

Práctica 3:

Sistemas

Informáticos

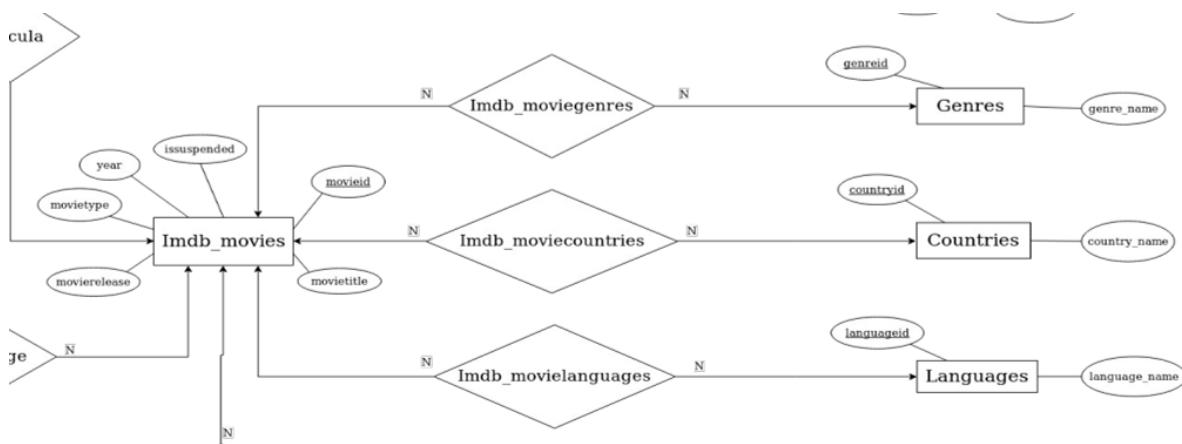
Autores: Daniel Mateo, Aitor Melero

Diseño:

El diagrama entidad relación de la aplicación se encuentra en el fichero “ER.pdf”, por comodidad.

En el diagrama hemos concluido que customers (los clientes), orders (los pedidos), products (los productos), inventory (donde se guarda el stock), imdb_movies (las películas), imdb_directorsmovies (los directores), imdb_actormovies (los actores) son entidades. Por otro lado, orderdetail es una relación entre los productos y los pedidos.

Una parte destacable, modificada del esquema original proporcionado en el material de la práctica, es la siguiente parte del diagrama:



Hemos sacado los géneros, los países, y los idiomas de sus respectivas tablas imdb en entidades aparte como se ve en el dibujo. Transformamos las tablas imdb en relaciones entre la entidad imdb_movies y genres, countries y languages. Todo esto lo hemos hecho porque lo pide el enunciado. Aquí van las creaciones de dichas tablas en el “actualiza.sql”:

Tabla genres

```
-- TABLA GENRES
CREATE TABLE genres (
  genreid SERIAL NOT NULL,
  genre_name VARCHAR(32) NOT NULL,
  PRIMARY KEY(genreid)
);
```

Tabla countries

```
-- TABLA COUNTRIES
CREATE TABLE countries (
  countryid SERIAL NOT NULL,
  country_name VARCHAR(32) NOT NULL,
  PRIMARY KEY (countryid)
);
```

