

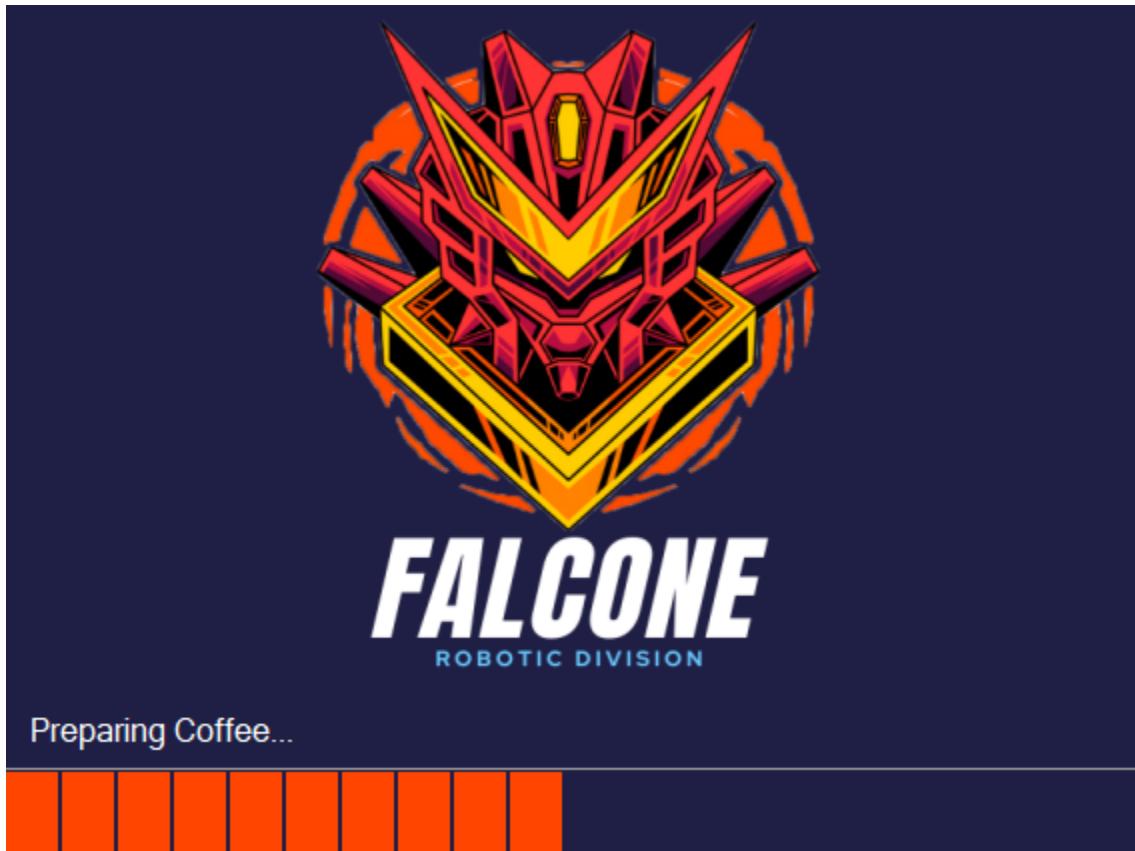
## TP3 Final - Laboratorio 2

Amilcar Facundo Falcone - 2D.

