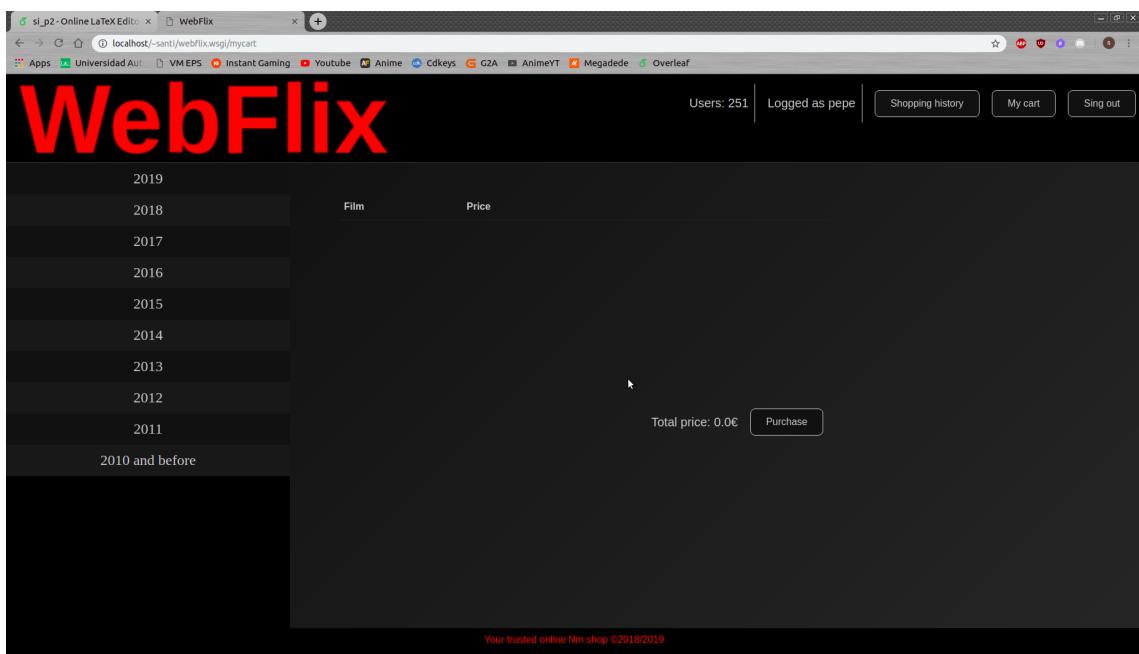


Figura 26: Compramos el resto.

The screenshot shows a web browser window for the 'WebFlix' application. At the top, there are tabs for 'Online LaTeX Editor' and 'WebFlix'. The address bar shows the URL: `localhost/~santi/webflix/wsgi/shoppingHistory`. The page header includes the 'WebFlix' logo, user information ('Users: 588 | Logged as pepe'), and navigation links for 'Shopping history', 'My cart', and 'Sign out'. On the left, a sidebar lists years from 2019 down to 2010 and before. The main content area displays a table of purchases for the year 2018:

Film	Price
El caballero oscuro	14.39
Los increíbles 2	32.99
Celda 211	12.99
Dunkerque	24.99
Dunkerque	24.99
Celda 211	12.99
Infiltrados	26.99

Below the table, it says 'Your balance: 103.37€' and has 'Add money:' buttons (white and red). At the bottom, it says 'Your trusted online film shop ©2018/2019'.

Figura 27: Historial actualizado.

The screenshot shows a web browser window for the 'WebFlix' application. The layout is similar to Figure 27, with the 'WebFlix' logo, user information ('Users: 259 | Logged as pepe'), and navigation links. The sidebar shows years from 2019 down to 2010 and before. The main content area displays a table of items in the cart:

Film	Price
Goldfinger	20.99
Interestelar	31.99

At the bottom, it says 'Total price: 52.98€' and has a 'Purchase' button. The footer says 'Your trusted online film shop ©2018/2019'.

Figura 28: Carrito a borrar.

