Funcionamiento del logueo:

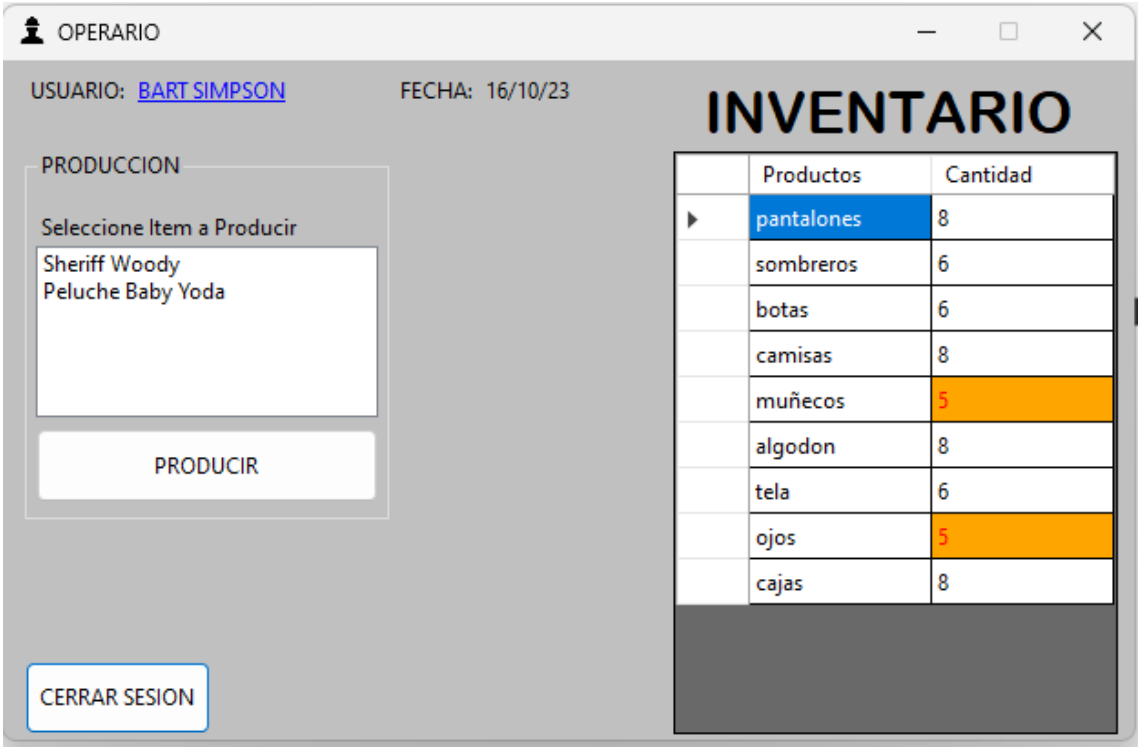
Como primera pantalla, vamos a tener la pantalla de logueo. La cual le pedirá al usuario, un usuario y una contraseña.

Tocando el botón operario, nos seteara automáticamente un usuario y una contraseña de un empleado operario, lo mismo ocurriría con el botón supervisor.

Lista de usuario creados:

NOMBRE TRABAJADOR	USUARIO	CONTRASEÑA	TIPO EMPLEADO
Thiago Castello	S-42343951	42343951	SUPERVISOR
Hugo Basilotta	S-29739267	29739267	SUPERVISOR
Bart Simpson	S-34858321	34858321	OPERARIO
Homero Simpson	S-11111111	11111111	OPERARIO