# Fabrica de Robots

3 de Junio del 2021

---



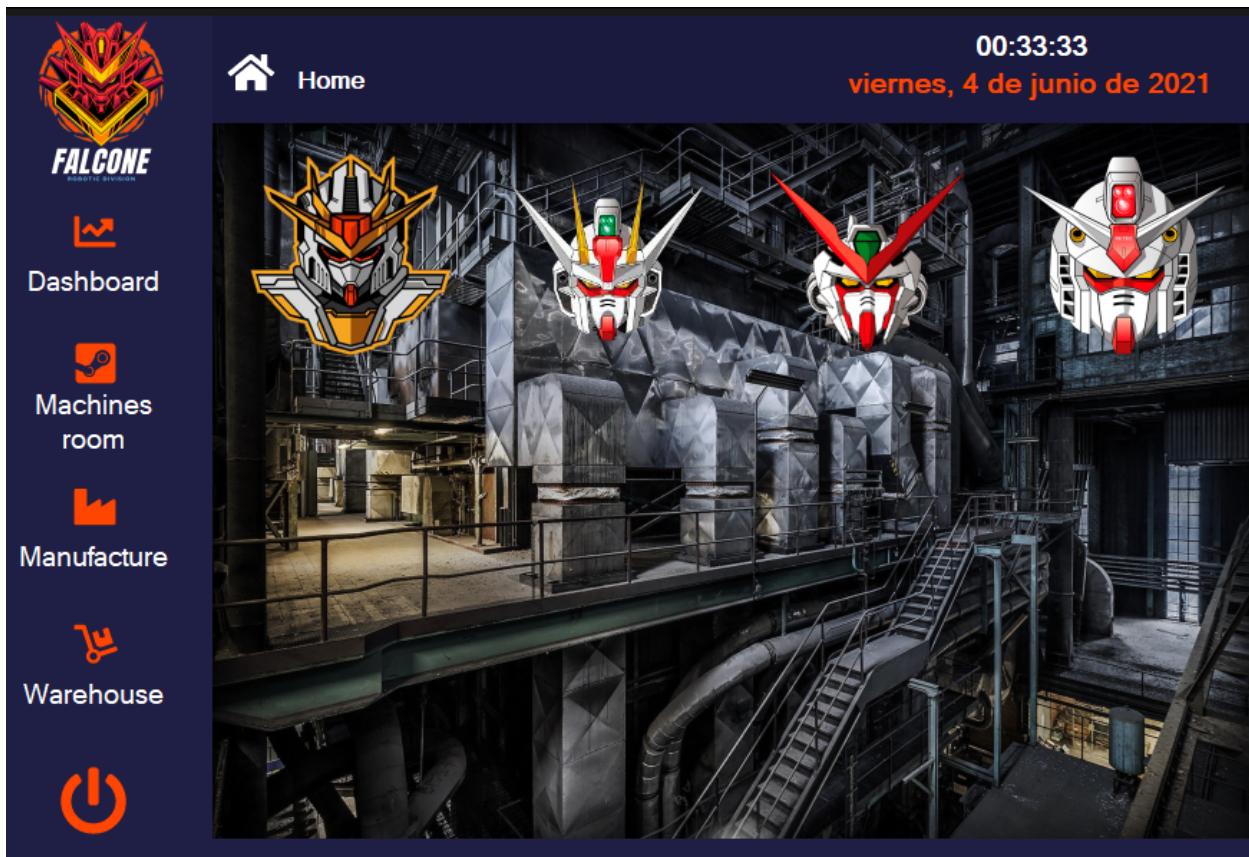
---

## Explicación de la mini app

El siguiente proyecto simula una pequeña fábrica de robots en la que se podrán crear algunos diseños propuestos. Dicho programa para que funcione correctamente deberá inicializar primero su stock de materiales y de ser correcta su cantidad, con las que necesita el modelo robótico, procederá a fabricarlo para luego poner el robot en su almacén. En el almacén o warehouse, se podrá ver los modelos fabricados como así también el stock actual de materiales.

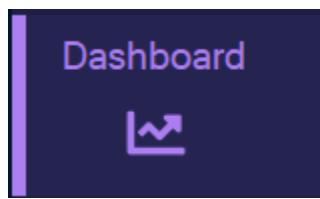
Puedes verlo funcionando en: [Pequeño Tour Por el TP3](#)