Tabla languages

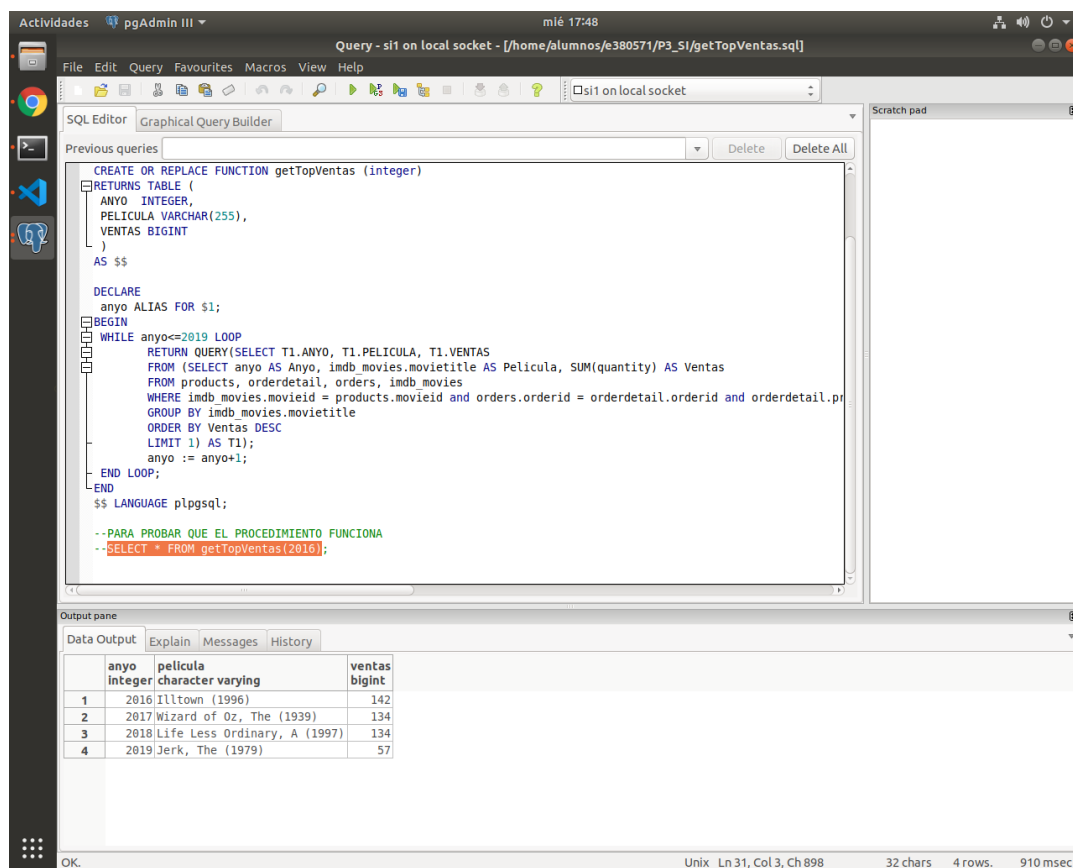
```
-- TABLA LANGUAGES
CREATE TABLE languages (
  languageid SERIAL NOT NULL,
  language_name VARCHAR(32) NOT NULL,
  PRIMARY KEY (languageid)
);
```

También hemos creado tablas auxiliares para poder llegar a las tablas que se acaban de mostrar. Se pueden ver todos los detalles de la implementación del actualiza en el propio fichero “actualiza.sql”. En dicho fichero aparecen unos pocos comentarios explicando las creaciones de tablas e inserciones que hemos ido realizando para conseguir el resultado final, el del diagrama entidad-relación. Cabe destacar que la tabla “errores” no está incluida en el diagrama porque la parte de errores se implementó más adelante debido a uno de los triggers pedidos, pero iría relacionada a la tabla de productos.

Consultas, procedimientos y triggers:

Para todas las consultas, procedimientos y triggers hemos cumplido los requisitos del enunciado y por ello hay poco que explicar. Todo lo usado en los ficheros correspondientes viene explicado brevemente en los comentarios de los propios archivos.

Si hay que destacar que los triggers los empezamos realizando con bucles dentro del propio trigger pero al final, por eficiencia, acabamos llamando desde los triggers a un procedimiento (el que se encarga de realizar lo especificado en cada archivo) repetidamente.



The screenshot shows the pgAdmin III interface. The SQL Editor window displays a PL/pgSQL function named `getTopVentas` that takes an integer parameter and returns a table with columns `anyo` (integer), `pelicula` (varchar(255)), and `ventas` (bigint). The function uses a loop to iterate from the input year down to 2019, querying the `products`, `orderdetail`, and `orders` tables to find the top movie for each year. The output pane shows the results of the function call `getTopVentas(2019)`.

anyo	pelicula	ventas
2019	Jerk, The (1979)	57
2018	Life Less Ordinary, A (1997)	134
2017	Wizard of Oz, The (1939)	134
2016	Il'town (1996)	142

Resultado de prueba del procedimiento `getTopVentas`.

