



Córdoba Pablo - TP4

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

IMPORTANTE - Instrucciones para correr la app

Script de la base de datos

Objetivo

Desarrollo

Wallet - Seguimiento de gastos e ingresos

Features alcanzadas

Temas aplicados

Base de datos

Archivos

Interfaces

Generics

Excepciones

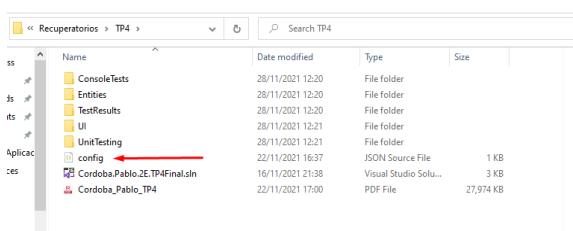
Task

Métodos de extensión

Eventos

Capturas y estado actual de la aplicación

IMPORTANTE - Instrucciones para correr la app



La connection string será obtenida del archivo "config.json" ubicado al mismo nivel que la solución.

Por favor modificar este archivo con la connection string correspondiente al equipo.

```
config.json
C: > Users > PabloCordoba > Desktop > Developing > Repos TP > tps_laboratorio_ii > Recuperatorios > TP4 > config.json > ...
1 {
2   "ConnectionString": "Data Source=BRS0003\\SQLEXPRESS; Initial Catalog=CordobaPablo-TP4; Trusted_Connection=True"
3 }
4
5
6
7
```

Script de la base de datos

Puede encontrar el script de la base de datos en el siguiente enlace:

<https://onlinegdb.com/qhcl1AdUzH>

GDB online Debugger | Code, Compile, Run, Debug online C, C++

Online GDB is online ide with compiler and debugger for C/C++. Code, Compiler, Run, Debug Share code snippets.

 <https://onlinegdb.com/qhcl1AdUzH>

O en la carpeta raíz del proyecto

Name	Date modified	Type	Size
ConsoleTests	28/11/2021 12:20	File folder	
Entities	28/11/2021 12:20	File folder	
TestResults	28/11/2021 12:20	File folder	
UI	28/11/2021 12:21	File folder	
UnitTesting	28/11/2021 12:21	File folder	
config	22/11/2021 16:37	JSON Source File	1 KB
Cordoba.Pablo.2E.TP4Final.sln	16/11/2021 21:38	Visual Studio Solu...	3 KB
Cordoba_Pablo_TP4	22/11/2021 17:00	PDF File	27,974 KB
DB-Script	28/11/2021 18:25	SQL Source File	45 KB



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

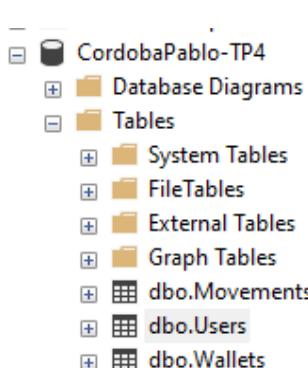
Se realizó una aplicación que permite registrar movimientos monetarios con el fin de tener un registro de ellos a lo largo del tiempo.

Features alcanzadas

- Leer y escribir datos en base de datos.
- Añadir gastos e ingresos.
- Realizar informes estadísticos a partir de los datos del sistema.
- Exponer la información de los movimientos y billeteras.
- 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).
- Informes relacionados únicamente con los datos de la billetera del usuario logueado.

Temas aplicados

Base de datos

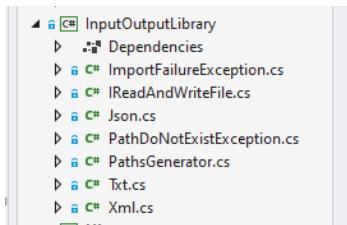


Se creó una base de datos con registros de ejemplo creados desde la api Mockaroo.

Todas las interacciones con la base de datos se realizaron desde una clase estática llamada DbService.

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.

Además, se utilizó un archivo para obtener la connection string.

```

159 // <summary>
160 // Importa el archivo de configuración del proyecto que contiene la connection string.
161 // Este archivo debe estar al mismo nivel que la solución.
162 // </summary>
163 
164 public static void GetProjectConfig()
165 {
166     ProjectConfigurationData config = new ProjectConfigurationData("testando");
167 
168     JsonProjectConfigurationData configDataSerializer = new JsonProjectConfigurationData();
169 
170     configDataSerializer.Import(@"..\..\..\..\", "config.json", ref config);
171 
172     Core.ConfigData = config;
173 }
174 }
175

