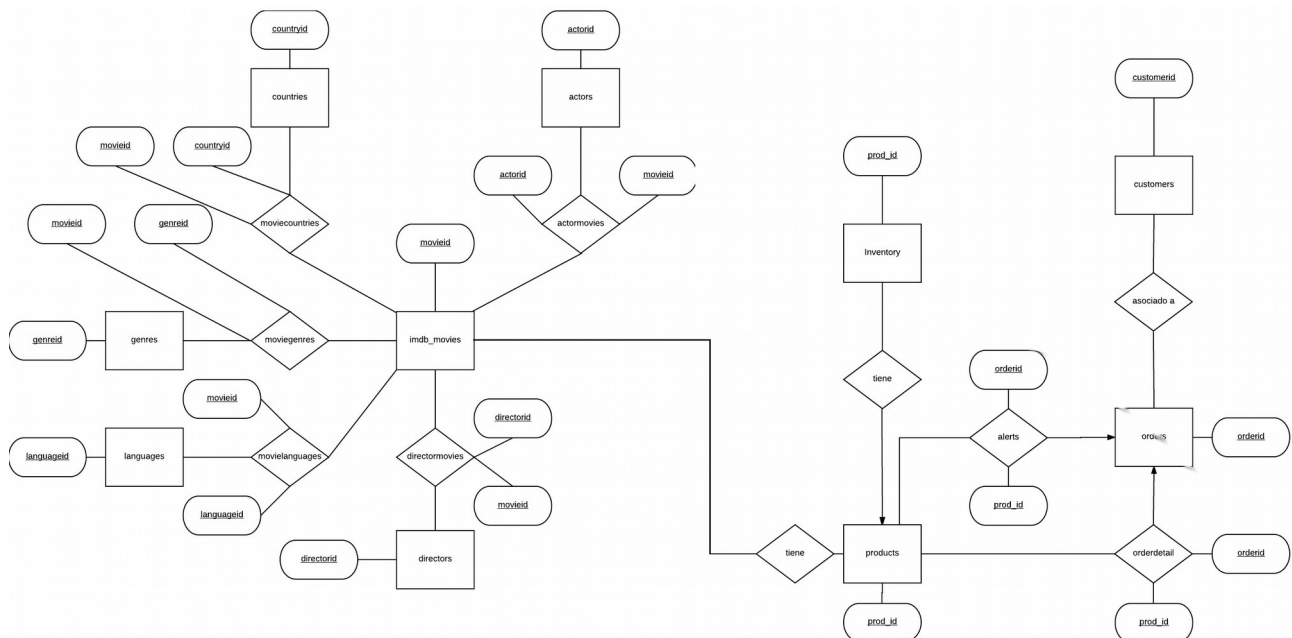


### Memoria 3 Prácticas de Sistemas Informáticos

actualiza.sql: (a) y (b)



Como vemos en el diagrama, sólo mostramos los atributos que son clave primaria, o que son más relevantes (esto es porque si contuviera todos los atributos existentes en la base de datos, el diagrama sería demasiado grande, y sería más difícil interpretarlo) y todo ello con la notación vista en teoría.

De cara a lo especificado en el enunciado, vamos a comentar los diferentes aspectos enumerados:

- i) los atributos clave primarias están subrayados
- ii) Las claves externas las hemos definido mediante relaciones entre entidades
- iii) De las tablas que existen, hemos identificado las entidades como rectángulos, las relaciones como rombos, y los atributos como los óvalos.
- iv) La cardinalidad queda denotada por los grafos, que si tienen una flecha en una dirección de la relación, indica que la entidad señalada tiene una correspondencia de 1 con respecto a la entidad en el otro extremo del grafo.
- v) Hemos eliminado todas las entidades débiles, haciendo que cada tabla de nuestra base de datos tenga su propia clave primarias.
- vi) También hemos eliminado los atributos multivaluados del apartado b para crear las relaciones y entidades correspondientes
- vii) existen los atributos derivados netamount y totalamount, que se calculan en el script setOrderAmount.sql
- viii) existe una participación total entre todas las tablas de la BBDD.

