



# Córdoba Pablo - TP3

≡ Author	Pablo Miguel Córdoba - 2E
🕒 Created	@October 30, 2021 3:47 PM
✓ Reviewed	<input type="checkbox"/>

Disclaimer

Objetivo

Desarrollo

Wallet - Seguimiento de gastos e ingresos

Features alcanzadas

Features en desarrollo

Temas aplicados

Archivos

Interfaces

Generics

Excepciones

Capturas y estado actual de la aplicación

# Disclaimer

8/11/2021



Este proyecto no fue terminado en el tiempo esperado. Contiene únicamente las funcionalidades básicas.

## Objetivo

- Desarrollar funcionalidades para un sistema de análisis de datos:
  - A modo de ejemplo (no podrá ser utilizado por ningún alumno): ingreso los datos de encuestas de 2 candidatos a presidente y analizo en qué rango etario tienen más votos, donde menos, en qué sector de la sociedad, si es más votado por hombres, mujeres o no binarios, etc.
  - Se deberá poder agregar elementos diferentes para analizar y comparar.
  - Se deberá poder importar y exportar información, o consultar alguna fuente, o dar datos de entrada, o similar.
  - También se deberá poder cargar y exportar esa información a mano.
  - Darle el enfoque que crean correcto, teniendo en cuenta que el sistema tiene que permitir el análisis de datos.

## Desarrollo

### Wallet - Seguimiento de gastos e ingresos

Se realizó una aplicación que permite registrar movimientos monetarios con el fin de tener un registro de ellos a lo largo del tiempo.

## Features alcanzadas

- Levantar datos desde archivos.
- Añadir gastos e ingresos.
- Guardar toda la información del sistema en un archivo.
- Realizar informes en base a los datos del sistema.
- Exponer la información de los movimientos y billeteras.

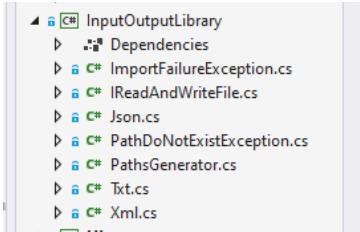
## Features en desarrollo

- Diseño de UI
- Logueo para acceder a la billetera propia, y no visualizar las de todos los usuarios.
- Seleccionar categorías para los movimientos (ej: indumentaria, alimentos, servicios, etc).
- Añadir categorías personalizadas.
- Informes relacionados únicamente con los datos de la billetera del usuario logueado.
- Planificación de gastos.
- Control y alertas de deudas.
- Exportar ticket de movimiento (serialización lista).

## Temas aplicados

### Archivos

Se creó una librería dedicada al manejo de archivos.



Contiene clases para todas las serializaciones vistas en clase.

Esta librería permite un manejo muy ágil de la escritura y lectura de archivos.

Por el momento no es 100% genérica debido a que se decidió restringir la posibilidad de importar o exportar archivos a los objetos que hereden de [DataEntity](#).

DataEntity es una clase abstracta que define los datos mínimos necesarios para que una entidad instanciable pueda ser controlada por la aplicación. Estos datos son [Id](#) y [TextInfo](#).

- Los *id* serán utilizados no solo como identificador del objeto dentro del sistema, sino que como nombre de los archivos.
- *TextInfo* permite que un objeto tenga información en formato string, independientemente de cómo se haya sobreescrito el método *ToString()*. Esto es debido a que *ToString()* es un método que se utiliza de un modo forzado en componentes como *listBoxes*, y no siempre se desea utilizar la misma información que se muestra por ejemplo en un *listBox*, para un archivo de texto que se crea a partir del mismo objeto.

## Interfaces

Se utilizó una interfaz genérica para la InputOutputLibrary.

Esta permitirá implementar los métodos para importar y exportar archivos a partir de objetos recibidos como parámetro.

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using Entities;
7
8 namespace InputOutputLibrary
9 {
10     public interface IReadAndWriteFile<T>
11     {
12         public void Export(string path, T entity);
13         public void Import(string path, out T entity);
14     }
15 }
16
```

## Generics

Además del uso de generics en la interfaz utilizada, se implementaron clases genéricas para todas las serializaciones programadas.

Contienen la restricción previamente explicada sobre DataEntity.