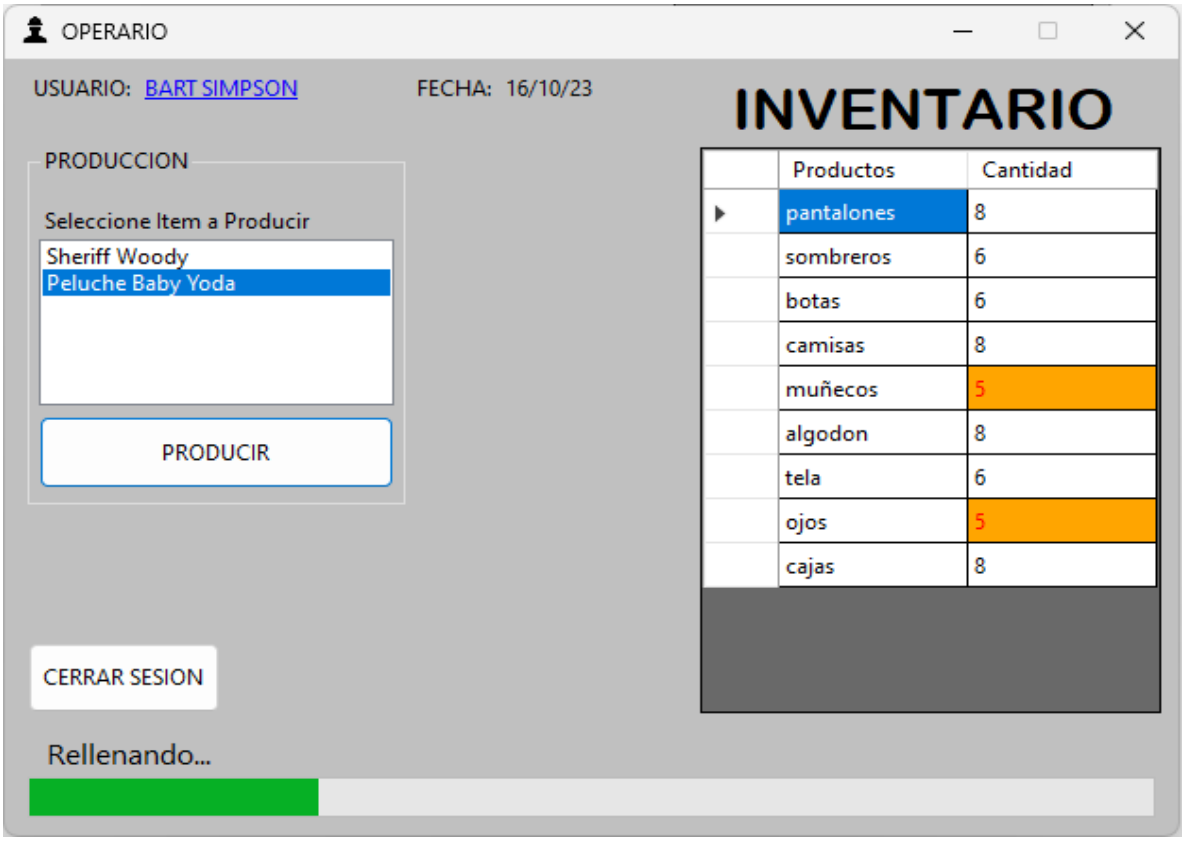
Pantalla Operario



Dentro de la pantalla de los operarios, a la izquierda podremos ver los productos disponibles que tenemos para fabricar, actualmente contamos con la producción de figuras de acción de Woody el Vaquero, y peluches de Baby Yoda.

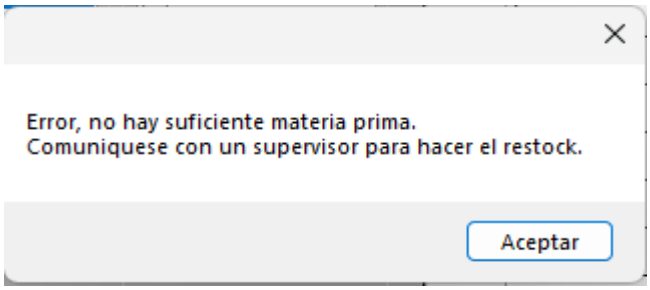
A la derecha, tenemos el stock, el cual nos informara todos los productos que tengamos dentro de nuestro stock.

En caso de generar un producto, la pantalla nos informara sobre el proceso de nuestro producto mediante una barra de progreso.

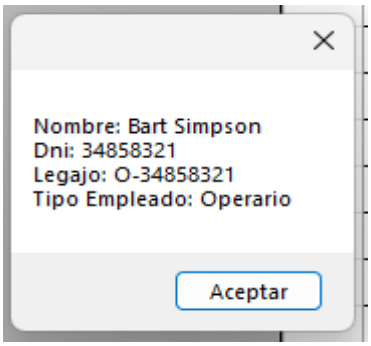


La cual en caso de terminar correctamente el proceso de fabricación, nos avisara y se descontaran los productos utlizados, agregando nuestro peluche recién creado.

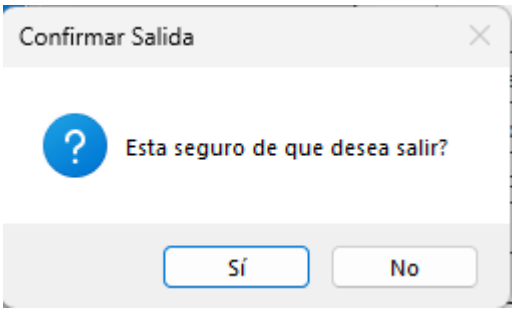
En caso de no contar con materia prima suficiente, el programa nos informara mediante un mensaje en la pantalla.