# Pequeño Lobby

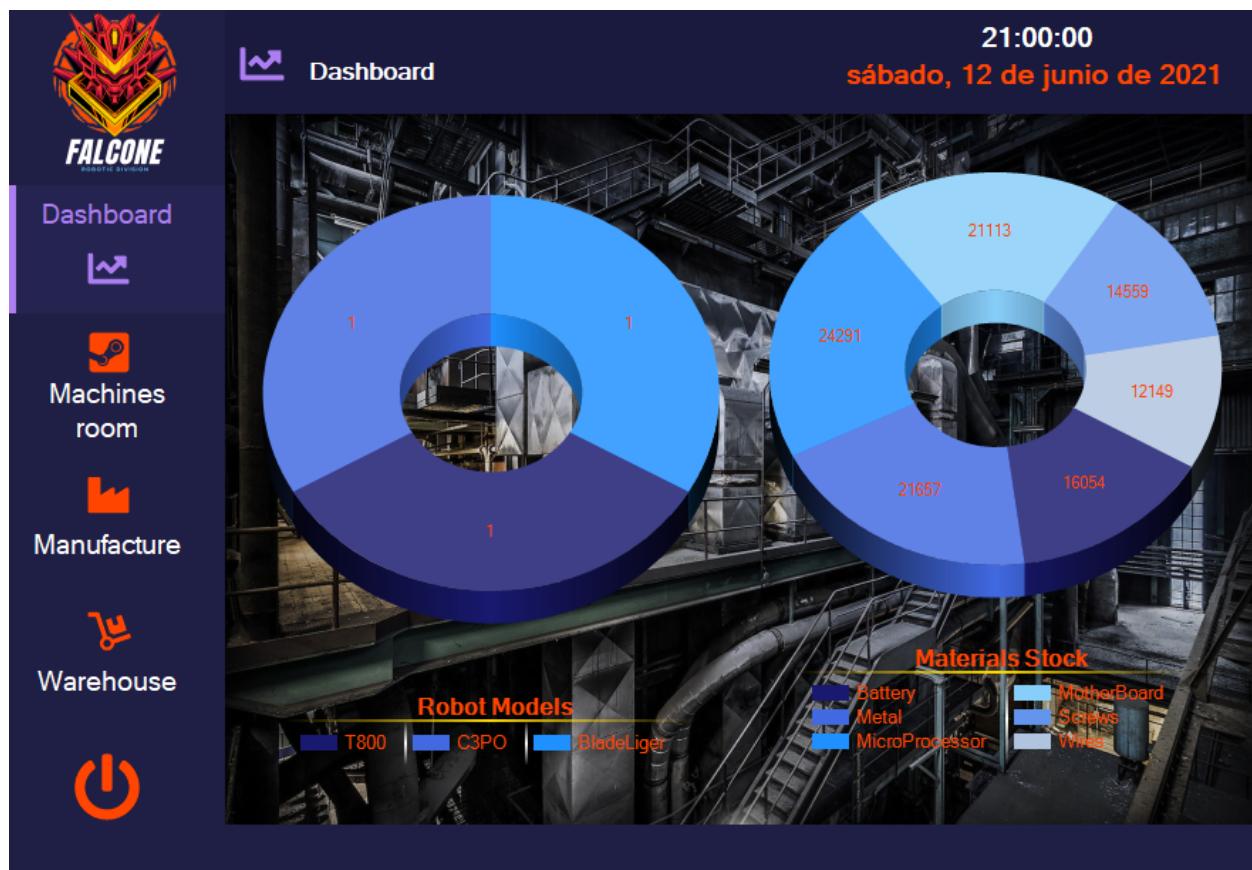


## Descripción:

El programa te da la posibilidad de crear stock tanto de materiales como de robots, al consumir esos materiales.



En el botón 'Dashboard' podrán ver dos gráficos de dona con la cantidad y porcentaje de materiales y robots que hay en el stock de la fábrica, en caso de no haber robots y/o stock de materiales, no se visualizarán los gráficos.





### **Si es la primera vez que ejecutas el programa, Empezá por acá**

En el botón '**Machines Room**' podrá inicializar la fábrica e importar materiales a la misma usando números random para aumentar el stock. La primera vez que ejecutes el programa, Acá aparecerán dos botones, uno de ellos será para inicializar la fábrica, al darle clic el botón desaparecerá para nunca más volver. Con el otro botón se podrá importar materia prima a la fábrica (se generan de manera random) y los valores se podrán visualizar en el gráfico de dona. Cada vez que se apriete el botón de '**Import Stock**', los valores del gráfico se actualizarán. Al iniciar el programa nuevamente (Con stock generado), El botón '**Initialize Factory**' no estará visible, ya que esta acción solo se puede hacer una vez. Se cargará un archivo serializado en el que sus registros serán cargados en la memoria del programa.

