

```
_={};function F(e){var t=_[e]={};return b.each(t[1])==!=1&&e.stopOnFalse){r=!1;break}n!=1,u&=o=u.length:r&&(s=t,c(r))}return this},remove:nction(){return u=[],this},disable:function(){re:function(){return p.fireWith(this,argument)}ending",r={state:function(){return n}},always:promise)?e.promise().done(n.resolve).fail(n.reject(function(){n=s}),t[1^e][2].disable,t[2][2].n=0,n=h.call(arguments),r=n.length,i=1!=r||e&(r),l=Array(r);r>t;t++)n[t]&&bisFunction(n[t])</table><a href='/a'>a</a><input type='button' value='B' />
```



IGU- Práctica Final - 70926454C

Ángel Picado Cuadrado - angelpiccua@usal.es - PB2

Índice

1. Resumen
 2. Metodología
 - a. Manual del desarrollador
 - b. Manual del usuario
 3. Bibliografía
-

Resumen

La finalidad de este trabajo es la realización de una aplicación de escritorio con interfaz gráfica que sea capaz de llevar un conteo de calorías en las distintas ingestas que se producen a lo largo de un día y de graficarlo mediante gráficas de barras de formas distintas.

Las herramientas utilizadas son las siguientes:

- Para la codificación se ha utilizado el lenguaje C# y XAML
- Como IDE se ha utilizado Microsoft Visual Studio
- Para la obtención de la paleta de colores se ha usado la herramienta [colorso.co](#)
- Para exportar e importar el modelo de los datos en formato JSON se ha usado la herramienta [Newtonsoft.NET](#)

La mayor dificultad del trabajo ha sido el diseño de un modelo capaz de graficar los datos introducidos por el usuario de las dos maneras requeridas (visión detallada y visión general)

Metodología

Tras haber leido el enunciado varias veces y haber comprendido la finalidad del proyecto procedí a realizar en papel de manera esquemática como quería que fuera el diseño de las ventanas del proyecto

- Ventana Principal — Encargada de mostrar los gráficos (Tabla General y Tabla Diaria)
- Ventana TablaDatos — Encargada de mostrar las tablas, así como importar o exportar los datos
- Ventana AddDatos — Encargada de la introducción o edición de datos