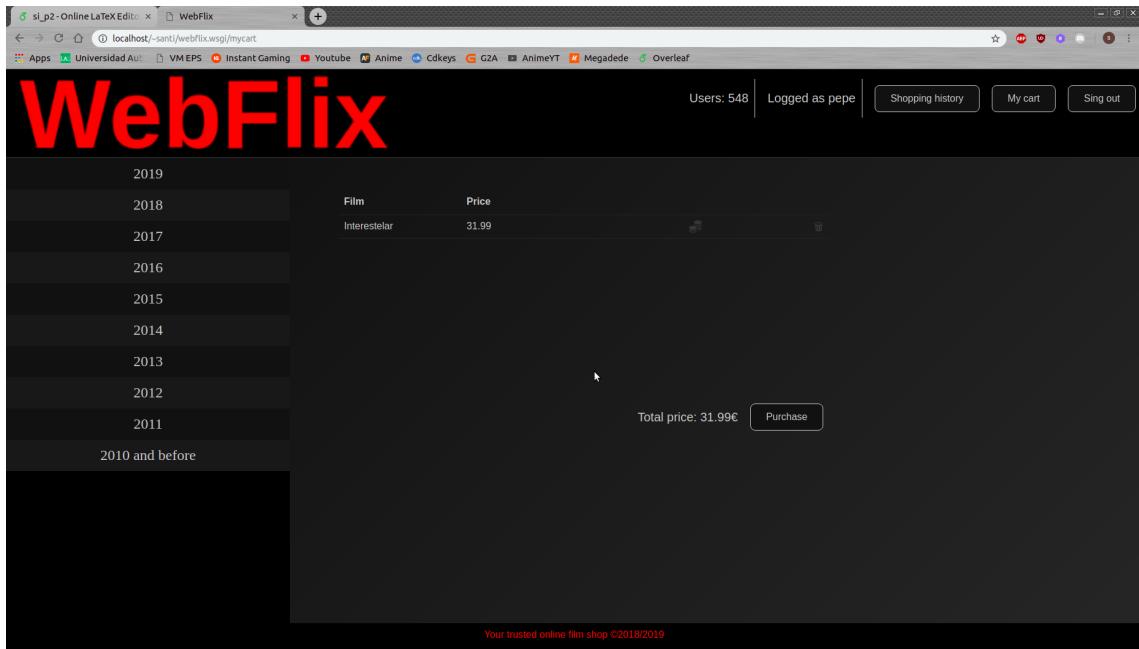


Figura 29: Tras eliminar desde el carrito.

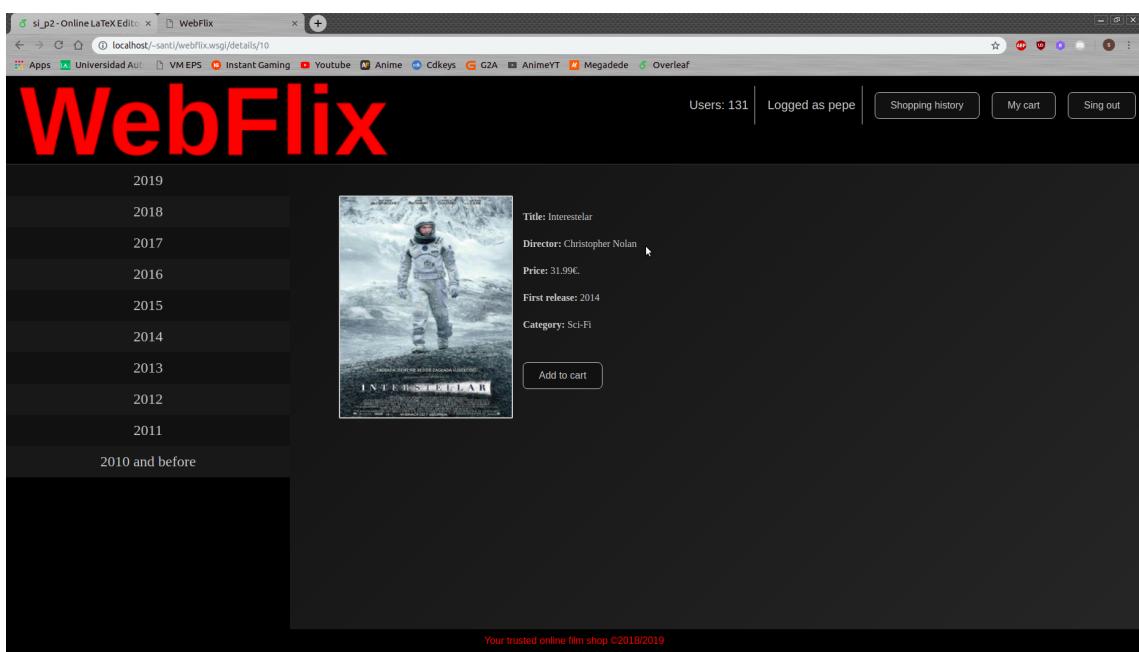


Figura 30: Tras eliminar desde detalles.

El resto de funcionalidades deberían ser capturadas en vídeo ya que dependen de flujos determinados. Además, el código de la práctica es bastante autoexplicativo en el resto de los casos.

5. Conclusiones

La mayor dificultad de esta práctica ha residido en buscar fuentes para implementar las funcionalidades pedidas en Internet. Gracias a ello, hemos aprendido

mucho Python (y en particular, mucho Flask), mucho sobre la generación dinámica de 'html', sobre 'css' y sobre todas las variedades de JavaScript (JavaScript, JQuery y AJAX). Y el resultado ha sido una aplicación fiable.