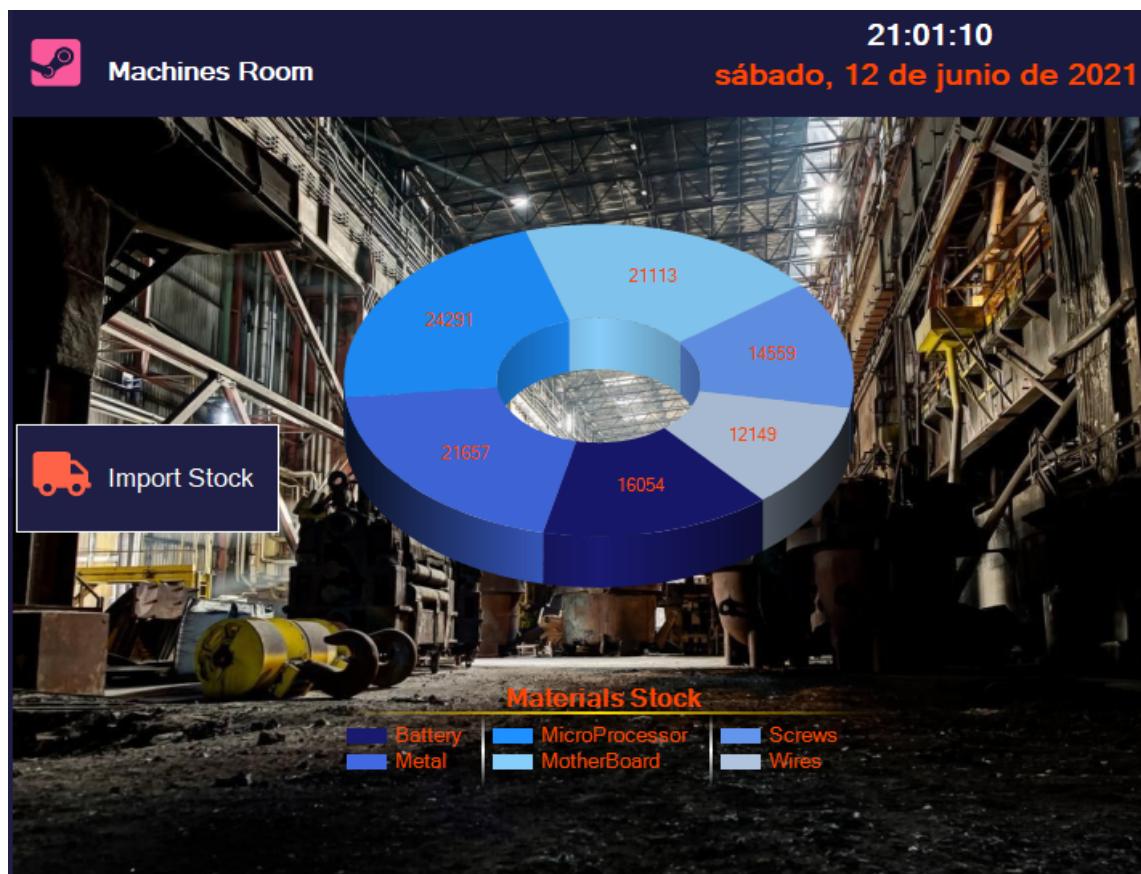


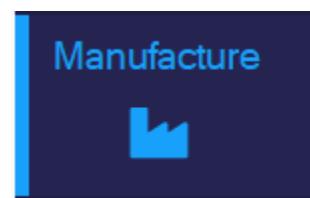
Machines Room

21:19:48

sábado, 12 de junio de 2021



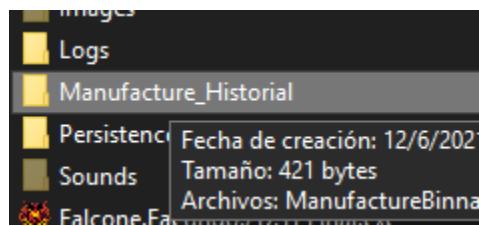




Manufacture: En el botón '**Manufacture**' podrán crear hasta 9 tipos distintos de robots (en caso de que la cantidad de materiales del almacén sean suficientes, caso contrario tirara una excepción mediante un formulario).



