**Ejercicio 5 - Definición e implementación de un**

**protocolo de aplicación**

**5.1. Descripción de la aplicación, funcionalidad y actores que intervienen**

Hemos decidido implementar un servicio de entradas de cine. Para ello hemos creado un servicio cliente/servidor parecido a los de los otros ejercicios de la práctica.

Esta aplicación nos ofrece distintas funcionalidades. En primer lugar, hacer *login* y *logout* de la misma, así como registrar nuevos usuarios. A la hora de registrar un nuevo usuario se exige una confirmación de contraseña para más seguridad y se comprueba que el nombre de usuario no esté ya en uso. Tras esto volvemos al menú de inicio y se ha de elegir una de las funcionalidades ya nombradas. Una vez que estamos *logeados*, se nos ofrecen tres opciones, comprar una entrada, devolver una entrada (permitido únicamente si el usuario ha comprado alguna previamente), o ver las entradas ya compradas por el usuario.

Hemos creado dos archivos para almacenar distintos datos de interés. Uno de ellos es *usuarios.txt*,en el que se encuentran los usuarios junto a sus respectivas contraseñas. Este archivo es el que comprobamos en el momento de hacer *login*. Otro es *cartelera.txt*, el cual almacena las películas que se ofertan. Este archivo es el que se lee en el momento de mostrar la cartelera.

Una vez la aplicación está iniciada, en el momento de comprar una película, se crean dos archivos para la misma, el primero, llamado “*película*”*.txt*, en el que se indica en la primera línea el número de filas y columnas de la sala de cine (así almacenamos dónde se sienta cada usuario y no hay dos personas con la misma entrada), y, en las sucesivas líneas, se almacena el código de la entrada junto al usuario que la ha comprado y su asiento, el segundo se llama “*nombre\_usuario”.txt*,en este archivo, que nunca está vacío (si lo estuviese el programa lo elimina), se almacena en cada línea el código de entrada, la película y el asiento de las entradas que el usuario ya ha comprado. Cuando el usuario compra una entrada el asiento seleccionado no puede estar en *“pelicula”.txt* y tiene que cumplir los límites de tamaño de la sala especificados en la primera línea del archivo *“pelicula”.txt*, y cuando el usuario devuelve una entrada tiene que seleccionar una de las que tiene en posesión de *“nombre\_usuario”.txt*. En caso contrario se devuelve un error pidiendo que se vuelva a introducir.

De esta forma hemos diseñado un servicio de compra de entrada de cine completo, que cumple con las interacciones básicas de uno real, y teniendo en cuenta y resolviendo todos los conflictos reales posibles.

**5.2. Diagrama de estados del servidor**

(LOGIN \* PASS \*/OK)

(\*/ERROR)

(\*/ERROR)

(LOGOUT/BYE)

(LOGOUT/BYE)

(Conexión/HELLO)

(COMPRAR\*/OK)

(CANCELAR/OK)

(\*/ERROR)

(SIGNUP USER \* PASS \*/OK)

(VER/OK)

(\*/ERROR)

(CANCELADO/OK)

(COMPRADO/OK)

(SELECT/OK)

(SELECT/OK)