Actividades pgAdmin III mié 17:47

Query - si1 on local socket - [/home/alumnos/e380571/P3_SI/getTopMonths.sql]

File Edit Query Favourites Macros View Help

SQL Editor Graphical Query Builder

Previous queries Delete Delete All

```

--RETURNS TABLE (
--  ANYO DOUBLE PRECISION,
--  MES DOUBLE PRECISION,
--  IMPORT INTEGER,
--  PRODUCTOS INTEGER
--)
--AS $$

DECLARE
  product ALIAS FOR $1;
  totalimport ALIAS FOR $2;
BEGIN
  return query(
    SELECT anno, MONTH, CAST(importe AS INTEGER) AS importe, CAST(quantity AS INTEGER) AS quantity
    FROM (SELECT EXTRACT(year FROM orderdate) AS anno, EXTRACT(month FROM orderdate) AS month, SUM(totalamou
    FROM orders
    NATURAL JOIN orderdetail
    GROUP BY anno, month
    ORDER BY anno, month) AS UWU
    WHERE quantity >= totalimport OR importe >= product
  );
END
$$ LANGUAGE plpgsql;

--PARA PROBAR EL BUEN FUNCIONAMIENTO DEL PROCEDIMIENTO
--SELECT * FROM getTopMonths(19000,320000);

```

Scratch pad

Output pane

Data Output Explain Messages History

	anyo double precision	mes double precision	import integer	productos integer
1	2013	11	102931	875
2	2013	12	305477	2614
3	2014	1	463395	3881
4	2014	2	551341	4629
5	2014	3	866353	7262
6	2014	4	937308	7897
7	2014	5	1191607	10146
8	2014	6	1316052	11078
9	2014	7	1482002	12547

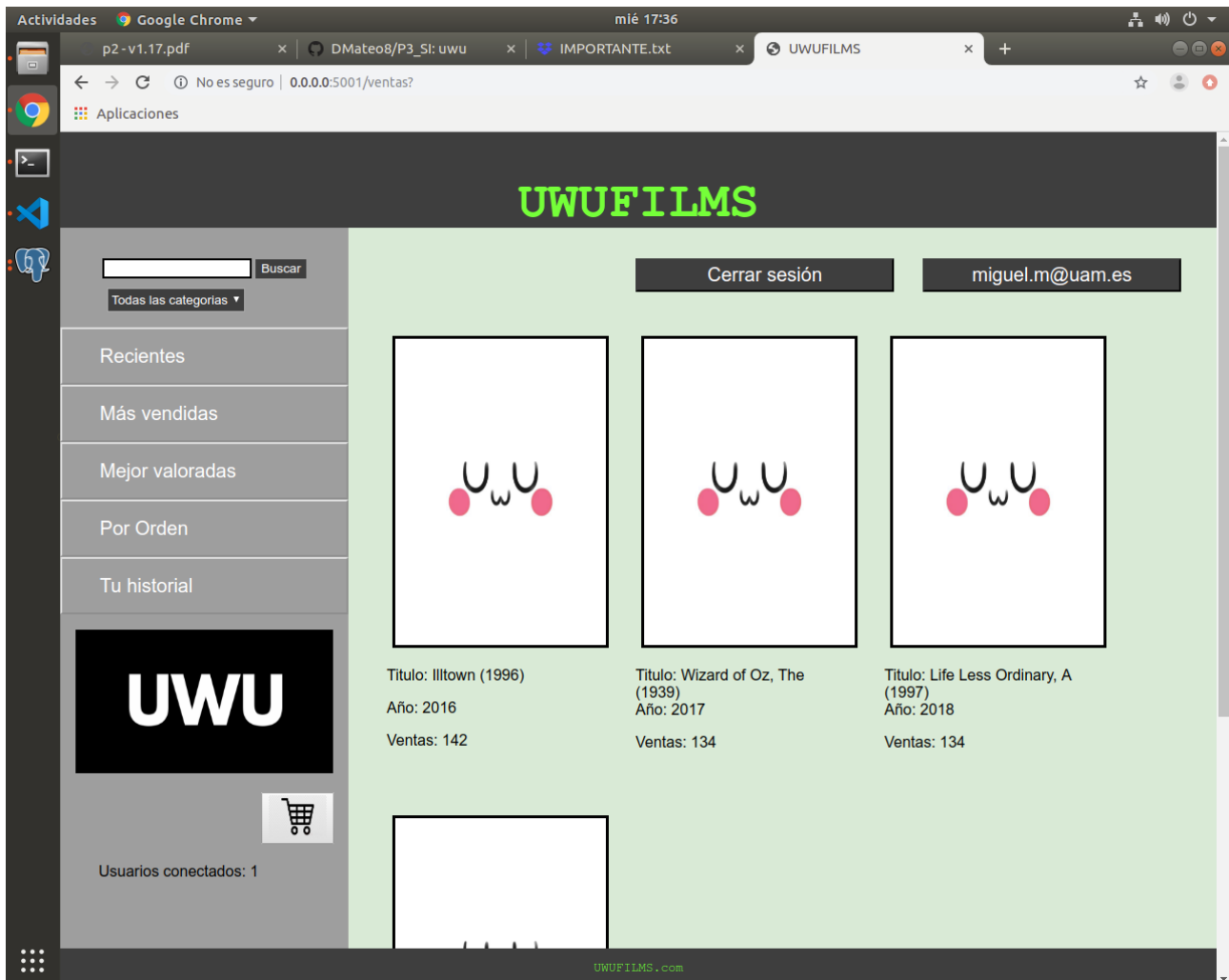
OK. Unix Ln 31, Col 3, Ch 926 40 chars 72 rows. 777 msec