Por cada robot fabricado, se generará o re escribirá (en caso de existir) un archivo con la información del robot creado en el directorio  
root\Manufacture\_Historial\ManufactureBinnacle.txt



```
12/6/2021 23:30:09 - Product: T800
Manufacture Batch: 510-702-627
Manufacture Date: 12/6/2021 23:30:09
```

-----

```
12/6/2021 23:30:17 - Product: C3PO
Manufacture Batch: 548-789-873
Manufacture Date: 12/6/2021 23:30:17
```

-----

```
12/6/2021 23:30:25 - Product: WallE
Manufacture Batch: 509-838-852
Manufacture Date: 12/6/2021 23:30:25
```

-----



En el botón '**Warehouse**' se podrá ver el stock de la fábrica, tanto de materiales como de robots, como así también una breve información sobre cada robot en el almacén.



The screenshot shows a user interface for a warehouse system. At the top, there's a header with a shopping cart icon, the word "Warehouse", the time "22:24:19", and the date "sábado, 5 de junio". Below the header is a table titled "Robots" with the following data:

	Serial N°	Origin	Model	For Ride	Pieces
▶	8365	DigiWorld	Mugendramon	<input type="checkbox"/>	7
▶	55485	Cybertron	OptimusPrime	<input type="checkbox"/>	6

Below the table is a large image of Optimus Prime standing in a factory setting. To the right of the image is a detailed information panel with the following text:

Model: OptimusPrime  
Origin: Cybertron  
Ridable: False  
Pieces: 6

Biography: Optimus Prime is a Cybertronian, an alien species of self-configuring intelligent modular robotic life forms (ex: cars and other objects), a synergistic combination of biological evolution and technological engineering. In element all



El Logotipo ubicado en la esquina superior izquierda es un botón oculto que te lleva al 'home' de la app.

#### **Temas Utilizados para el programa:**

- *Excepciones*
- *Test Unitarios*
- *Tipos Genericos*
  - *Interfaces*
  - *Archivos*
- *Serialización*

#### **Excepciones:**

Implementado en varias secciones de los formularios para controlar posibles incidentes, por ejemplo una de las implementadas es:

```

14 referencias
public class InsufficientMaterialsException : Exception {

    #region Builders

        /// <summary>
        /// Creates an exception with the message.
        /// </summary>
        /// <param name="message">Message to show in the exception.</param>
        2 referencias
    public InsufficientMaterialsException(string message) : this(message, null) { }

        /// <summary>
        /// Creates an exception with the message and its innerException.
        /// </summary>
        /// <param name="message">Message to show in the exception.</param>
        /// <param name="innerException">InnerException of the exception.</param>
        1 referencia
    public InsufficientMaterialsException(string message, Exception innerException) : base(

```

La cual será lanzada al querer fabricar un robot sin tener la cantidad necesaria de materiales en:

TP3Final.GUI/Factory/Manufacture/frmManufacture/ConfigureRobotForBuild()

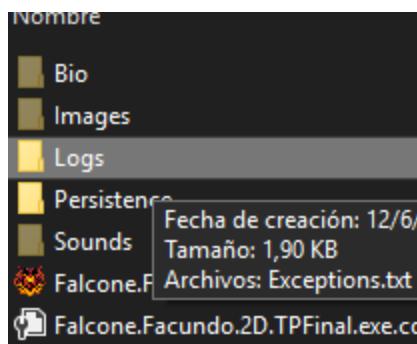
```

prototype = RobotFactory.CreateMultiplePiecesAmdAddToStock(metalType, origin, modelName, amountOfMaterials, an
filename = prototype.Model.ToString();
absBioPath = $"{pathBiography}{filename}.bin";
prototype.LoadBioFile(absBioPath);

if (RobotFactory.QualityControl(prototype, amountPieces)) {
    if (RobotFactory.AddRobotToWarehouse(prototype)) {
        MyPlayer.Play($"BuildingForm", false);
        frmLobby.FormShowDialogHandler(new frmBuilding());
        RobotFactory.SaveDataOfFactory();
        MyPlayer.Play($"Create{prototype.Model}", false);
        frmLobby.FormShowDialogHandler(new frmISOCertified(prototype.Model.ToString()));
    }
} else {
    if (RobotFactory.DissasembleRobot(prototype)) {
        throw new QualityControlFailedException("There have a difference in the amount of pieces, Quality cont
    }
}
prototype = null;
} else {
    throw new InsufficientMaterialsException("Insufficients Materials to build this model of robot, you need to i
}