## Cambios principales del script

Para empezar hemos añadido todas las foreign key obvias que faltaban por definir, de atributos en ciertas tablas que hacen referencia a las primary keys de otras tablas (por ejemplo el orderid en orderdetail que hace referencia al orderid de orders, y otros casos similares).

También hemos quitado constraints NOT NULL de campos que no nos interesaban en customers, como el city o creditcardtype, que no usamos en nuestro register, y que quitado para que no nos pidan obligatoriamente que los rellenemos, o harían nuestro formulario de register muy largo. También hemos cifrado en md5 todas las contraseñas de esta tabla, para respetar los criterios de la práctica anterior.

Lo siguiente que hemos hecho es añadir primary keys a varias tablas, y para eso hemos tenido que eliminar filas que consideramos duplicadas si queremos usar dichas primary keys.

Las tablas de las que hemos eliminado filas duplicadas para poder crear primary keys son movielanguages y orderdetail.

También hemos actualizado las secuencias para que el valor actual sea el de el ultimo id de sus respectivas tablas, y de esa forma asigne automáticamente el siguiente id disponible a cada nueva fila.

Luego del apartado (b), para convertir cada atributo multivaluado (tablas movie "algo") en relaciones entre las tablas movies y unas nuevas tablas, hemos creado unas nuevas tablas languages, genres y countries, que siguen la misma modulación que hay por ejemplo entre movies, movieactors y actors.

Para el apartado (h) hemos creado la tabla de alerts, en la que, cuando el trigger updInventory detecte que no hay stock suficiente para una compra, se crea una fila en esta tabla, con el orderid (numero de pedido) y el prod\_id (numero de producto que ha dado error).

### setPrice.sql: (c)

En este apartado ejecutamos un simple UPDATE en el que dado el "price" actual en products, calculamos y rellenamos el "price" que debería tener en el pedido de la tabla "orderdetail", que fue un 2% menor cada año con respecto al actual de "price" de la tabla products.

	orderid [PK] integer	prod_id [PK] integer	price numeric	quantity integer
1	1	1014	10.96	1
2	1	1288	10.96	1
3	1	1938	10.96	1
4	2	2443	15.98	1
5	2	3229	13.98	1
6	3	268	17.98	1
7	3	696	15.98	1
8	3	1467	20.38	1
9	3	1766	17.98	1
10	3	3215	22.78	1
11	3	3777	12.98	1
12	3	3802	9.98	1
13	3	4256	12.98	1
14	3	4505	20.38	1
15	3	4794	9.98	1
16	4	701	28.46	1
17	4	2159	18.96	1
18	4	4713	17.96	1
19	4	4795	9.96	1
20	4	5989	17.96	1
21	4	6181	14.36	1
22	4	6233	15.96	1
23	4	6627	13.96	1
24	5	254	14.94	1
25	5	1826	14.94	1
26	5	2984	18.94	1

*Aquí podemos ver, cómo se han rellenado los campos price de la tabla orderdetail de forma correcta (antes de ejecutar el script, los campos price estaban en blanco)*

### setOrderAmount.sql: (d)

En este apartado hemos creado el procedimiento almacenado y después lo ejecutamos.

La lógica interna del procedimiento es muy simple:

Para cada pedido (orderid) sumamos todos los price\*quantity de cada uno de sus productos  
Y con la suma resultante, vamos rellenando los campos “netamount” de la tabla orders  
y posteriormente los campos totalamount (netamount + netamount\*tax%)