Ejemplo de prueba del procedimiento para el getTopMonths.

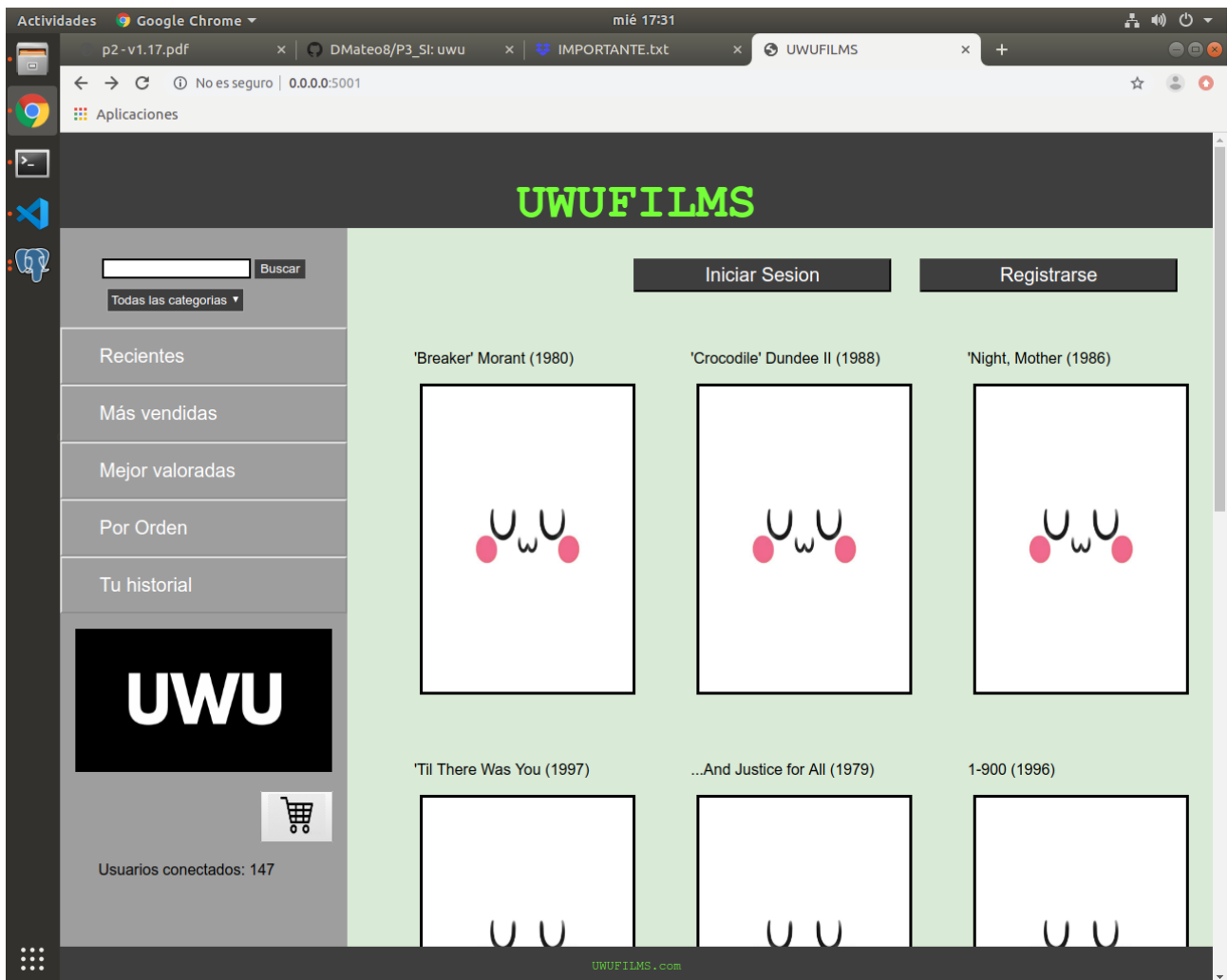
NOTA: El tiempo de ejecución de cada fichero es pequeño, salvo el del setPrice que tarda unos 10 segundos. Hemos intentado mejorar el tiempo de ejecución pero ha sido imposible. Por último, el orden de ejecución del código SQL en la base de datos es: actualiza.sql, setPrice.sql, setOrderAmount.sql, getTopVentas.sql, getTopMonths.sql, updOrders.sql e updInventory.sql.