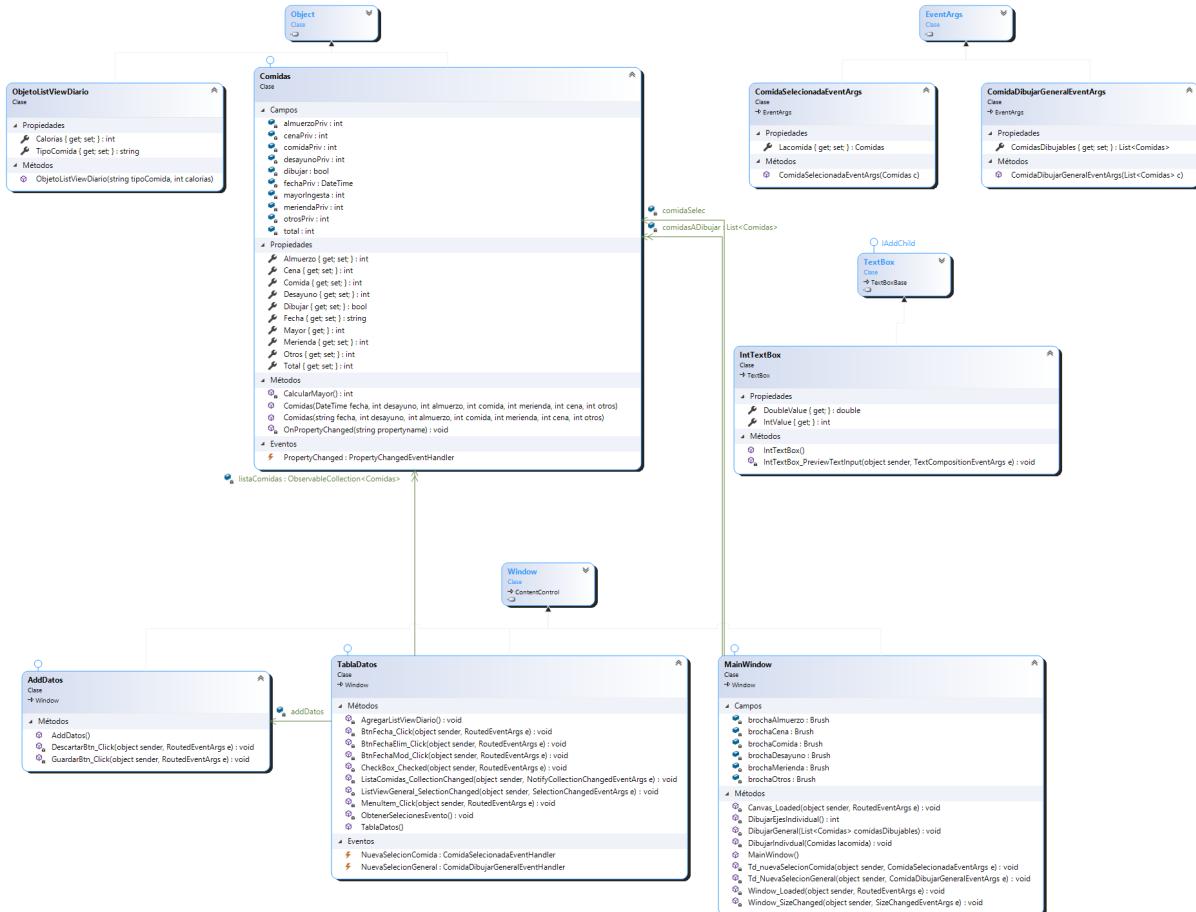
Para el diseño visual de la interfaz se ha decidido que sea de tonos claros con colores no muy fuertes y con una interfaz agradable para que pueda ser usada por todo tipo de usuarios.

En cuanto al icono de las ventanas se ha sacado de una página de imágenes vectoriales gratuita

Manual del Desarrollador

El patrón de diseño utilizado para este proyecto ha sido el patrón ModelViewViewModel (MVVM) en lo máximo de lo posible.

Se ha realizado un diagrama de clases con la opción que proporciona Visual Studio



Paleta de colores usada

Como se ha especificado anteriormente se ha usado la herramienta coolors.co para crear una paleta de colores para los gráficos

					
Desayuno	Almuerzo	Comida			
HEX RGB HSB CMYK NAME	6889DE 104, 185, 222 199, 53, 87 53, 16, 0, 12 Cyan Process	HEX RGB HSB CMYK NAME	219EBC 33, 158, 188 192, 82, 74 82, 15, 0, 26 Blue Green	HEX RGB HSB CMYK NAME	023047 2, 48, 71 200, 97, 28 97, 32, 0, 72 Prussian Blue
					
Merienda	Cena	Otros			
HEX RGB HSB CMYK NAME	FEB703 254, 183, 3 43, 99, 100 0, 27, 98, 0 Selective Yellow	HEX RGB HSB CMYK NAME	FB8500 251, 133, 0 32, 100, 98 0, 47, 100, 1 Tangerine	HEX RGB HSB CMYK NAME	8D4C02 141, 76, 2 32, 99, 55 0, 46, 98, 44 Brown

Paleta de colores de los gráficos

Aquí está la paleta de colores usada en distintos formatos

COOLORS

Colors - coolors.co

Descripción Clases:

- Clase **Comidas** : Esta clase sirve de modelo de los datos, en ella se almacenan todos los datos de las ingestas, la fecha y un bool que indica si se va a dibujar o no en la gráfica general de la ventana MainWindows. Esta clase hereda de **INotifyPropertyChanged**
 - Esta clase tiene 2 constructores, uno que se usa para la introducción de datos a través de la ventana AddDatos y otro que se usa para la introducción de datos a través de un archivo JSON (**[JsonConstructor]**)
 - Además de las ingestas el constructor calcula la mayor ingesta que será utilizada en la representación diaria
 - Cuando se produce un cambio en una propiedad se manda un evento **PropertyChanged**
- Clase **IntTextBox** : Esta clase sirve para asegurarse que los datos introducidos como calorías son positivos y enteros, esta clase hereda de **TextBox**
- Clase **ObjetoListViewDiario** : Esta clase sirve para crear la ListView detallada con la fecha que se ha seleccionado en la ListView general, tiene 2 campos: el

tipo de ingesta y las calorías, cuando se selecciona una fecha se llena con las calorías por ingesta