	orderid [PK] serial	orderdate date	customerid integer	netamount numeric	tax numeric	totalamount numeric	status character varying(10)
1	1	2015-07-22	693	32.88	15	37.8120	Shipped
2	2	2016-01-28	693	29.96	15	34.4540	Shipped
3	3	2016-09-11	693	161.40	18	190.4520	Paid
4	4	2015-08-19	693	137.58	15	158.2170	Shipped
5	5	2014-09-01	693	91.64	15	105.3860	Shipped
6	6	2014-03-05	693	64.86	15	74.5890	Shipped
7	7	2017-01-26	851	12	18	14.16	Shipped
8	8	2015-11-05	851	141.64	15	162.8860	Processed
9	9	2013-04-14	851	208.06	15	239.2690	Shipped
10	10	2012-07-25	851	56.5	15	64.975	Shipped
11	11	2012-08-30	851	93.6	15	107.640	Shipped
12	12	2012-06-18	851	72.5	15	83.375	Shipped
13	13	2014-05-11	851	148.16	15	170.3840	Shipped
14	14	2016-07-21	851	165.80	15	190.6700	Shipped
15	15	2017-03-10	851	11	18	12.98	Shipped
16	16	2016-08-10	851	109.04	15	125.3960	Shipped
17	17	2014-02-12	851	146.36	15	168.3140	Shipped
18	18	2014-09-05	851	158.40	15	182.1600	Shipped
19	19	2012-04-11	851	108.1	15	124.315	Shipped
20	20	2015-10-18	851	38.88	15	44.7120	Processed
21	21	2013-01-14	851	15.92	15	18.3080	Processed
22	22	2012-09-07	851	118.8	15	136.620	Shipped
23	23	2014-11-06	2959	95.08	15	109.3420	Processed
24	24	2013-09-11	2959	36.04	15	41.4460	Shipped
25	25	2017-03-02	2959	129.6	18	152.928	Shipped
26	26	2015-01-18	2959	151.64	15	174.3860	Processed

Aquí podemos ver como se han rellenado los campos netamount y totalamount de las filas de la tabla orders tras ejecutar el script (previamente vacías)

### getTopVentas.sql: (e)

En este procedimiento almacenado hacemos uso de varias subquerys dentro de otras para solventar el problema.

En q, la más interna, obtenemos años y productos y cantidad de veces que se vendio en ese año

En q2, obtenemos a partir de q y la tabla products, los años, películas correspondientes a los productos de q, y la cantidad de veces que se vendieron ese año

En q3, mediante q2, obtenemos el máximo de veces que se vendió una peli por año (sin la peli)

Y mediante q4 calculamos el q2 anterior

De esta forma, con un natural join entre q4 y q3, sacamos los años, el número máximo de pelis vendidas y la peli en cuestión asociada.

select * from getTopVentas(2011);				
Output pane				
Data Output	Explain	Messages	History	
	fecha double precision	id integer	titulo character varying	ventas numeric
1	2017	149475	Gang Related (1997)	57
2	2017	229931	Life with Mikey (1993)	57
3	2015	442893	Wizard of Oz, The (1939)	132
4	2012	281791	No Looking Back (1998)	101
5	2017	201944	Jerk, The (1979)	57
6	2016	229764	Life Less Ordinary, A (1997)	134
7	2014	189256	Illtown (1996)	142
8	2013	236107	Love and a .45 (1994)	136
9	2011	244463	Male and Female (1919)	9

OK.

Aquí podemos ver la consulta realizada, y el resultado consecuente.

### getTopMonths.sql: (f)

En este procedimiento almacenado hay 2 subqueries principales

En la primera, calculamos las fechas e importes acumulados por mes y año

En la segunda calculamos el numero de ventas en cada mes y año

Y al final simplemente vemos cuales han superado algun umbral de los 2 e imprimimos por pantalla las fechas (año y mes), el importe acumulado, y el numero de productos vendidos

```
SELECT * FROM getTopMonths(19000, 320000) order by fecha
```

	fecha text	money numeric	sales2 bigint
1	2011-11	15915.1030	117850
2	2011-12	46530.3340	361110
3	2012-01	72373.525	555583
4	2012-02	88431.205	690540
5	2012-03	135570.675	1045996
6	2012-04	146922.965	1143337
7	2012-05	188757.890	1444533
8	2012-06	205966.265	1592518
9	2012-07	232382.570	1789922
10	2012-08	276802.240	2135495
11	2012-09	291545.815	2241649
12	2012-10	337120.660	2615695
13	2012-11	333826.715	2577640
14	2012-12	328833.530	2540839
15	2013-01	351102.1760	2691345
16	2013-02	315967.6060	2437279
17	2013-03	342380.0010	2646275
18	2013-04	328885.6710	2518607
19	2013-05	339001.3870	2610539
20	2013-06	323588.1730	2594453
21	2013-07	349703.0450	2603379
22	2013-08	348071.2360	2673093
23	2013-09	355669.3090	2718624
24	2013-10	354726.7230	2718567

OK

Aqui tambien podemos ver la query hecha y el resultado consecuente. Los campos son el año y el mes, y el money (importe acumulado ese año y mes) y el sales2 (numero de ventas acumuladas ese año y mes), ambos los mostramos para ver que las filas que mostramos tras la consulta, son de meses donde se superan los umbrales prefijados en la consulta.