Parte de la funcionalidad del código:

Comenzamos hablando del index. Aunque se nos pide mostrar las películas más vendidas desde el 2016 (getTopVentas) en el propio index, nosotros teníamos un botón ya preparado desde la práctica 1 y, por lo tanto, hemos usado dicho botón (más vendidas) para mostrar estos resultados. Realmente este contenido se renderiza en el index.



Ejemplo de las películas más vendidas.



Ejemplo del index nada más iniciar la aplicación.

En cuanto al registro e inicio de sesión, hemos borrado las carpetas de usuario y accedido a la base de datos mediante funciones de database.py. Hemos mantenido el template de la práctica anterior. Respecto al login, hemos decidido que se utiliza el email como nombre ya que es la columna que no tiene valores repetidos (username no era única, hablando de customers).

Actividades Google Chrome mié 17:32

p2-v1.17.pdf x DMateo8/P3_Sl: uwu x IMPORTANTE.txt x UWUFILMS

No es seguro | 0.0.0.0:5001/registro?

Aplicaciones

UWUFILMS

Todas las categorías ▾


Recientes


Más vendidas

Mejor valoradas

Por Orden

Tu historial





Usuarios conectados: 2622

Registro

Nombre de usuario:

Contraseña:
Fortaleza: Segura :)

Contraseña(confirmar):

Email:

Tarjeta de credito:

UWUFILMS.com

Ejemplo de registro.