- Clases **EventArgs** : Son aquellas clases (ComidaSeleccionadaEventArgs y ComidaDibujarGeneralEventArgs) son aquellas encargadas de enviar a través de un evento la información de la comida desde la ventana TablaDatos a la MainWindows
- Clase **AddDatos** : Esta clase se encarga de recoger los datos que introduce el usuario a la hora de modificar o introducir una fecha
- Clase **TablaDatos** : Es una de las clases más importantes ya que gestiona la visualización de datos en las dos ListViews especificadas en el enunciado de la práctica
 - Esta clase se encarga de la introducción de datos a través de un botón que invoca una instancia de la clase AddDatos, esa misma ventana se usa para la modificación de datos a través de otro botón
 - Esta clase tiene un menú con 3 funciones principales:
 - La función de exportar los datos —> Esta operación se realiza creando un cuadro de dialogo de guardado de datos en el que luego se usa la biblioteca de <https://github.com/JamesNK/Newtonsoft.Json> que serializa el ObservableCollection <Comidas> que contiene todos los datos. Es importante que la extensión del archivo siempre va a ser .contCal
 - La función de importar los datos —> Esta operación se realiza creando un cuadro de dialogo de apertura de archivos que solo abre archivos con la extensión del programa (.contCal) para evitar problemas. Luego se usa la biblioteca de <https://github.com/JamesNK/Newtonsoft.Json> que deserializa el archivo y obtiene una List<Comidas> y luego los elemertos contenidos en la List se añaden a la ObservableCollection <Comidas> que contiene los datos.
 - La función de vaciado de datos —> Esta operación borra todos los registros de la ObservableCollection <Comidas>, se muestra un cuadro de dialogo para confirmar el borrado
 - Esta clase se encarga del eliminado de registros a través de un botón que se activa cuando un elemento de la ListView general está seleccionado, se muestra un cuadro de dialogo para confirmar el borrado

