

**BLOCKBUSTER IS BACK!!**

Alumno: Ignacio Martín Smirlian

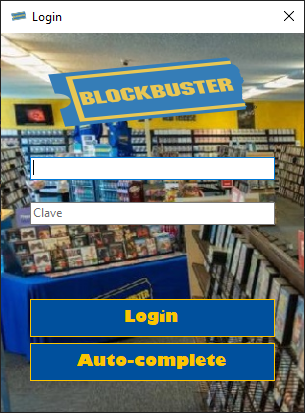
Materia: Laboratorio de Programación II

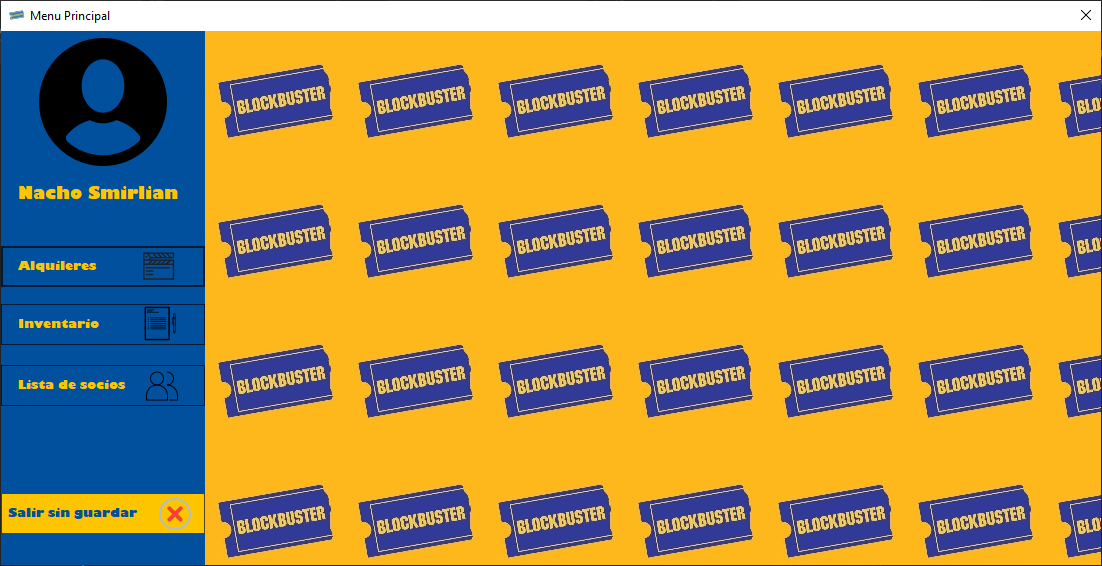
Instancia: Entrega Trabajo Práctico #3

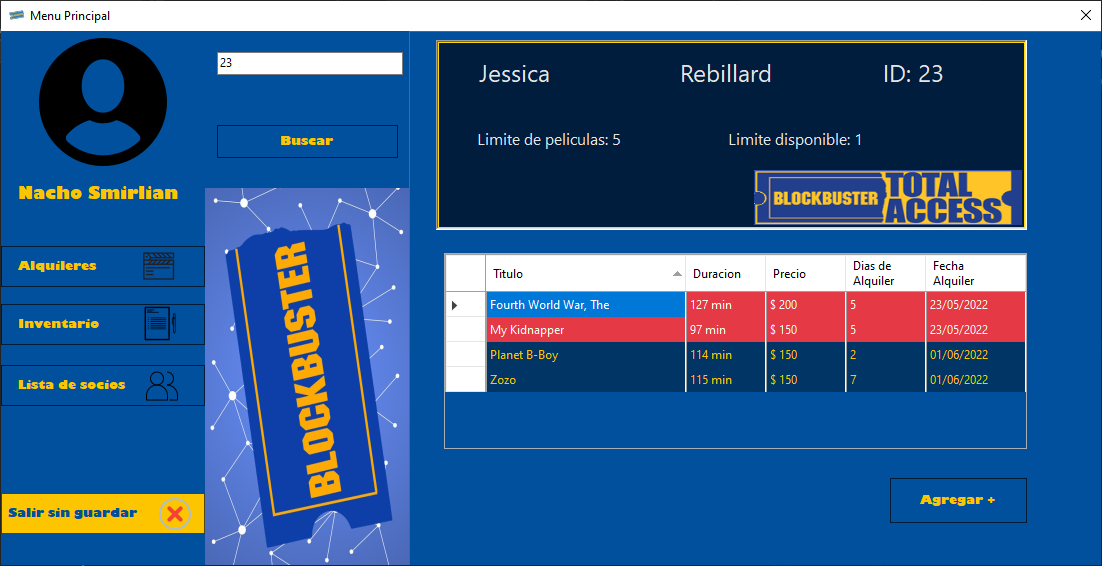
Bienvenidos a mi TP3!! En este caso, el tema del programa fue elegido por mi niño interior. Espero que les guste!