Actividades Google Chrome mié 17:33

p2-v1.17.pdf x DMateo8/P3_Sl: uwu x IMPORTANTE.txt x UWUFILMS

No es seguro | 0.0.0.0:5001/crear_usuario

Aplicaciones

UWUFILMS

Todas las categorías ▾


Recientes


Más vendidas

Mejor valoradas

Por Orden

Tu historial





Usuarios conectados: 371

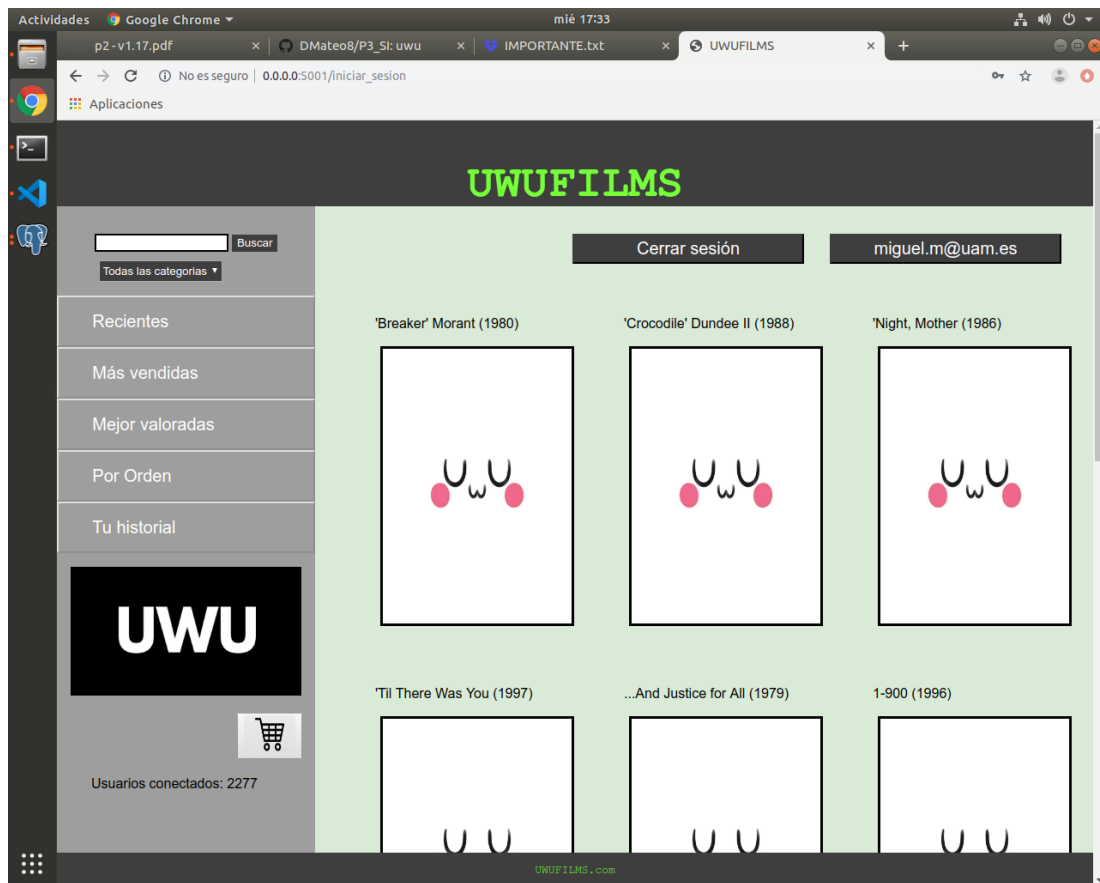
Inicia sesión

Nombre de usuario:

Contraseña:

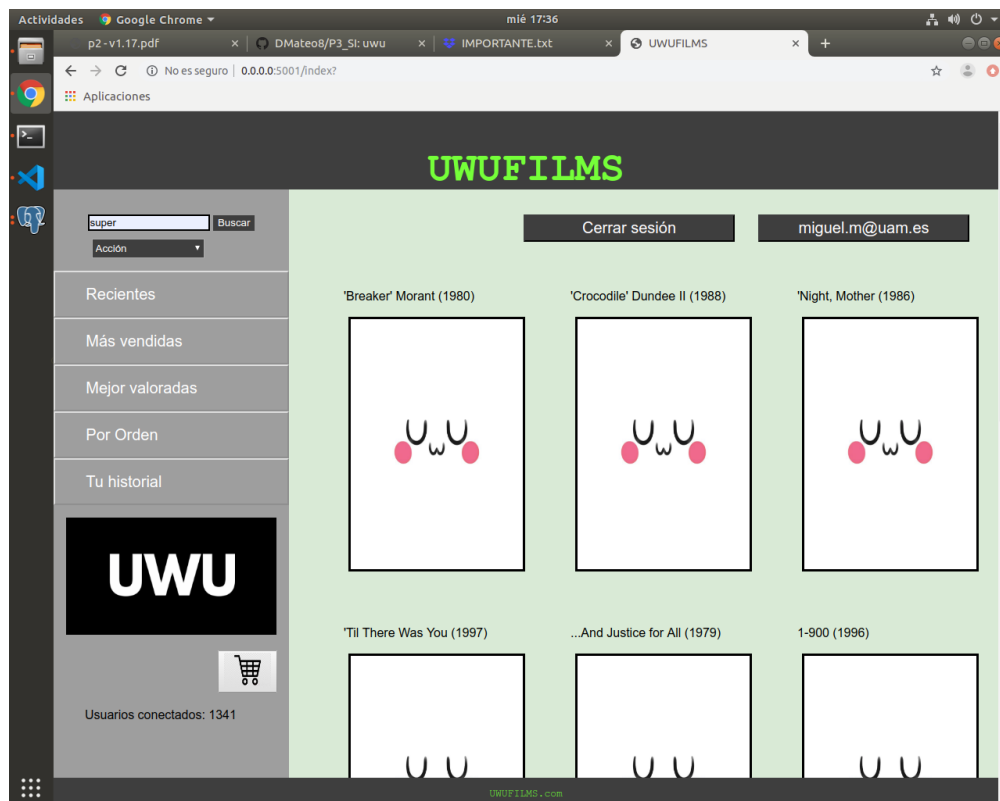
UWUFILMS.com

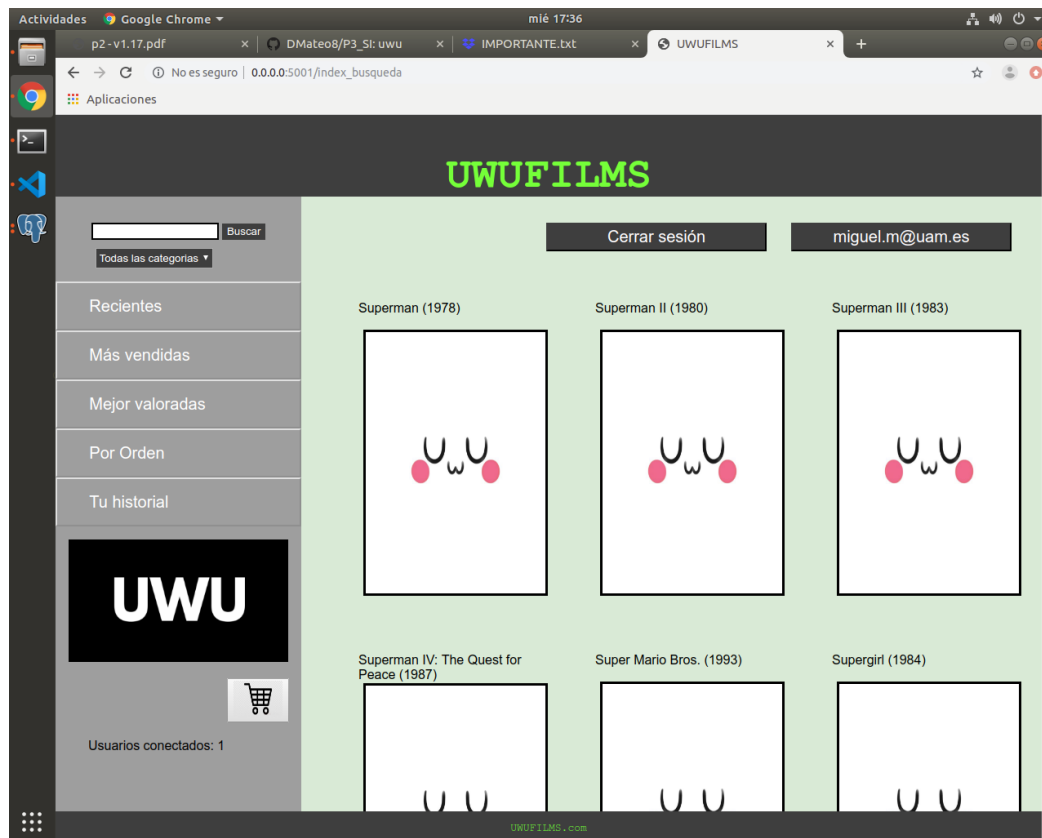
Ejemplo de login.



Resultado del login.