### updOrders.sql: (g)

Este trigger tiene 3 casos diferentes dependiendo del evento que lo active, que son cuando se añade un elemento al carrito, cuando se elimina, y cuando se modifica la cantidad.

Los 3 son esencialmente iguales, salvo por sus usos específicos de NEW y OLD

Lo que hacen esencialmente cada uno, es hacer un update del netamount y el total amount del carrito correspondiente al producto de orderdetail que se ha añadido/eliminado/modificado

Tienda de DVDs

Saldo: 936.87€

Cerrar Sesión pruebas Cesta


Seleccione Categoría

- Action
- Adult
- Adventure
- Animation
- Biography

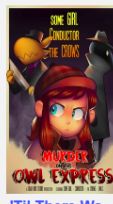
Cesta

Total:47.15 €

comprar



10 Things I H...



'Til There Wa...

Para mas informacion:  
javier.gomezmartinez@estudiante.uam.es  
carlos.1@estudiante.uam.es

Para mostrar la funcionalidad de este apartado, aprovechamos el codigo php hecho y mostramos que en la cesta imprimimos directamente el totalamount del pedido (que se actualiza con el trigger cada vez que añadimos un producto a la base de datos)

### updInventory.sql: (h)

este trigger se activa cuando en la tabla orders se modifica el status de una fila de NULL a algo distinto de NULL

Una vez se activa, lo que hace es formalizar la compra, es decir, recorrer la tabla de inventory asociada a cada producto en la cesta, y para cada uno de esos elementos reducirle el stock y aumentarle los sales.

Antes de actualizar los valores del stock y sales del inventory de un producto, comprueba que hay stock suficiente para formalizar la compra. Una vez comprueba que lo hay, ejecuta lo arriba descrito.

Si no hay stock suficiente, restaura el status de orders a NULL y crea una fila en la tabla alerts (orderid y prod\_id), que se comprueba en el php para ver si algo ha salido mal. Para que el usuario sea notificado, imprimimos un mensaje por pantalla pidiéndole que retire el producto de la cesta, pues no queda stock.

Además de lo arriba descrito, actualiza el orderdate de la cesta para marcar la fecha de compra.

### PHP (i), (j) y (k)

A parte de lo que detallamos abajo más específicamente como pide el enunciado, hemos actualizado varias funciones de utils.php para que ahora funcionen con la base de datos en lugar de con ficheros xml.

En general, nos hemos encargado de erradicar todo rastro de uso de ficheros xml y sustituido la funcionalidad aportada por estos con la misma funcionalidad pero utilizando base de datos en su lugar.

### register.php

Para este módulo, hemos sustituido el username como campo para identificar al usuario, por el email, puesto que en la base de datos que nos dan, es un campo unique más intuitivo para el usuario que el customerid (en contraposición con el username que no es único tal y como nos los dan).

Por tanto, el único cambio sobre la lógica principal que seguimos aquí sería el comprobar que el email insertado en el campo email del registro, sea uno que no existe ya en la base de datos.

Una vez comprobado que no existe, procederíamos a insertar los datos en sql.

Tienda de DVDs

Registrarse Login Cesta

## Register

Username

asdasd

E-mail

a@a.a

email ya existe

Contraseña

.....