# Flujo del programa:

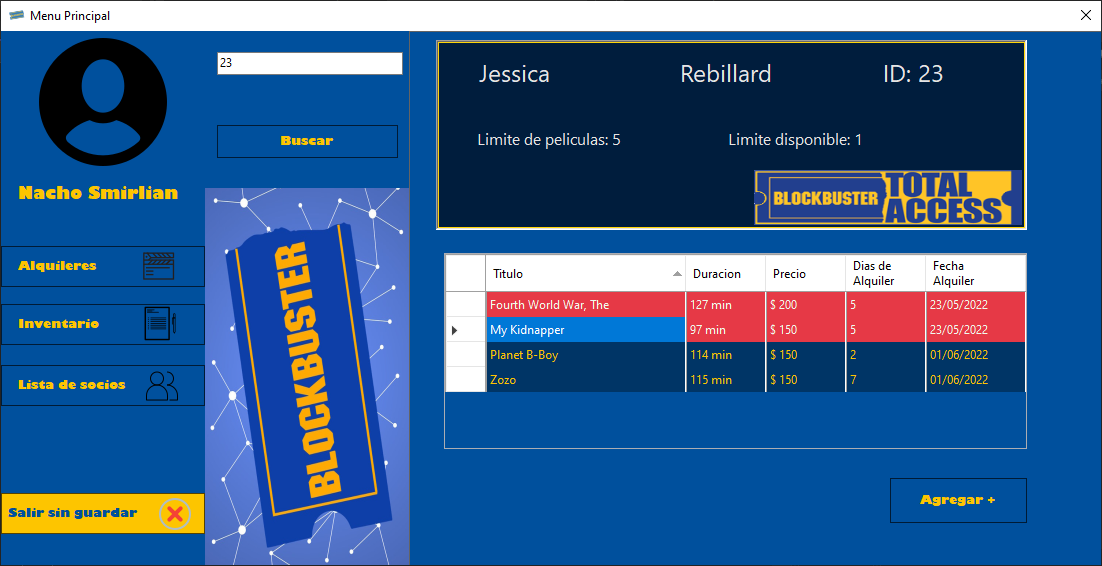
El flujo del programa es muy simple ya que en este caso decidí incursionar en manejar todos los formularios (o al menos los principales) desde el menú principal.

Al iniciar, nos vamos a encontrar con la clásica pantalla de Log In. Si presionamos el botón de autocompletar van a ingresar con “mi usuario” pero pueden probar cualquier de los otros usuarios (se encuentran en la carpeta Debug dentro de la carpeta Recursos. El archivo se llama baseDatosEmpleados).

Una vez logueados, podemos acceder al menú principal desde donde podremos acceder a los 3 “módulos” de este programa. 