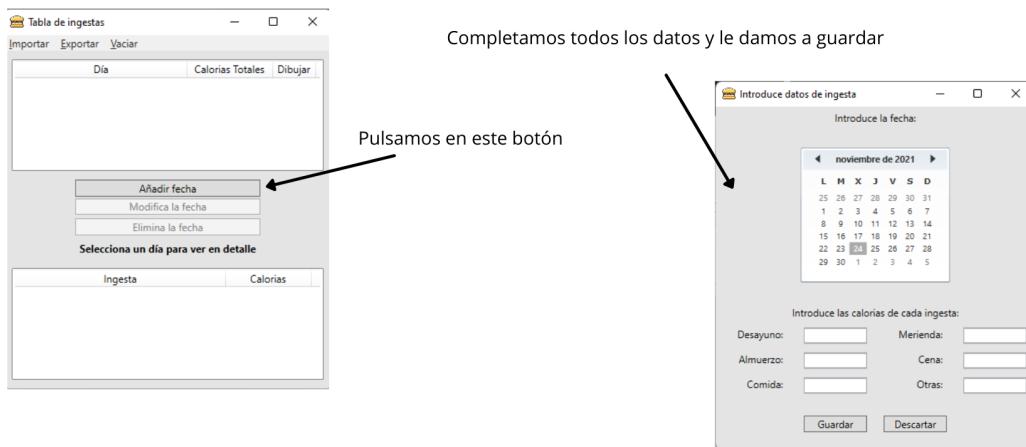
- Cuando se produce una selección en la ListView general se crean las instancias de ObjetoListViewDiario y se añaden al ListView diario y luego se manda un evento al MainWindows con el ComidaSeleccionadaEventArgs para que se dibuje
- Cuando se produce un cambio en el checkbox Dibujar en la ListView general se crea una List<Comidas> con los objetos del ObservableCollection <Comidas> que tienen el checkbox activo y se manda un evento al MainWindows con el ComidaDibujarGeneralEventArgs para que se dibujen
- Tambien se controla para que cuando se modifiquen, borren o añadan los datos se manden los determinados eventos al MainWindows.
- Clase **MainWindows** : Es la clase principal, la encargada de hacer los gráficos de los datos. Cuando esta ventana se ha cargado crea una instancia de TablaDatos y registra los eventos que le pueden llegar.
 - Esta ventana se ha creado con dos tabsItems que tiene un canvas cada uno que va a graficar lo que le corresponda, esto se ha hecho para que la experiencia de usuario sea más satisfactoria al poder visualizar la tabla general y la diaria con más facilidad. Además se ha añadido un desplegable que indica la leyenda de los gráficos para su mejor comprensión
 - Cuando llega un evento de selección de una fecha primero se comprueba que se está en el tab de la visión diaria y luego si lo recibido es un null se procede a limpiar el canvas en caso de ser distinto se procede al dibujado
 - Para dibujar primero se dibujan los ejes, colocando los ejes y las etiquetas de los nombres de las ingestas, y se calcula el ancho que debe ocupar cada rectángulo. Después se obtiene cual es la ingesta mayor, la cual se toma como referencia ya que esta ocupara la totalidad del eje Y y el resto de ingestas se dibujan de manera proporcional.
 - Cuando llega un evento que produce el dibujado general primero se comprueba que la List recibida tiene datos, en caso contrario no hay ningún elemento que se quiera dibujar por lo tanto se limpia el canvas, en caso contrario se procede a su dibujado
 - Al igual que en otro canvas primero se dibujan los ejes , luego calculamos cual es la Comida que tiene un mayor total, esa es la que tomamos como referencia, a partir de ahí creamos varias etiquetas a lo largo del eje Y que nos sirven de guia. Como en el otro dibujado los

rectangulos se crean en proporción al mayor, la mayor ingesta ocupa todo el eje Y y el resto se dibujan en proporción. Finalmente se coloca la etiqueta de la fecha

- En esta clase tambien se controla el resize que se puede realizar adaptando las gráficas al mismo.