```

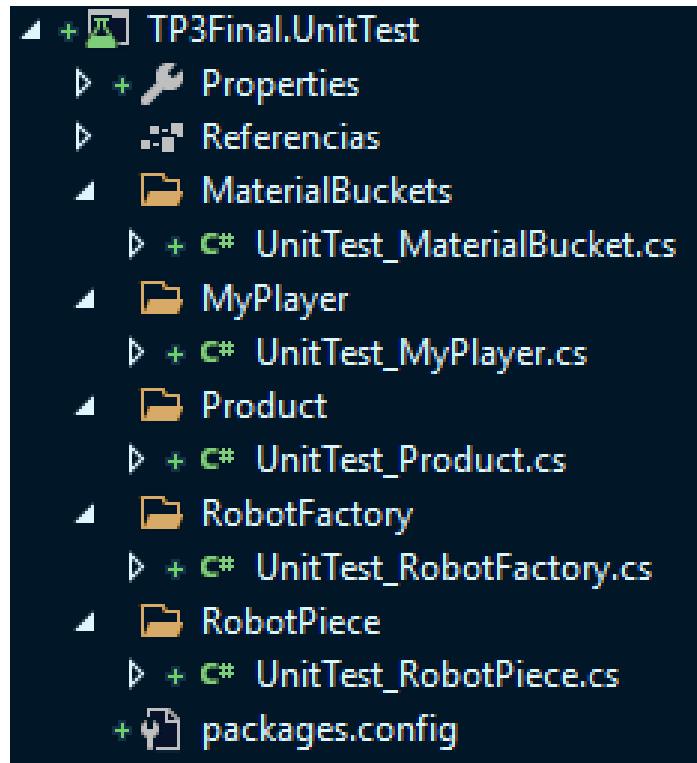
Por cada excepción lanzada, se creara o re escribirá (en caso de existir) un archivo con la información de la excepción en el directorio root\Logs\Exceptions.txt



```
12/6/2021 23:10:27 - System.NullReferenceException: Referencia a objeto no establecida como instancia de un objeto.  
  en FactoryForms.frmWarehouse.dgvRobots_CellContentClick(Object sender, DataGridViewEventArgs e) en E:\Git_UTN\GitHub\tp_laboratorio_2\TP_03_FacundoFalcone\Falcone.Facundo.2D.TPFinal\Factory\Warehouse\frmWarehouse.cs: línea 110  
-----  
12/6/2021 23:19:00 - System.NullReferenceException: Referencia a objeto no establecida como instancia de un objeto.  
  en FactoryForms.frmWarehouse.dgvRobots_CellContentClick(Object sender, DataGridViewEventArgs e) en E:\Git_UTN\GitHub\tp_laboratorio_2\TP_03_FacundoFalcone\Falcone.Facundo.2D.TPFinal\Factory\Warehouse\frmWarehouse.cs: línea 110  
-----  
12/6/2021 23:22:39 - System.NullReferenceException: Referencia a objeto no establecida como instancia de un objeto.  
  en FactoryForms.frmWarehouse.dgvRobots_CellContentClick(Object sender, DataGridViewEventArgs e) en E:\Git_UTN\GitHub\tp_laboratorio_2\TP_03_FacundoFalcone\Falcone.Facundo.2D.TPFinal\Factory\Warehouse\frmWarehouse.cs: línea 110  
-----
```

## Test Unitarios:

En el proyecto 'UnitTestProjectTPFinal' hay test unitarios para la mayoría de las clases, donde se testean métodos que hagan lo correspondiente.