Ademas tocando en el nombre de usuario, se nos abrirá un mensaje con los datos del trabajador.

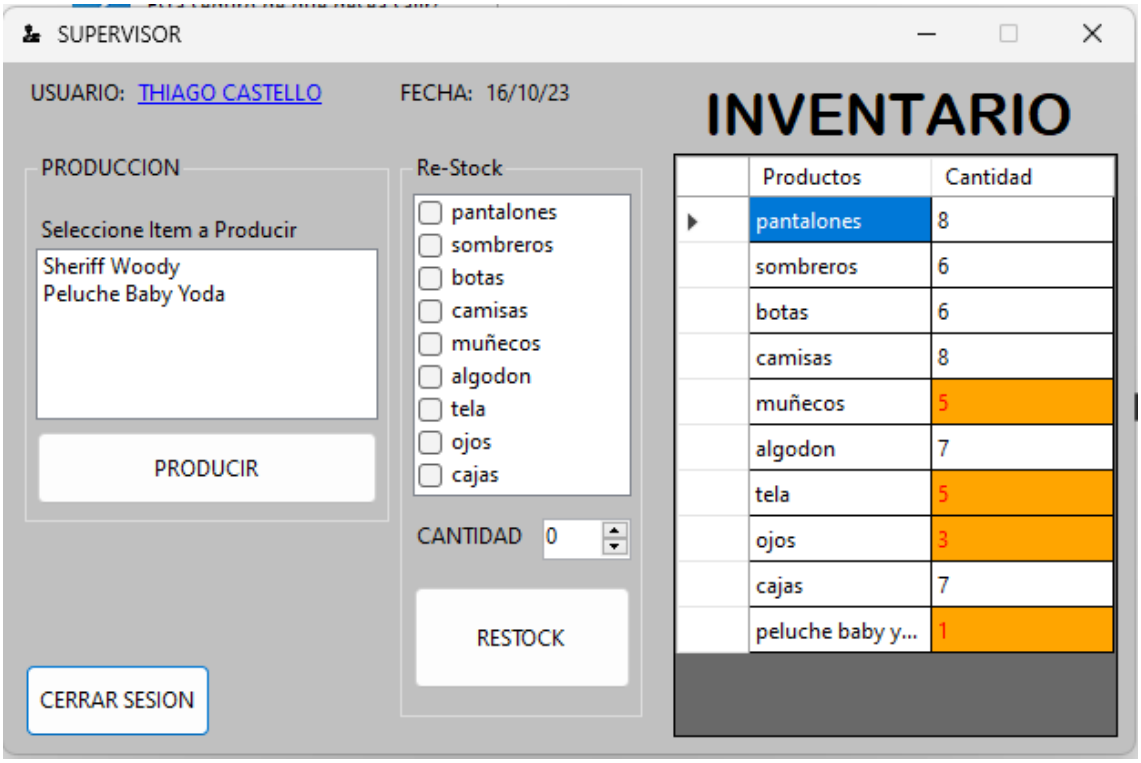


Para finalizar, tenemos el boton de cerrar sesion, el cual nos preguntara si queremos confirmar el cierre.

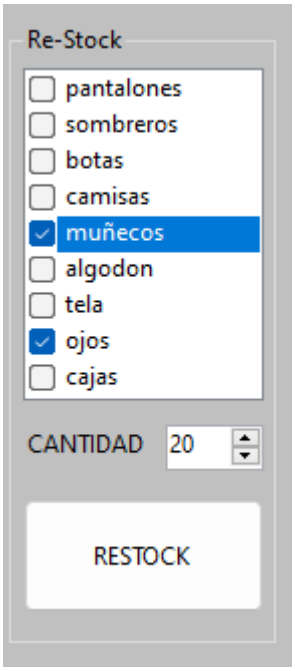


Pantalla supervisor:

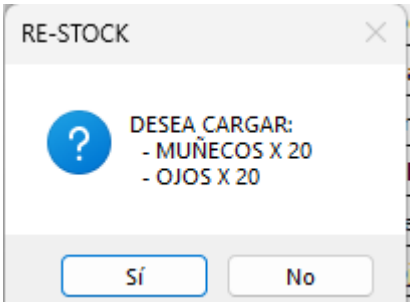
Dentro de la pantalla tenemos la misma informacion con respecto al stock y a la produccion que un operario, pero con la diferencia de tener acceso a reponer el stock.