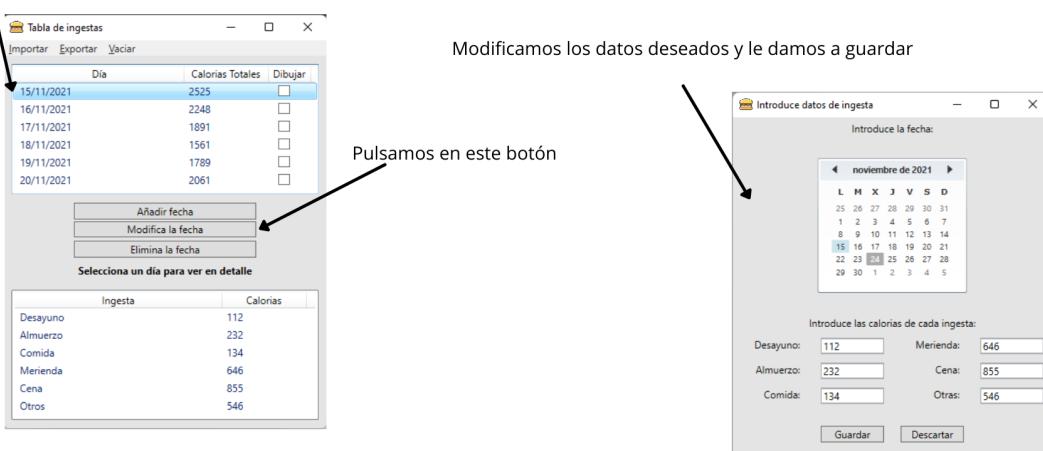
Manual del Usuario

Agregar ingestas

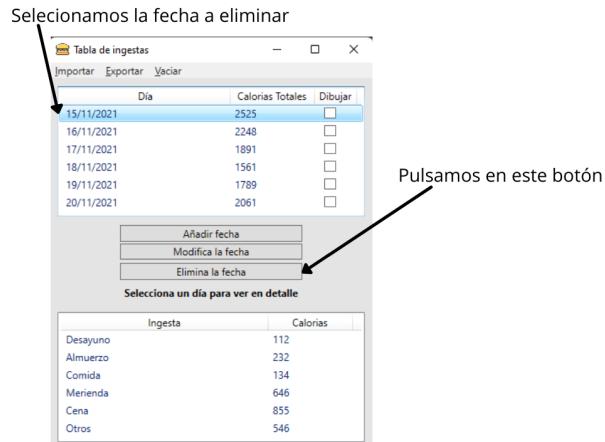


Modificar ingestas

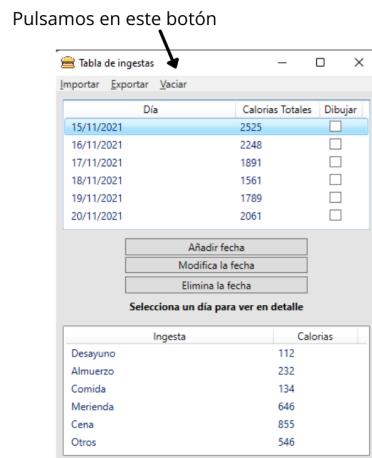
Seleccionamos la fecha a modificar



Eliminar ingestas

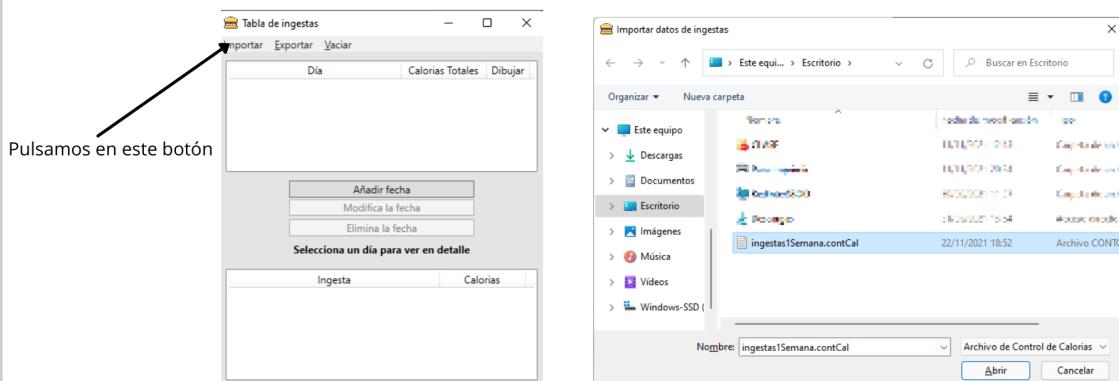


Vaciar todas las ingestas



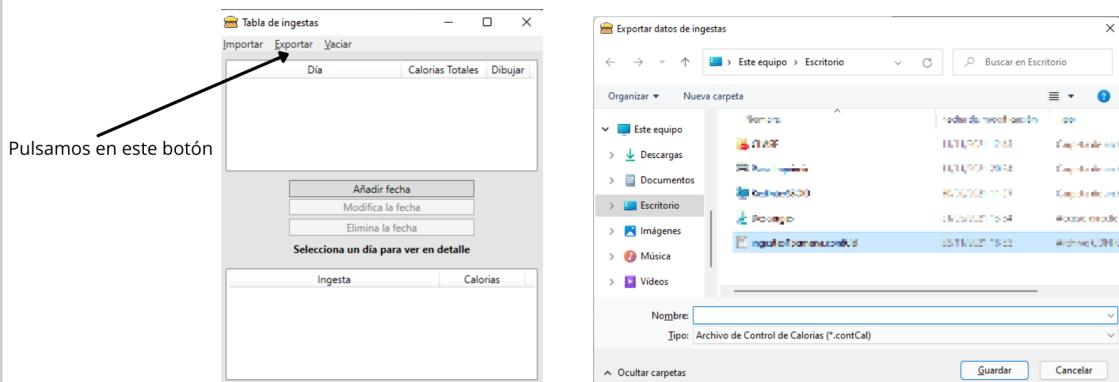
Importar datos

Seleccionamos el archivo .contCal deseado y le damos a **Abrir**



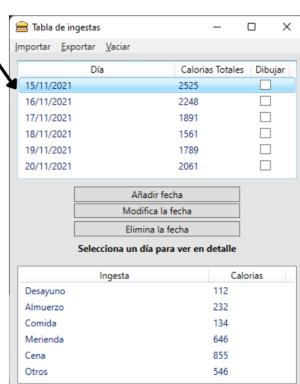
Exportar datos

Navegamos hasta el destino, establecemos un nombre al archivo .contCal y le damos a **Guardar**

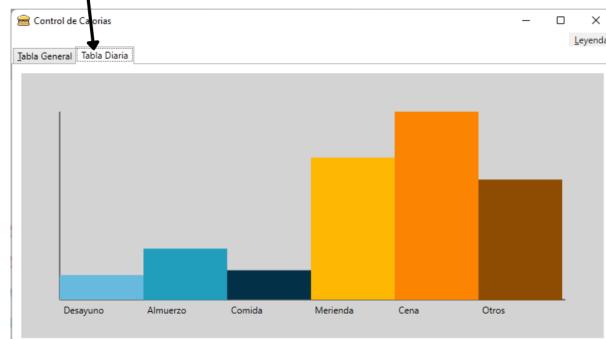


Visualizar datos diarios

Seleccionamos el día que queremos visualizar en la tabla superior

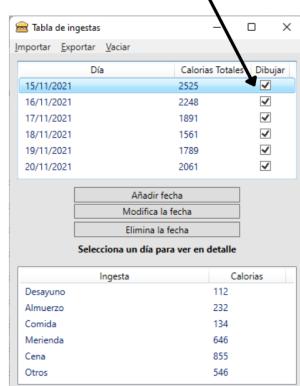


Elegimos que queremos ver la vista diaria

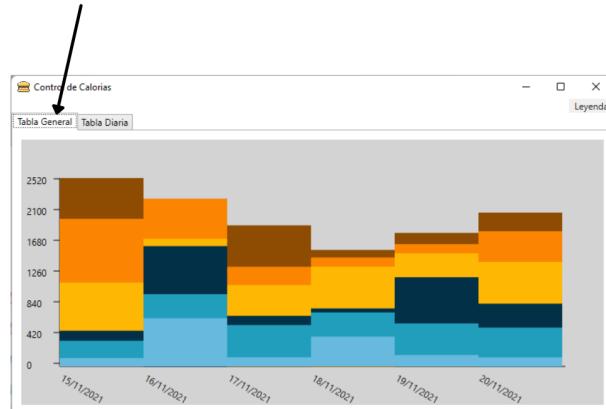


Visualizar datos generales

Seleccionamos con el checkbox los días que queremos visualizar en el gráfico



Elegimos que queremos ver la vista general



Bibliografía

- Como colocar elementos en un canvas —><https://docs.microsoft.com/es-es/dotnet/desktop/wpf/controls/how-to-create-and-use-a-canvas?view=netframeworkdesktop-4.8>
- Añadir checkbox a un listview —><https://docs.microsoft.com/es-es/dotnet/api/system.windows.forms.listview.checkboxes?view=windowsdesktop-6.0>
- Serializar y deserializar objetos en C# a JSON —><https://www.newtonsoft.com/json/help/html/Samples.htm>
- Crear cuadros de diálogo de guardado C# —><https://wpf-tutorial.com/es/47/dialogos/el-dialogo-de-guardado-savefiledialog/>
- Crear cuadros de diálogo de apertura —><https://wpf-tutorial.com/es/46/dialogos/el-dialogo-openfiledialog/>
- Clase Calendar C# —><https://docs.microsoft.com/es-es/dotnet/api/system.globalization.calendar?view=net-6.0>