Accediendo al primer botón, entramos en la **sección alquileres**. Aquí podremos buscar a cada socio por su número de ID y traer toda su información (nombre, apellido, límite de películas, alquileres activos, etc). 

Si el usuario presenta alquileres prontos a vencerse (pero aún no vencidos) la fila de ese alquiler se coloreará de **NARANJA** y si el alquiler ya está vencido, el color será **ROJO**. Si ninguno de estos escenarios se da, el color ser el clásico azul oscuro.

En este menú nosotros tenemos 3 acciones posibles:

BUSCAR >>> Apretando este botón nos traerá el usuario cuyo ID

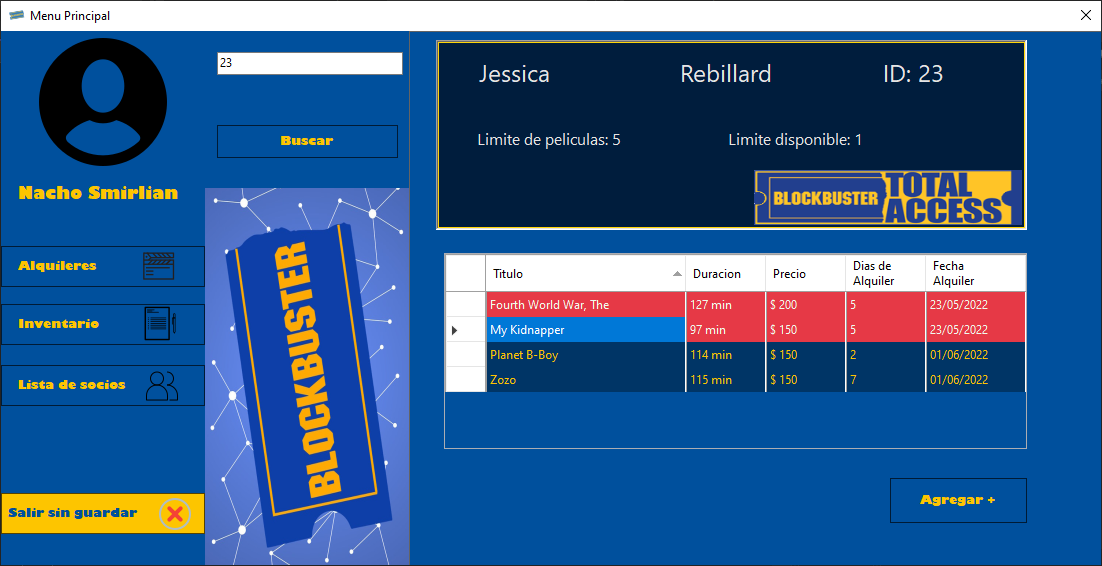
ingresemos en el textbox.

DEVOLUCION >>> Esto se da apretando la fila del alquiler que deseamos devolver.



En este pequeño formulario nos muestra la información de la película a devolver y su penalidad (existe un cálculo interno donde la penalidad es igual a $150 por día de demora, pero, además, si el usuario el premium, este paga solo un 30% de la penalidad).

Si presionamos Confirmar, la película se borra de la lista de alquileres de socio, y vuelve a la lista de películas de Blockbuster (aumentando en uno el stock de la película). Si, por el contrario, cerramos este form, la devolución se anula.

>>> Si presionamos este botón, abriremos el formulario para

agregar un alquiler a este Socio

\*Si el usuario ya llegó a su límite de películas (5 socios clásicos y 20 socios premium) este botón estará deshabilitado.



Aquí encontramos 2 datagrid. El primero contiene todas las películas disponibles (aquellas que no haya stock no aparecerán) con un buscador de titulo dinámico.

Por debajo, está el datagrid con toda la información del “CandyShop”. Aquí el socio puede elegir snacks o golosinas para comer. Estos se registrarán en la compra actual, pero no quedarán guardados en el historial del socio.