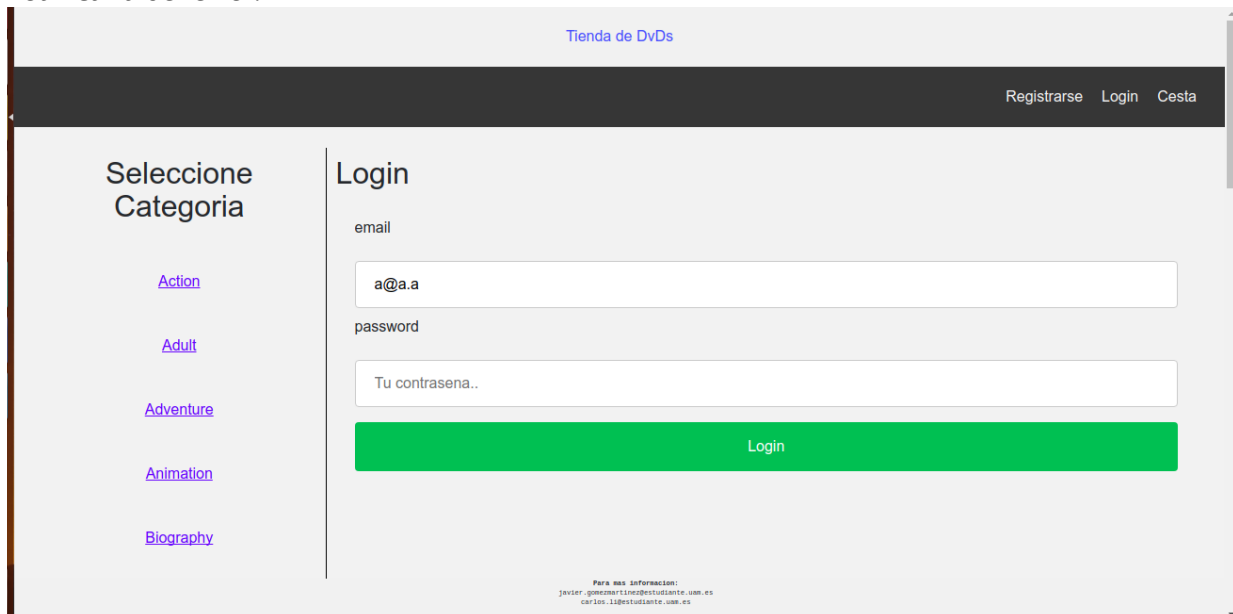
Repite tu contraseña

Para más información:  
javier.gomezmarlin@estudiante.uma.es  
carlos.liguestuante.uma.es

*Aquí podemos ver el register, que es esencialmente igual al de la práctica anterior salvo por el hecho de acceder a los usuarios existentes mediante postgres, y que ahora el campo unique con el que determinamos si un usuario existe, pasa a ser el email.*

## login.php

Los cambios principales de este módulo se encuentran principalmente en flogin, que es el módulo donde se ejecuta la funcionalidad principal, donde para hacer login simplemente ejecutamos una consulta viendo si existe un usuario con dicho email y contraseña a la vez, en caso contrario, notificaría del error.



The screenshot shows a web browser window with the title 'Tienda de DVDs'. The page has a dark header with links for 'Registrarse', 'Login', and 'Cesta'. On the left, there is a sidebar titled 'Seleccione Categoria' with links for 'Action', 'Adult', 'Adventure', 'Animation', and 'Biography'. The main content area is titled 'Login' and contains two input fields: 'email' with the value 'a@a.a' and 'password' with the placeholder 'Tu contraseña..'. Below the password field is a large green button labeled 'Login'. At the bottom of the page, there is a small footer with contact information: 'Para mas informacion: javier.gonzalez@estudiante.uam.es carlos.1@estudiante.uam.es'.

*Aqui podemos ver la pagina de login, que visualmente es igual, pero ahora todas las comprobaciones las hace mediante postgres, y no, mediante xml.*

## CESTA (y sus respectivos cambios generales)

Los cambios que afectan a la funcionalidad de la cesta se encuentran repartidos alrededor de varios módulos ya que es importante asegurar la integridad de la cesta de la compra.

La funcionalidad implementada permite a un usuario añadir productos a la cesta habiendo hecho o no Login. A la hora de realizar la compra, se le pide hacer Login y una vez hecha la compra, se debe reflejar en el historial del usuario.