**5.3. Mensajes que intervienen**

**Cliente**

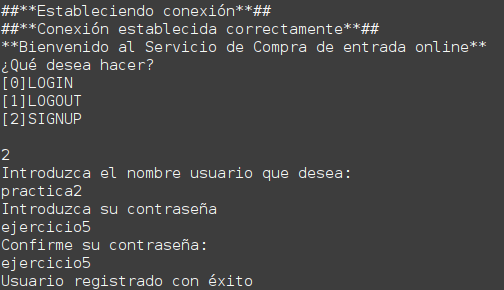
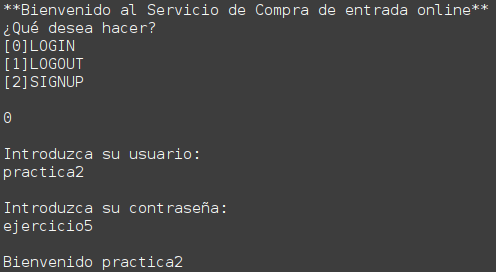
|  |  |  |
| --- | --- | --- |
| Código | Cuerpo | Descripción |
| 1000 | SIGNUP + USER + *user* + PASS + *password* | Mensaje enviado por el cliente para solicitar registrar un nuevo usuario *user* con contraseña *password* |
| 1001 | LOGIN + *user* + PASS + *password* | Mensaje enviado por el cliente para autentificarse con el servidor. |
| 1002 | LOGOUT | Mensaje enviado por el cliente para terminar la conexión con el servidor (Salirse de la aplicación). |
| 1003 | COMPRAR + *película* | Mensaje enviado por el cliente para indicar que quiere comprar la entrada para “*película*”. |
| 1004 | CANCELAR | Mensaje enviado por el cliente para indicar que quiere cancelar la compra de la entrada de “*película*”. |
| 1005 | VER | Mensaje enviado por el cliente para indicar que quiere seguir en la aplicación, ya sea para comprar una entrada o cancelarla. |
| 1006 | CANCELAR + código + película + asiento | Mensaje enviado por el cliente para indicar que quiere cancelar una película y le envía el código, el asiento y la película que quiere cancelar. |

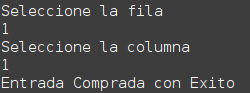
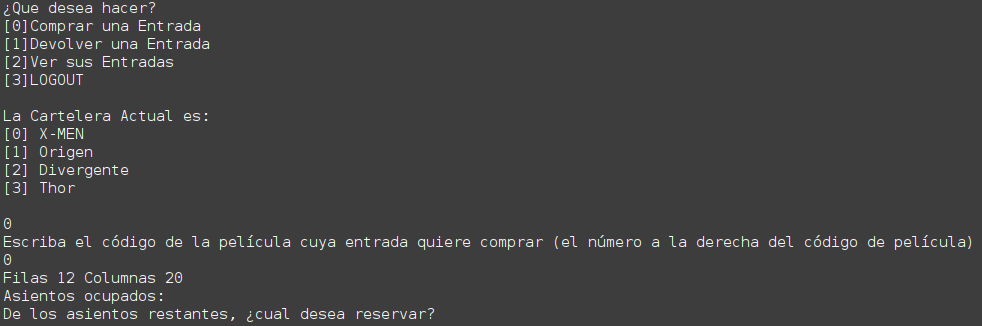
**Servidor**

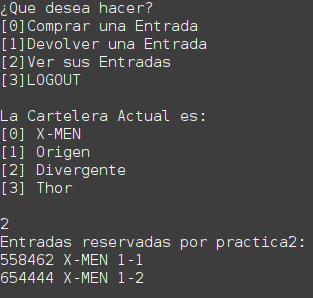
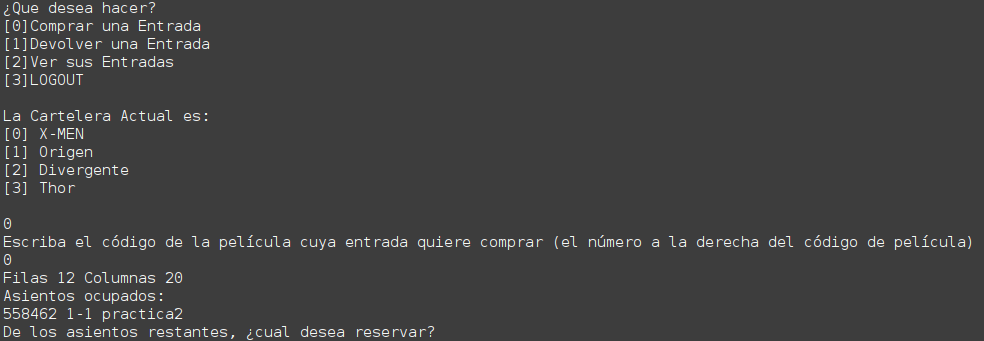
|  |  |  |
| --- | --- | --- |
| Código | Cuerpo | Descripción |
| 200 | OK | Mensaje enviado para informar que la petición se ha recibido correctamente. |
| 201 | HELLO | Mensaje enviado al establecer conexión |
| 300 | ERROR + “No disponemos de esa opción” | Mensaje enviado cuando el cliente solicita una petición que el servidor no puede procesar. |
| 301 | ERROR + “Usuario o *password* no existen” | Mensaje de error cuando el usuario no existe o su *password* no es correcta. |
| 303 | ERROR + “Lo sentimos, la sala está llena hoy, elija otra película” | Mensaje de error cuando el usuario intenta comprar una entrada para una película cuya sala para ese día está llena. |
| 304 | ERROR + “No tiene ninguna compra realizada” | Mensaje de error cuando el usuario intenta cancelar la compra de una entrada sin haber comprado una previamente. |
| 305 | ERROR + “Usuario ya registrado” | Mensaje de error que manda el servidor para indicar que el usuario ya está registrado |
| 500 | BYE + “¡Hasta la próxima!” | Mensaje enviado cuando el cliente decide abandonar el servidor. |