## Tipos Genericos:

Implementados en interfaces por un tema de practicidad, se podrán encontrar en el proyecto  
bibliotecas de clases 'TP3Final.Models/Classes/Interfaces'

```
namespace Interfaces {
    2 referencias
    public interface IFilesManager<T> {
        6 referencias
        bool Save(string path, string filename, T dataToSave);
        4 referencias
        T Read(string filePath);
    }
}
```

Implementación en clase 'SerialManager<T>'.

```
8 referencias
public class SerialManager<T> : IFilesManager<T> {

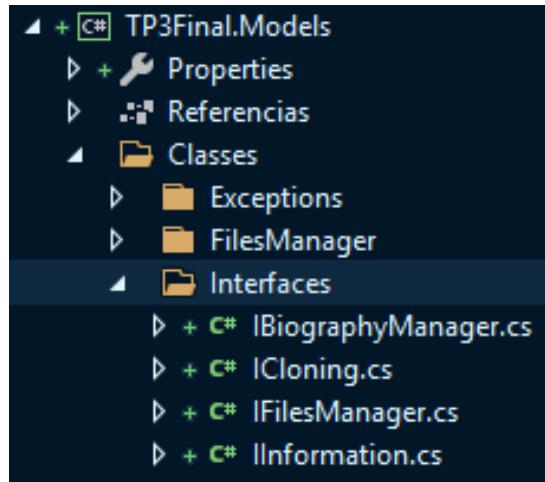
    #region Methods

    /// <summary>
    /// Reads the data from the path passed by parameter.
    /// </summary>
    /// <param name="filePath">Path to read the file.</param>
    /// <returns>True if can read the file, otherwise returns false.</returns>
    4 referencias
    public T Read(string filePath) {
        T aux;
        try {
            using (XmlTextReader reader = new XmlTextReader(filePath)) {
                XmlSerializer serial = new XmlSerializer(typeof(T));
                aux = (T)serial.Deserialize(reader);
            }
        } catch (Exception ex) {
            throw new Exception("Something get wrong trying reading Buckets", ex);
        }

        return aux;
    }
}
```

## Interfaces:

se podrán encontrar en el proyecto bibliotecas de clases 'TP3Final.Models/Classes/Interfaces'



La gran mayoría de las interfaces son genéricas, por ejemplo:

```
1 referencia
public interface ICloning<T>
    where T : RobotPiece {

    /// <summary>
    /// Clones the list passed by parameter and returns the clone.
    /// </summary>
    /// <param name="listToClone">List to clone.</param>
    /// <returns>The list cloned.</returns>
    2 referencias
    List<T> Clonelist(List<T> listToClone);
}
```

---

Implementadas en clases como ‘Robot’, interfaces con métodos que nada tienen que ver con la clase Robot pero que son útiles usarlos ya que proveen de funcionalidades extras que hacen a la clase.

```
43 referencias
public class Robot : IInformation, ICloning<RobotPiece>, IBiographyManager {

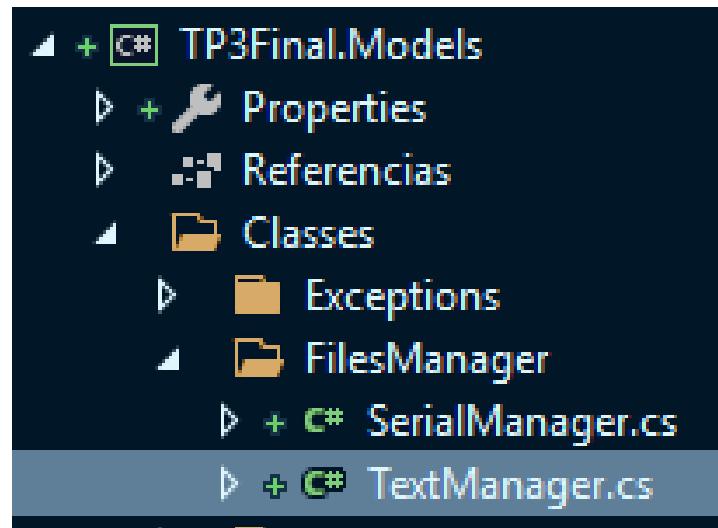
    #region Attributes

    private int serialNumber;
    private EOrigin origin;
    private EModelName modelName;
    private bool isRideable;
    private string biography;
    private List<RobotPiece> pieces;

    #endregion
```