Para elegir tanto las películas como los productos, simplemente hay que hacer click en la fila que deseamos agregar y ya aparecerá en el richtextbox de la derecha. Si hubo algún error, y deseamos sacar algún producto, simplemente haciendo click en el botón editar convertiremos el richtextbox en un checkboxlist donde podemos seleccionar los items a borrar.

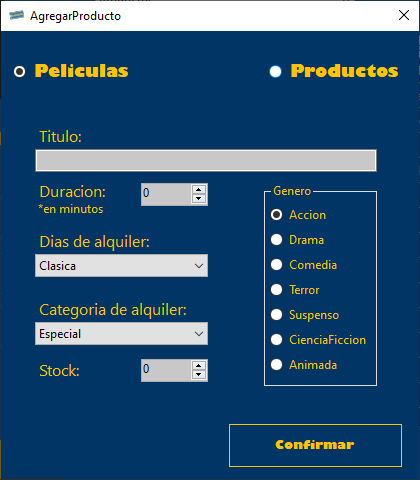
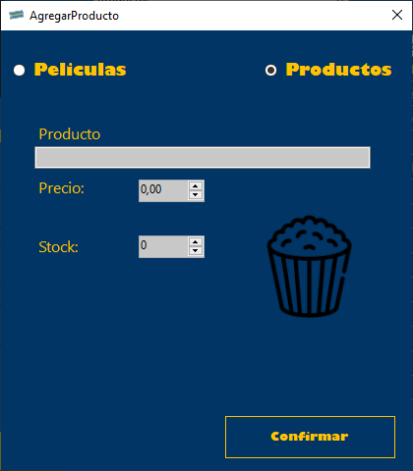
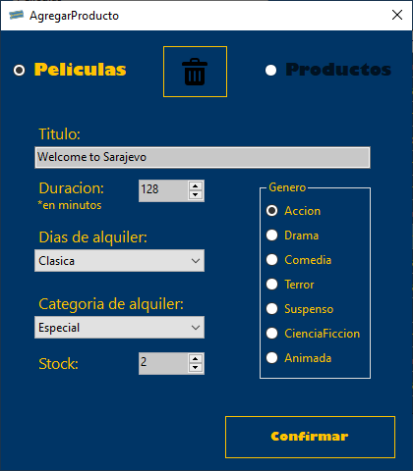
Cualquiera de estos movimientos que describí, impactan directamente en el inventario de los productos en tiempo real.

Para finalizar, simplemente Aceptar o cerramos el form, anulando toda la operación

**Segundo módulo, el inventario**. 

Aquí tenemos un formulario que consta casi únicamente de un datagrid con toda la información de los diferentes productos y películas. Para intercambiar entre películas y productos solo tenemos que seleccionar la opción correcta del combobox.

Este datagrid tiene un comportamiento similar al que encontramos en la pantalla anterior, pero se le agregó un pequeño efecto que resalta toda la fila en la que estamos parados, sin necesidad de seleccionarla (efecto hover).

En esta pantalla tanto si apretamos el boton AGREGAR + como si hacemos click en alguna fila del datagrid, el formulario que va a aparecer es exactamente el mismo. La única diferencia es que al presionar AGREGAR +, el formulario está vacío y nos permite elegir cargar un producto o una película. En cambio, al hacer click en una celda, el formulario aparecerá con toda la información ya cargada del item para que podamos editarlo, con el radio botón del otro elemento deshabilitado, y un botón de borrar, para eliminar el item.

Finalmente, llegamos al **último módulo, el de socios**. Aquí podremos realizar búsquedas rápidas de socios por ID, Nombre, Apellido o incluso e-mail.

El funcionamiento del datagrid es idéntico al formulario anterior. Al hacer click en una celda, nos abrirá un formulario con todos los datos de ese socio para que podamos modificarlo y actualizarlo. Si presionamos AGREGAR + abriremos un formulario en blanco.