Donde podremos seleccionar la materia prima que querramos agregar y su cantidad correspondiente, por ejemplo, queremos agregar 20 ojos y muñecos.

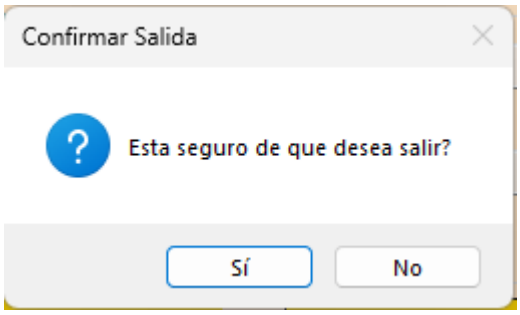


Luego nos preguntara si queremos confirmar el re-stock.



Y por ultimo agregara esas cantidades a nuestro stock.

Por ultimo, habiendo cerrado la sesión, dentro del menú principal tenemos el botón de salir, el cual nos pedirá una confirmación, para asi cerrar el programa.



Temas vistos en clase

Herencia:

tenemos una clase base llamada Trabajador, de la cual van a heredar los operarios y supervisores

```
5 referencias
public sealed class Supervisor : Trabajador
{
```

```
public sealed class Operario : Trabajador
{
    2 referencias
```

Sobrecarga:

Se utiliza la sobrecarga, por ejemplo dentro de los formularios para enviar como parámetro el usuario que accedió al sistema.

```
5 referencias
public partial class frmSupervisor : frmOperario
{
    0 referencias
    public frmSupervisor()
    {

    }
    1 referencia
    public frmSupervisor(string tbNombreUsuario) : base(tbNombreUsuario)
    {
        InitializeComponent();
    }
```

O para preguntar por el nombre del trabajador

```
public static bool operator == (Trabajador laburante, string nombre)
{
    return nombre == laburante.nombre;
}

0 referencias
public static bool operator !=(Trabajador laburante, string nombre)
{
    return !(nombre == laburante.nombre);
}
```

Propiedades:

Se utilizaron para el encapsulamiento de mi clase trabajador por ejemplo.

```
public string NombreUsuario
{
    get
    {
        return nombreUsuario;
    }
    set
    {
        nombreUsuario = value;
    }
}
6 referencias
public string Password
{
    get
    {
        return password;
    }
    set
    {
        password = value;
    }
}
```

Colecciones:

Se utilizaron diccionarios, para guardar las materias primas necesarias con sus cantidades respectivas para poder fabricar el producto. Y Las listas para guardas los procesos de fabricación de cada producto.

```
public sealed class Producto : Empresa
{
    private string nombre;
    private int id;
    private string marca;
    private List<string> procesosProduccion;
    private Dictionary<string,int> materiales;
```

Enumerados:

```
12 referencias
public enum TipoEmpleado {Operario, Supervisor};

15 referencias
public enum MateriasPrimas {pantalones, sombreros, botas, camisas, muñecos,
                            algodon, tela, ojos, cajas}

2 referencias
public enum ProcesosProduccionWoody {vestir, pintar, empaquetar}

2 referencias
public enum ProcesoBabyYoda {rellenar, coser, aplicarMagia, empaquetar}
```

Clases estáticas:

Se utilizaron para generar un Stock.

```
8 referencias
public static class Stock
{
    public static Dictionary<string, int> stock;

    0 referencias
    static Stock()
    {
        stock = new Dictionary<string, int>();
        CargarInventario();
    }
```

Polimorfismo:

Se utilizaron clases abstractas, la cual no nos permitirá instanciar un objeto de ese tipo

```
11 referencias
public abstract class Trabajador
{
    private string nombreUsuario = "";
    private string password = "";
    private string nombre = "";
    private ulong dni;
    private string legajo = "";
    private TipoEmpleado tipoTrabajador;

    2 referencias
    protected Trabajador(string nombre, ulong dni, TipoEmpleado tipoTrabajador)
    {
        this.nombre = nombre;
        this.dni = dni;
        this.tipoTrabajador = tipoTrabajador;
        CrearCuentaEmpleado();
    }
```

Métodos virtuales, que nos permitirá sobrescribir el método heredado por la clase base.

```
5 referencias
public virtual string MostrarInfo()
{
    StringBuilder sb = new StringBuilder();

    sb.AppendLine($"Nombre: {this.Nombre}");
    sb.AppendLine($"Dni: {this.Dni}");
    sb.AppendLine($"Legajo: {this.legajo}");

    return sb.ToString();
}
```

Métodos abstractos, los cuales deberán heredar obligatoriamente todas las clases base.