```

Name	Date modified	Type	Size
ConsoleTests	21/11/2021 23:19	File folder	
Entities	19/11/2021 00:34	File folder	
TestResults	22/11/2021 15:48	File folder	
UI	22/11/2021 14:32	File folder	
UnitTesting	22/11/2021 15:48	File folder	
config	17/11/2021 23:11	JSON Source File	1 KB
Cordoba.Pablo.IETP4final.sln	16/11/2021 21:38	Visual Studio Solut...	3 KB
Cordoba-Pablo-2E	08/11/2021 00:24	PDF File	24,756 KB

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
13 namespace InputOutputLibrary
14 {
15     public class Json<T> : IReadAndWriteFile<T> where T : DataEntity
16     {
17         public Json()
18         {
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 }
```

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



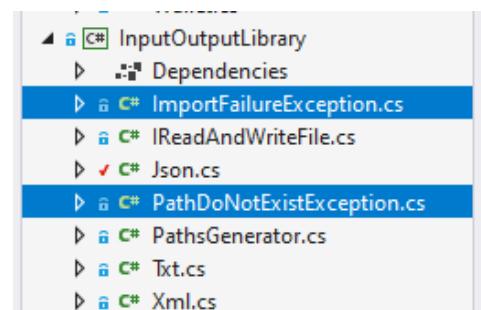
```
8
9     namespace InputOutputLibrary
10    {
11        public static class PathsGenerator<T> where T : DataEntity
12        {
13            private static string xmlPath;
14            private static string txtPath;
15            private static string jsonPath;
16
17            public static string 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) + @"\Wallets\", "PathsGenerator.xml");
24                PathsGenerator<T>.txtPath = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Desktop) + @"\Wallets\", "PathsGenerator.txt");
25                PathsGenerator<T>.jsonPath = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Desktop) + @"\Wallets\", "PathsGenerator.json");
26            }
27
28            public static string txtGeneratePath(T entity)
29            {
30                string fileName = PathsGenerator<T>.TxtPath + entity.Id.ToString() + ".txt";
31                return fileName;
32            }
33
34            public static string xmlGeneratePath(T entity)
35            {
36            }
37
38
39
40
41
42
43
44
45
46
47 }
```

Excepciones

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



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



También se utilizaron excepciones para errores al registrarse o loguearse.

En Entities.SystemCore.Core

```
106
107     /// <summary>
108     /// Recibe los datos de un usuario y los busca en la base de datos.
109     /// </summary>
110     /// <param name="name"></param>
111     /// <param name="password"></param>
112     /// Si puede loguear, establece el usuario actual y su wallet con los datos del usuario encontrado.
113     /// Si no puede, arroja una excepción del tipo InvalidUserException.
114     1 reference
115     public static void LogIn(string name, string password)
116     {
117         Core.ActualUser = null;
118         Core.UserWallet = null;
119
120         Core.ActualUser = DbService.GetUserByCredentials(name, password);
121         if (Core.ActualUser != null && ActualUser.IdWallet > -1)
122         {
123             Core.UserWallet = DbService.GetWalletById(Core.ActualUser.IdWallet);
124             if (Core.UserWallet != null && UserWallet.Id > -1)
125             {
126                 Core.UserWallet.MoneyMovements = DbService.GetMovements(Core.UserWallet.Id);
127             }
128         }
129         else
130         {
131             throw new InvalidUserException();
132         }
133
134     /// <summary>
135     /// Comprueba que no exista un usuario con el mismo nombre recibido por parámetro,
136     /// y si no lo encuentra, lo crea y lo añade a la base de datos.
137     /// </summary>
138     /// <param name="nombreDeUsuario"></param>
139     /// <param name="password"></param>
140     /// Si lo encuentra, arroja una excepción del tipo InvalidUserException
141     1 reference
142     public static void SignIn(string name, string password)
143     {
144         User newUser = DbService.GetUserByName(name);
145         if (newUser != null && newUser.IdWallet == -1 && newUser.Id == -1)
146         {
147
148             Wallet newWallet = new Wallet(0);
149             newUser = new User(name, password);
150             newUser.IdWallet = newWallet.Id;
151
152             DbService.UpdateUserWalletId(newUser);
153         }
154         else
155         {
156             throw new InvalidUserException();
157         }
158     }
```

Task

Se utilizó task para la carga de las estadísticas. Para simular un tiempo de procesamiento, se añadió un delay de 5 segundos, en los cuales el usuario visualizará la pantalla en estado de carga, pero podrá seguir utilizando la aplicación.