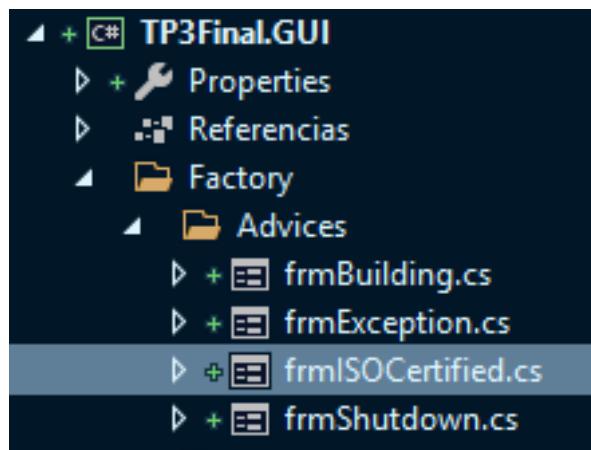
## Archivos:

Usados para escribir en txt el historial de fabricación de robots, como así también los 'logs' de las excepciones. Se implementa una interfaz en una clase de administrador de archivos



```
/// <summary>
/// Saves a stream into a file.
/// </summary>
/// <param name="path">Path to save the file.</param>
/// <param name="fileName">Name of the file to save in the path.</param>
/// <param name="dataToSave">Data to write into the file.</param>
/// <returns>True if can write the file, otherwise returns false.</returns>
6 referencias
public bool Save(string path, string filename, string dataToSave) {
    string absPath = $"{path}\\{filename}";
    StringBuilder data = new StringBuilder();
    data.AppendLine($"{DateTime.Now} - {dataToSave}\n-----");
    try {
        if (!Directory.Exists(path)) {
            Directory.CreateDirectory(path);
        }
        using (StreamWriter sw = File.AppendText($"{absPath}")) {
            sw.WriteLine(data);
            return true;
        }
    } catch (Exception ex) {
        throw new Exception("Something get wrong trying saving Document", ex);
    }
}
```

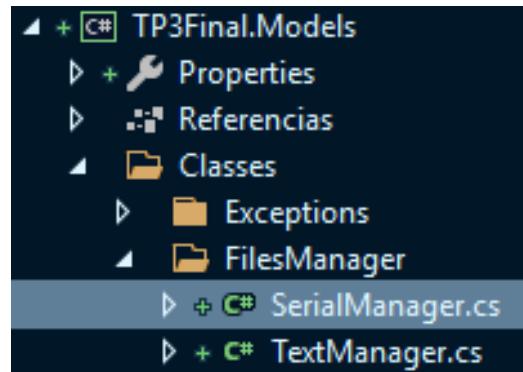
Usados en formularios al realizar ciertas acciones.



```
/// <summary>
/// EventHandler of the Load.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 referencia
private void frmISOCertified_Load(object sender, EventArgs e) {
    this.Opacity = 0.0;
    this.pbCertified.Value = 0;
    this.pbCertified.Minimum = 0;
    this.pbCertified.Maximum = 100;
    this.timeFadeIn.Start();
    ipbImage.BackgroundImageLayout = ImageLayout.Zoom;
    ipbImage.BackgroundImage = Image.FromFile($"{this.systemImagePath}\\"{this.productName}.png");
    warrantyMessage = WarrantyMessage();
    rtbWarrantyMessage.Text = warrantyMessage;
    logger.SaveFull(path, filename, warrantyMessage);
}
```

## Serialización:

Usados en Interfaces e implementado en la clase SerialManager, del tipo genérica, para serializar y deserializar archivos y listas.



```
/// <summary>
/// Reads the data from the path passed by parameter.
/// </summary>
/// <param name="filePath">Path to read the file.</param>
/// <returns>True if can read the file, otherwise returns false.</returns>
4 referencias
public T Read(string filePath) {
    T aux;
    try {
        using (XmlTextReader reader = new XmlTextReader(filePath)) {
            XmlSerializer serial = new XmlSerializer(typeof(T));
            aux = (T)serial.Deserialize(reader);
        }
    } catch (Exception ex) {
        throw new Exception("Something get wrong trying reading Buckets", ex);
    }

    return aux;
}
```

---

© Facu Falcone  
Berazategui, Buenos Aires, Argentina, 2021