Importante destacar que existen validaciones para cada campo (Nombre, Apellido, Email, Teléfono y Tarjeta) de acuerdo a los criterios de formato más comunes en Argentina (muchos socios ya cargados no respetan ese formato, principalmente teléfono, porque fueron traídos de Mookaro, por lo que al guardar una modificación, el programa va a pedirnos que ingresemos un formato válido)



*Información: Sólo se pide tarjeta de garantía para los socios clásicos ya que, en mi lógica, los usuarios premium ya tienen una suscripción mensual, por lo que no necesitan garantía.*

Llegando al final de la revisión del flujo, es importante destacar 2 momentos que suceden “por detrás” de la aplicación. En primer lugar, la CARGA DE DATOS.

Esta se realiza en 2 partes. Al abrir el formulario de Login se cargan al sistema toda la información de los usuarios. Luego, al abrir el menú principal, se leen toda la información de las películas, productos, y socios.

En segundo lugar, el GUARDADO DE LOS DATOS. Esto se hace de manera automática al cerrar el programa (informándonos si se pudo guardar todo con éxito o no). En este punto se actualizan todas las bases de datos y se guarda la facturación diaria que se va guardando a lo largo de todo el programa a través de cada transacción de compra o devolución. (el nombre del archivo donde se guarda la facturación diaria lleva la fecha del día, por lo tanto, si ya existe un archivo con la fecha de hoy, se agregará la información a ese archivo. De lo contrario se creará un nuevo archivo). Sin embargo, hay casos en que el usuario puede querer salir sin guardar sus cambios, y es por eso que incluí un botón de Salir sin guardar en el panel principal.

Por último, me parece importante destacar que en esta instancia yo elegí hacer la clase Blockbuster estática ya que en mi parcial yo había utilizado una clase restaurante instanciable y me pareció interesante probar el funcionamiento con una clase estática.

# Temas:

* Excepciones:

A lo largo de todo el programa fui evaluando los escenarios donde podía llegar a darse una excepción y se agregar los try/catch necesarios. A su vez, se han creado 4 excepciones específicas para la validación de Email, Nombre o Apellido, Tarjeta de crédito y Teléfono.

* Archivo >> Biblioteca de clases >>> Excepciones
* Aplicado en >>> AgregarSocio
* Pruebas unitarias:

En el proyecto ”TestUnitarios” podemos encontrar diversos test unitarios para los métodos de validación previamente mencionados.

* Archivo >>> TestUnitarios >>> Excepciones
* Aplicado en >>> AgregarSocio
* Tipos genéricos:

Puede verse en 2 partes en el TP. En primer lugar, en todos los métodos de serialización de la ClaseSerializadora que leen o escriben archivos de tipo T. También puede verse que la clase Alquiler es genérica. Esto es así porque mi idea era expandir el sistema incluyendo videojuegos y series, por lo que los alquileres puedan no ser necesariamente una película. Lamentablemente, por falta de tiempo no logré implementarlo.

* Archivo >>> TestUnitarios >>> Excepciones
* Aplicado en >>> AgregarSocio
* Interfaces:

En el TP se usa la interfaz IModificarse, que incluye la firma de 2 métodos que son utilizados tanto por el formulario AgregarProducto, AgregarSocio y CargarDatosForm.

* Archivo >>> IModificarse
* Aplicado en >>> AgregarProducto, AgregarSocio y CargarDatosForm
* Archivos y serialización:

Esto se puede ver en toda la información que el programa maneja, ya que todos los datos de usuarios, películas, socios, etc, son cargados a través de archivos JSON o XML, y luego, todas las modificaciones son serializadas nuevamente y guardadas en los formatos indicados.

* Archivo >>> ClaseSerializadora
* Aplicado en >>> Login load event, MenuPrincipal load event, MenuPrincipal formClosing event.

# Conclusión:



*Pues más o menos copié mi parcial 1 y agregué fotos de Blockbuster, archivos y una interfaz porque se me hizo bonita*