```
12  namespace InputOutputLibrary
13  {
14      public class Json<T> : IReadAndWriteFile<T> where T : DataEntity
15  {
16      public Json()
17      {
18          edit
19          public void Export(string path, T entity)
20          {
21              try
22              {
23                  if (!Directory.Exists(PathsGenerator<T>.JsonPath))
24                  {
25                      Directory.CreateDirectory(PathsGenerator<T>.JsonPath);
26                  }
27
28                  if (path != null)
29                  {
30                      JsonSerializerOptions options = new JsonSerializerOptions();
31                      options.WriteIndented = true;
32                      string serializedEntity = JsonSerializer.Serialize<T>(entity, options);
33                      File.WriteAllText(path, serializedEntity);
34                  }
35              }
36              catch
37              {
38                  throw new Exception("Error al exportar la entidad de ID: {entity.Id}.");
39              }
40          }
41      }
42
43      2 references
```

También se utilizó una clase genérica para crear los paths de los archivos

```

8
9  namespace InputOutputLibrary
10 {
11      public static class PathsGenerator<T> where T : DataEntity
12      {
13          private static string xmlPath;
14          private static string txtPath;
15          private static string jsonPath;
16
17          public static string XmlPath { get => xmlPath; }
18          public static string TxtPath { get => txtPath; }
19          public static string JsonPath { get => jsonPath; }
20
21          static PathsGenerator()
22          {
23              PathsGenerator<T>.xmlPath = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Desktop) + @"\Wallet\";
24              PathsGenerator<T>.txtPath = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Desktop) + @"\Wallet\";
25              PathsGenerator<T>.jsonPath = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Desktop) + @"\Wallet\";
26          }
27
28
29          public static string txtGeneratePath(T entity)
30          {
31              string fileName = PathsGenerator<T>.TxtPath + entity.Id.ToString() + ".txt";
32              return fileName;
33          }
34
35          public static string xmlGeneratePath(T entity)
36          {

```

143 % No issues found Ln: 1 Ch: 1 SPC CRLF

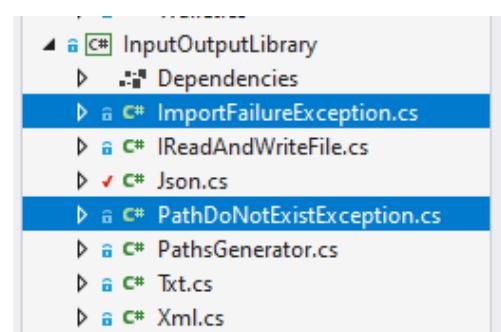
## Excepciones

Se utilizaron excepciones para los errores relacionados con la creación de archivos.

```

25
26  public void Import(string path, out T entity)
27  {
28      entity = default;
29      try
30      {
31          if (!Directory.Exists(PathsGenerator<T>.TxtPath) || !File.Exists(path))
32          {
33              throw new PathDoesNotExistException("No existe la carpeta o el archivo.");
34          }
35
36          string body = File.ReadAllText(path);
37
38          entity.TextInfo = body;
39      }
40      catch (Exception error)
41      {
42          throw new ImportFailureException($"Error al importar el archivo en: {path}.", error);
43      }
44  }
45
46  }
47

```



## Capturas y estado actual de la aplicación



# Incompleto

[Nuevo ingreso](#)[Nuevo gasto](#)[Guardar](#)

Usuario:

**Pablo**

Total de movimientos

**24****X**

Saldo:

**\$ 43459**

Cantidad de movimientos el día de hoy

**16**

Usuarios

2 - Pablo  
3 - Juana  
4 - Jose  
5 - Pepe  
6 - Ramiro  
7 - Luciana

Movimientos

Fecha	Monto	Tipo
07/11/2021 22:22:30	\$100	Gasto
07/11/2021 20:14:25	\$800	Gasto
07/11/2021 20:13:40	\$800	Gasto
07/11/2021 20:11:58	\$565	Gasto
04/11/2021 20:12:02	\$76	Gasto

El/los usuario/s con el movimiento de más dinero

**3 - Juana 4 - Jose**

El/los usuario/s con más movimientos

**2 - Pablo 7 - Luciana**

El mayor ingreso registrado

**\$10000**

## Nuevo movimiento

Saldo actual

\$ 43459

Saldo después del movimiento

\$ 42959

Fecha del movimiento

08 November 2021



Monto

500



Aceptar

Cancelar