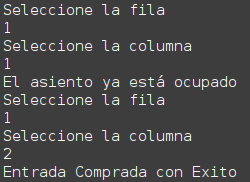
**5.4. Evaluación de la práctica**

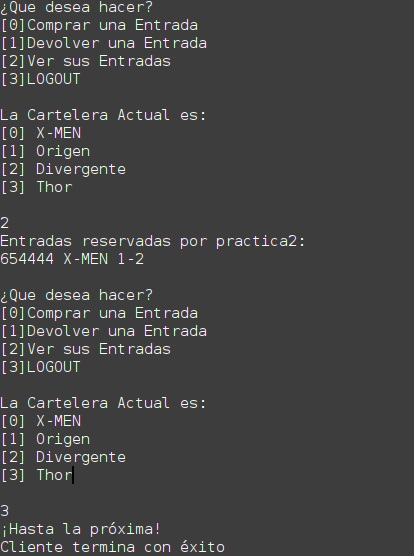
A continuación, mostraremos un proceso completo del funcionamiento del servicio. En primer lugar, mostramos como iniciamos el servicio, registramos un nuevo usuario (1) y hacemos *login* con ese usuario (2).

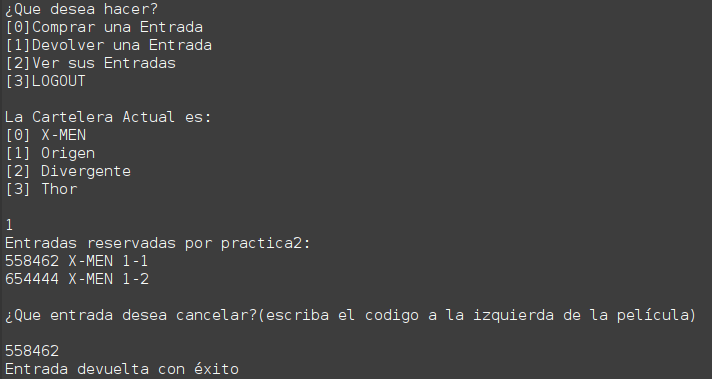


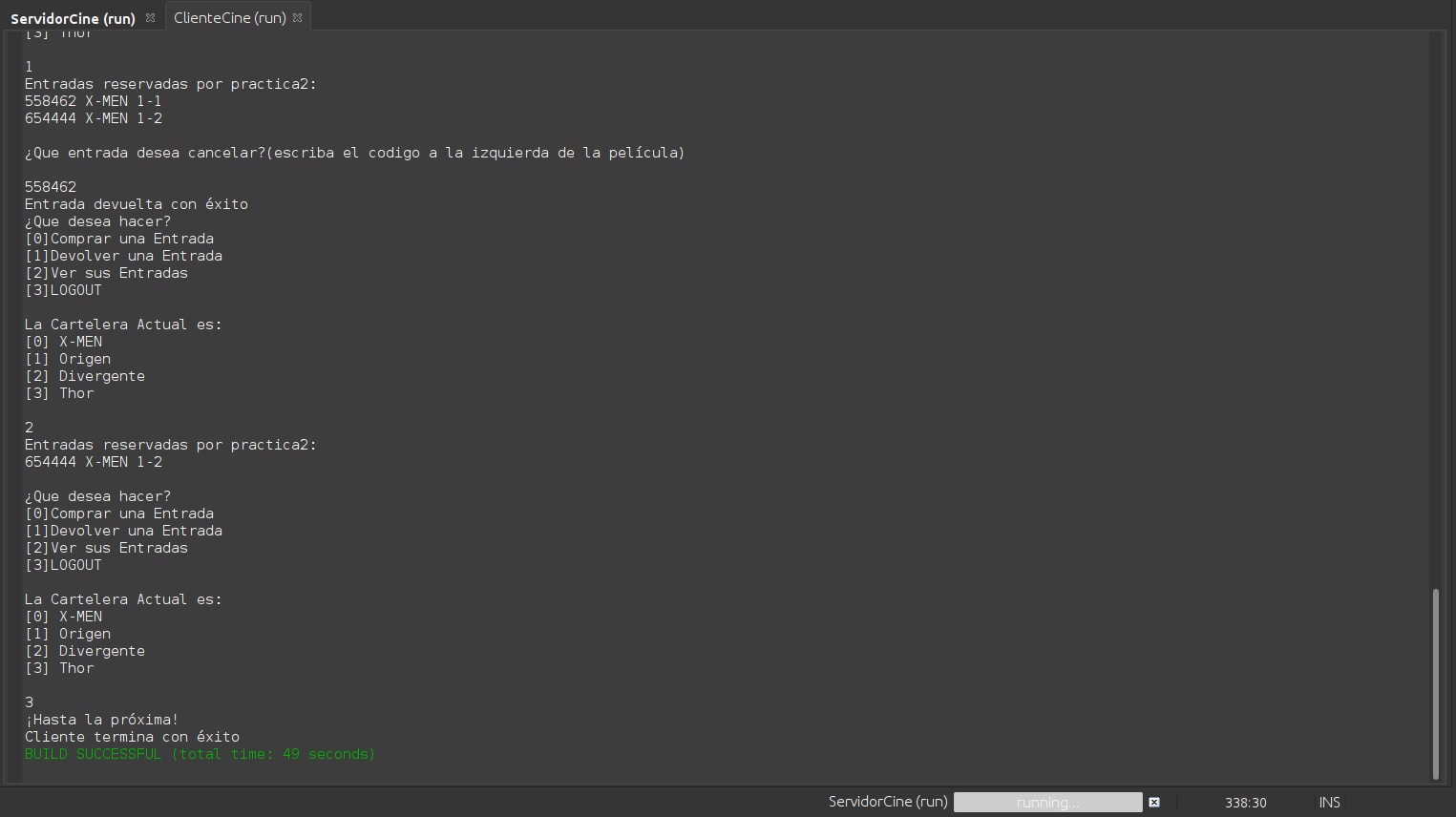
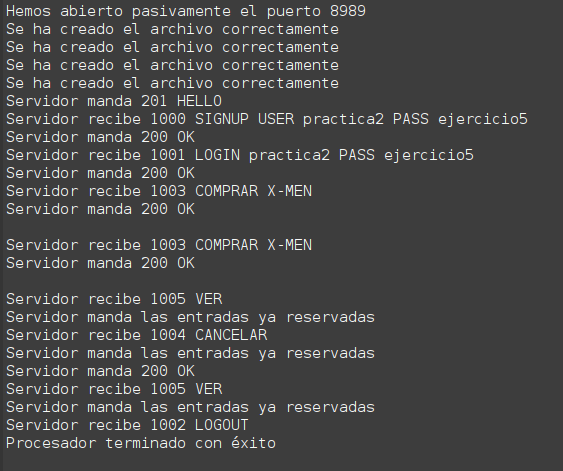
Ahora decidimos comprar dos entradas, la primera la compramos sin problemas (3), en la segunda intentamos comprar un asiento ya ocupado para comprobar que no nos deja y ya compramos la segunda (4). Tras esto, las mostramos (5).





En tercer lugar, cancelamos una de las entradas compradas (6) y mostramos de nuevo nuestras entradas para comprobar que se ha cancelado correctamente y hacemos *logout* (7).



Y ya en último lugar, mostramos como, tras hacer *logout*, el servidor sigue funcionando (8) y también los mensajes recibidos por el servidor (9).