Para empezar, continua existiendo la funcionalidad para añadir elementos a la cesta sin necesidad de haber hecho login, utilizando una variable de sesión.

Una vez hecho el login, estos datos se migran a la base de datos (orders y orderdetail) con cierto cuidado por detalles como que el carrito no tenga duplicados ejemplares iguales, etc. Todo de acuerdo a las especificaciones de la aplicación que decidimos para las prácticas anteriores. Este código se puede encontrar en flogin.php.

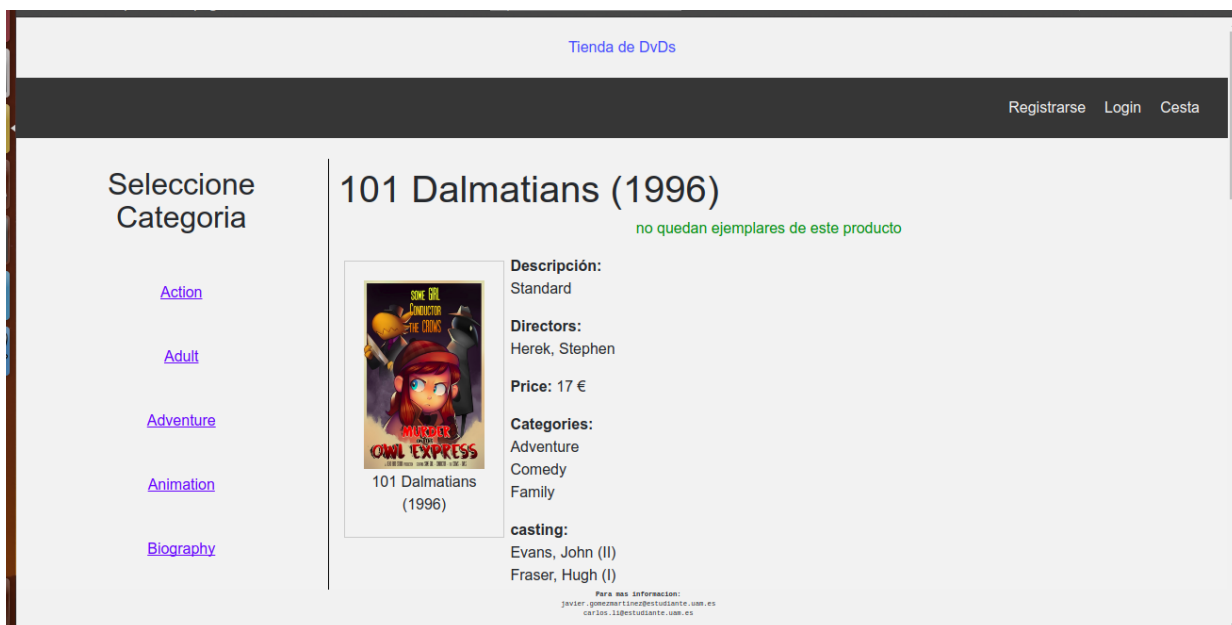
También es necesario hacer comprobaciones pertinentes en la página de cada producto a la hora de querer añadir un ejemplar a la cesta. Esto pertenece a product.php

A continuación, en la página de la cesta, se ha de mostrar el contenido de la cesta independientemente de si está almacenada en una variable de sesión (si no hizo Login) o en las tablas de la base de datos.