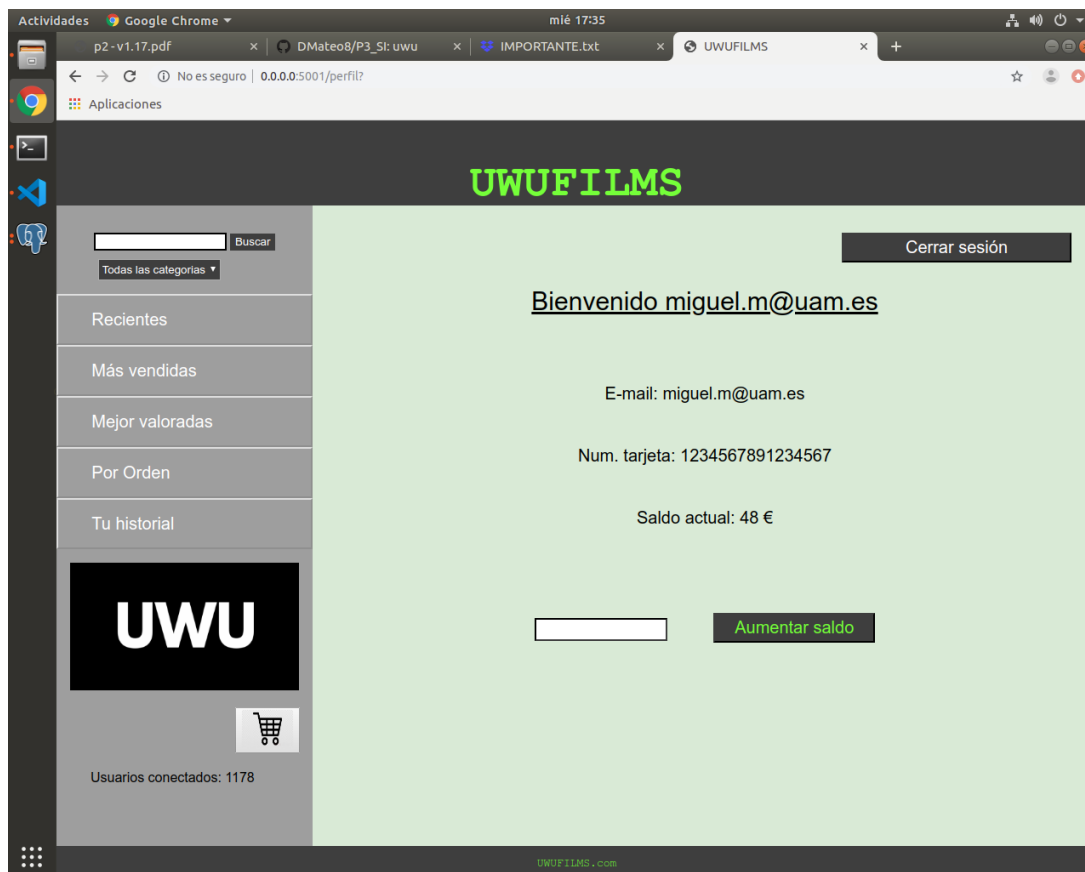
La búsqueda es como la anterior práctica, pero ahora usamos la base de datos en lugar de diccionarios JSON.





Ejemplo de búsqueda.

El perfil del usuario donde se añade el saldo (un update en el income del customer):



En el historial se recuperan los pedidos del usuario:

UWUFILMS

Cerrar sesión | tidily.hah@kran.com

Tu historial:

Fecha	Precio(€)	Estado	Mas detalles
2016-09-04	172.30	Shipped	TAX
2019-01-26	14.16	Shipped	TAX
2016-05-10	161.13	Shipped	TAX
2015-01-14	17.00	Processed	TAX
2018-07-21	187.17	Shipped	TAX
2018-08-10	123.12	Shipped	TAX
2017-10-17	43.10	Processed	TAX

UWUFILMS.com

Ejemplo de historial.

UWUFILMS

Cerrar sesión | tidily.hah@kran.com

Tu historial:

Fecha	Precio(€)	Estado	Mas detalles
2016-09-04	172.30	Shipped	15% Precio neto: 149.83 €
2019-01-26	14.16	Shipped	TAX
2016-05-10	161.13	Shipped	TAX
2015-01-14	17.00	Processed	TAX

UWUFILMS.com

Ejemplo de historial abriendo más detalles (con tax y el precio neto). No hemos puesto los productos del pedido porque era más complejo.

NOTA: Debido a que íbamos pillados de tiempo y teníamos errores en la implementación del carrito, no nos dio tiempo a hacer la funcionalidad del carrito.

Archivos:

Se proporcionan en el zip los siguientes archivos:

- En la carpeta “Modelo Entidad-Relacion” se encuentra el pdf con el modelo ER.
- En la carpeta “SQL” se encuentran todos los archivos sql con las consultas, procedimientos y triggers pedidos.
- Dentro de la carpeta “app” se encuentra toda la funcionalidad de la aplicación. En el archivo “routes.py” están las funciones del controlador y en “database.py” las funciones de acceso a la BD utilizadas en el anterior nombrado. En “templates” las plantillas html y en “static” ficheros JS y CSS.
- El archivo “start.wsgi”.

NOTA: No incluimos las sesiones ni el entorno que hemos utilizado en las pruebas, por lo que es necesario realizar los comandos de virtualenv y el entorno Flask antes para lanzar la aplicación sin problemas.