```
--  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
  
    1 reference  
public Task LoadCategoriesStats()  
{  
    return Task.Run(() =>  
    {  
        this.ResetDefaultText();  
        int index = 0;  
        int x = 0;  
        int y = 0;  
        float monthTotal = 0;  
        float previousMonthTotal = 0;  
        float percentage = 0;  
        bool nowIsMore;  
        bool initialValueIsNull;  
  
        Thread.Sleep(5000);  
        if (pnlStats.InvokeRequired)  
        {  
            pnlStats.BeginInvoke((MethodInvoker)delegate()  
            {  
                pnlStats.Controls.Clear();  
                Bank.ExpensesCategories.ForEach(category =>  
                {  
                    if (Core.UserWallet != null)  
                    {  
                        monthTotal = Core.UserWallet.getMonthTotal(DateTime.Now, Movement.eType.Gasto, category);  
                        percentage = Core.UserWallet.CompareNowWithPreviousMonth(Movement.eType.Gasto, category, out nowIsMore, out initialValueIsNull);  
                        previousMonthTotal = Core.UserWallet.getMonthTotal(DateTime.Now.AddMonths(-1), Movement.eType.Gasto, category);  
                        pnlStats.Controls.Add(new ucStat(category, monthTotal, previousMonthTotal, percentage, nowIsMore, initialValueIsNull));  
                    }  
                });  
            });  
        }  
    });  
}
```

3 references

```
public async void LoadStats()  
{  
    await LoadCategoriesStats();  
    LoadHeaderValues();  
}
```

Se utilizó async await para que primero se carguen las estadísticas de las categorías, y luego se carguen las generales del mes.

Métodos de extensión

Se extendió la clase DataEntity con un método que recibe una cadena de caracteres, y un delegado que funcionará como data accesor para acceder a campos de la entidad. El método busca la cadena en la propiedad de la entidad y devuelve un valor booleano que representa si la encuentra o no.

```
L:\Users\pc\Documents\Visual Studio 2019\Projects\...\\
...
namespace Entities.SystemCore
{
    0 references
    public static class ExtensionMethodsClass
    {
        2 references | ✓ 1/1 passing
        public static bool SearchStringOnEntity(this DataEntity entity, string searchString, Func<DataEntity, string> dataAcessor)
        {
            if (dataAcessor(entity).Contains(searchString))
            {
                return true;
            }
            return false;
        }
    }
}
```

Eventos

Se utilizaron eventos para notificar a los demás formularios que ya están abiertos, que se ha añadido un nuevo movimiento, para que rendericen su información nuevamente.

También para realizar acciones similares cuando el usuario cierra o inicia sesión.

```
17
18     namespace UI
19     {
20         6 references
21         public partial class frmMovements : Form
22         {
23             public event EventHandler newMovement;
24
25             1 reference
26             private void frmMainDashboard_Load(object sender, EventArgs e)
27             {
28                 movements.newMovement += onNewMovement;
29                 this.logIn += onLogIn;
30                 this.logOut += onLogOut;
```



The screenshot shows a code editor window with the file "frmMainDashboard.cs" open. The code is written in C# and defines a class "UI" with several methods:

```
64     lblNumActualBalance.Text = "$00,000.00";
65     lblUserName.Text = "Nombre";
66     lblNumWalletId.Text = "";
67     lblNumTotalExpenses.Text = "$00,000.00";
68     lblNumTotalIncomes.Text = "$00,000.00";
69 }
70
71     private void onNewMovement(object sender, EventArgs e)
72     {
73         LoadUserData();
74     }
75
76     private void onLogIn(object sender, EventArgs e)
77     {
78         stats.LoadStats();
79         movements.LoadWalletData();
80         this.LoadUserData();
81     }
82
83     private void onLogOut(object sender, EventArgs e)
84     {
85         this.IsLogged(false);
86         stats.ResetDefaultText();
87     }
88 }
```

The code includes comments indicating references to other parts of the application: "LoadUserData()", "stats.LoadStats()", "movements.LoadWalletData()", and "this.LoadUserData()".

Capturas y estado actual de la aplicación

Pablo
Wallet ID: 1

Saldo actual	Ingresos totales	Gastos totales
\$46,803.00	\$323,620.00	\$415,201.40

Movimientos

Fecha	Monto	Tipo	Categoría
21/11/2021 00:00:00	15642	Gasto	Varios
21/11/2021 00:00:00	20000	Gasto	Varios
19/11/2021 00:00:00	800	Gasto	Entretenimiento
15/11/2021 00:00:00	23149.92	Ingreso	Entretenimiento
14/11/2021 00:00:00	17007.59	Gasto	Servicios

+ Cerrar sesión

Pablo
Wallet ID: 1

Saldo actual	Ingresos totales	Gastos totales
\$46,803.00	\$323,620.00	\$415,201.40

Movimientos

Mes	Ingresos	Gastos
November	82.83%	57.04%
Estadísticas	menos que el mes pasado	menos que el mes pasado
Varios \$50,080.52	63.72% menos que el mes pasado	57.04% menos que el mes pasado
Indumentaria \$7,603.22	84.44% menos que el mes pasado	57.04% menos que el mes pasado

+ Cerrar sesión

Nuevo movimiento

Saldo actual

\$ 43459

Saldo después del movimiento

\$ 42959

Fecha del movimiento

08 November 2021



Monto

500



Aceptar

Cancelar