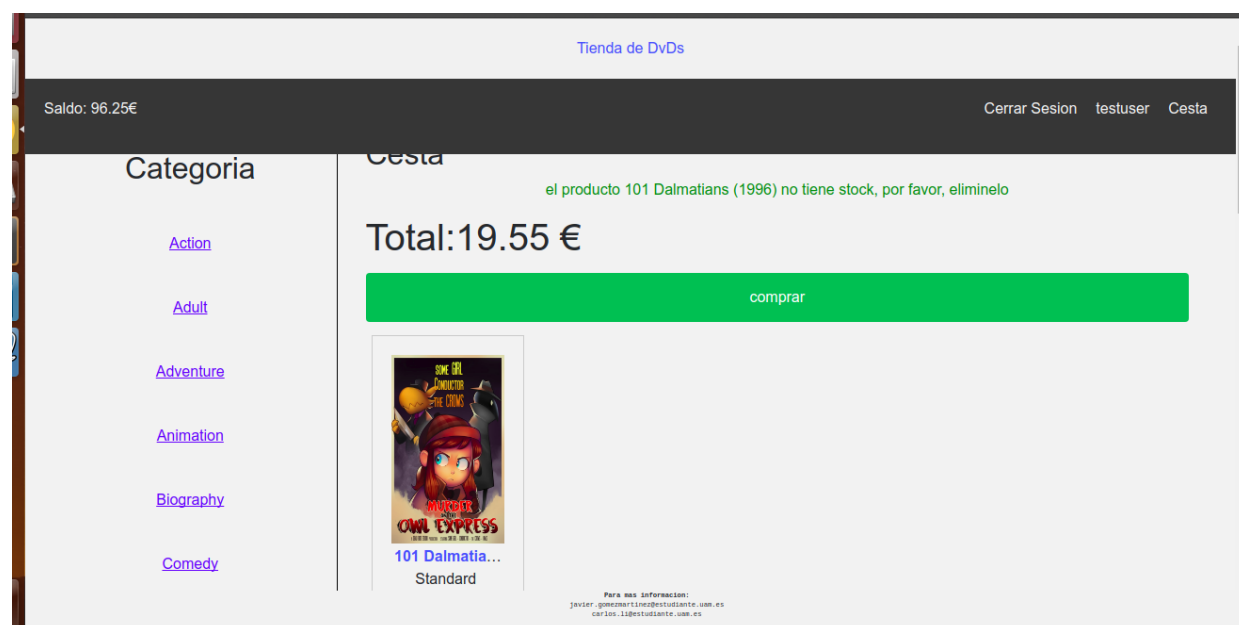
Además de esto, se implementa la funcionalidad de marcar un carrito como comprado, haciendo comprobaciones equivalentes a las de la práctica anterior (que el usuario tiene stock suficiente, que la cesta no esté vacía...) y algunas nuevas como que queden ejemplares de todos los productos a comprar (esta medida sirve de refuerzo a la antes comentada en la página de producto debido a que puede haber varios usuarios comprando los mismos productos a la vez y uno de ellos deje al otro sin ejemplares). Hacemos 2 comprobaciones, primero, si queda stock antes de añadir el producto al carrito, pues si no queda, se le imprime al usuario que no queda, y no se añade al carrito. De forma

complementaria, una vez añadido al carrito el producto, si ocurre como hemos dicho previamente (que otro usuario compre el ultimo stock de la tienda mientras nuestro usuario aún tiene el producto en su carrito), se activará el trigger updInventory, que no le avisará que no queda stock del producto en su carrito, y por tanto, le pedirá que retire el producto de su carrito antes de poder finalizar la compra.

Si la compra se efectúa correctamente, se actualiza el saldo consecuentemente y se verá reflejada en la página del historial (estos códigos se encuentran en cesta.php, fcesta.php y history.php)



En esta imagen tenemos el primer caso, cuando en la pagina de productos nos avisa que no quedan ejemplares, pues el stock está a 0 (obviamente, no dejamos que añada el producto a su carrito)



Aquí en cambio, tenemos el segundo caso, cuando el producto se añadió al carrito cuando aún quedaba stock, pero, antes de finalizar la compra, otro usuario agotó el último producto del stock, por tanto, lo que ocurre es que se le dice que lo elimine del carrito, o no puede finalizar la compra

## Capturas adicionales

Tienda de DVDs

Registrarse Login Cesta

Seleccione Categoría

[Action](#)  
[Adult](#)  
[Adventure](#)  
[Animation](#)  
[Biography](#)

Home

AÑO	PELICULA	VENTAS
2017	Gang Related (1997)	57
2017	Jerk, The (1979)	57
2017	Life with Mikey (1993)	57
2016	Life Less Ordinary, A (1997)	134
2014	Illtown (1996)	142
2015	Wizard of Oz, The (1939)	132

Search..

Para mas informacion:  
javier.gomezmartinez@estudiante.uam.es  
carlos.llorestadante.uam.es

Aquí tenemos el index, que contiene la tabla de top ventas, la barra de búsqueda, y las películas por defecto que vende (aún conserva la funcionalidad de la práctica anterior, siendo adaptada a postgres)

Tienda de DVDs

Saldo: 936.87€ Cerrar Sesión pruebas Cesta

[Adventure](#)  
[Animation](#)  
[Biography](#)  
[Comedy](#)  
[Crime](#)  
[Documentary](#)  
[Drama](#)  
[Family](#)

Historial:

2017-11-29(mostrar menos)

**'Breaker' Mo...**  
Standard  
Beresford, Br...  
13 €

**'Crocodile' D...**  
Standard  
Cornell, John (I)  
15 €

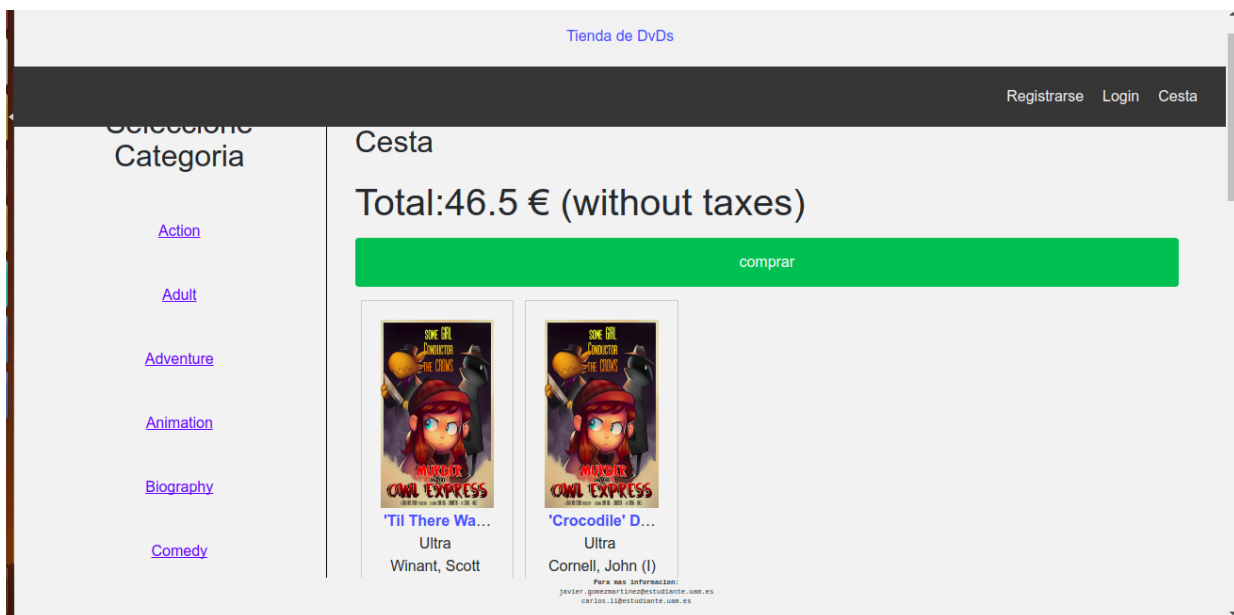
**'Til There Wa...**  
Gold  
Winant, Scott  
19.2 €

**Catwalk (1995)**  
Standard  
Spencer, Hen...  
19 €

Para mas informacion:  
javier.gomezmartinez@estudiante.uam.es  
carlos.llorestadante.uam.es

Aquí tenemos el historial tras comprar de la cesta con un usuario llamado pruebas, cargado desde la base de datos.





Aquí tenemos la cesta creada sin haber hecho login, en el caso en el que intentara comprar los productos, mandaría al usuario a la pantalla de login para validar su membresía. En cuanto consiga hacer login correctamente, se intentaría combinar la cesta creada sin haber hecho login, con una cesta que pudiera haber existido previamente en la cuenta del usuario, y que no hubiera sido terminada. (Aquí contamos con la premisa de que en esta tienda virtual, sólo vendemos una copia de cada película por usuario, pues es una tienda de películas digitales).