

Documentación correspondiente a la entrega del 4° Trabajo Practico de Programación / Laboratorio II

Alumno: Sinnott Segura, Gonzalo.

En el presente se pretende demostrar la implementación de un sistema de gestión de partes para el armado de guitarras y también el armado del producto final y generación de reportes. El mismo cuenta con 3 vistas, las cuales se detallaran a continuación, junto con el funcionamiento de las mismas.

INDICE:

MANUAL DEL USUARIO: Pag 1

IMPLEMENTACION DE LOS TEMAS VISTOS EN CLASE: Pag 4

DOCUMENTACION: Pag 5

DIAGRAMA DE CLASES: Pag 9.

MANUAL DE USUARIO:

-VISTA STOCK:

En esta vista el usuario puede administrar el stock de partes para el armado del producto. Al iniciar el programa el mismo carga por defecto la base de datos **Stock** y refleja su contenido en los datagridviews correspondientes. Caso contrario lanzara una excepción y no se podrá continuar la ejecución del programa. **VER CONNECTION STRING DE LA CLASE DAO EN LA BIBLIOTECA ENTIDADES SI SE TIENE LA BASE DE DATOS Y EL ERROR PERSISTE.**

GUITAR CUSTOM SHOP

STOCK ENSAMBLAJE INFORMES

PIEZA:

TIPO:

FABRICANTE:

AGREGAR

Id	ClassType	Type	Manufacturer	EntryDate
1	Tuners	N/A	Gotoh	11.07.21
2	Electronics	Gibson	N/A	11.07.21
3	Wood	Nogal	N/A	11.07.21
4	Pickup	Single	DiMarzio	11.07.21
5	Tuners	N/A	Grover	11.07.21
6	Electronics	Fender	N/A	11.07.21
7	Wood	Nogal	N/A	11.07.21
8	Pickup	P90	EMG	11.07.21

ELIMINAR

- 1) En esta sección el usuario ingresa la pieza a registrar con todas sus características, y al presionar el botón AGREGAR, el ingreso se verá Reflejado en el DataGridView de la derecha y se agregara a la tabla Piezas de la base de datos Stock. Si no se eligen las características de la pieza, la misma no se agregara y una ventana emergente indicara al usuario que faltan elegir características de la pieza a agregar.
- 2) En esta sección el usuario tiene la vista del stock de partes con sus características,
- 3) En esta sección al presionar el botón ELIMINAR, se eliminara de la base de datos la pieza seleccionada del datagridview y se vera reflejada esta acción en dicho datagridview. Si no se elige una pieza y se presiona el botón, una ventana emergente indicara al usuario que para usar esta característica debe elegir una pieza a eliminar.

VISTA ENSAMBLAJE:

En esta vista el usuario puede, a partir del stock de partes disponibles armar productos

GUITAR CUSTOM SHOP

STOCK **ENSAMBLAJE** INFORMES

MADERA

Type	Manufacturer	EntryDate
Nogal	N/A	11.07.21
Nogal	N/A	11.07.21

PICKUPS

Type	Manufacturer	EntryDate
SingleCoil	DiMarzio	11.07.21
P90	EMG	11.07.21

CLAVIJAS

Type	Manufacturer	EntryDate
N/A	Gotoh	11.07.21
N/A	Grover	11.07.21

ELECTRONICA

Type	Manufacturer	EntryDate
Gibson	N/A	11.07.21
Fender	N/A	11.07.21

GUITARRA: **FABRICAR**

GUITARRAS FABRICADAS

Model: LesPaul
Wood: Id: 6 | Tipo: Fresno | Fabricante: N/A | Ingreso: 11.07.21 |
Electronics: Id: 1 | Tipo: Gibson | Fabricante: N/A | Ingreso: 11.07.21 |
Tuners: Id: 3 | Tipo: N/A | Fabricante: Gotoh | Ingreso: 11.07.21 |
Pickup: Id: 8 | Tipo: Humbucker | Fabricante: EMG | Ingreso: 11.07.21 |
Manufacture Date: 11.07.21
Serial Number: LE-16

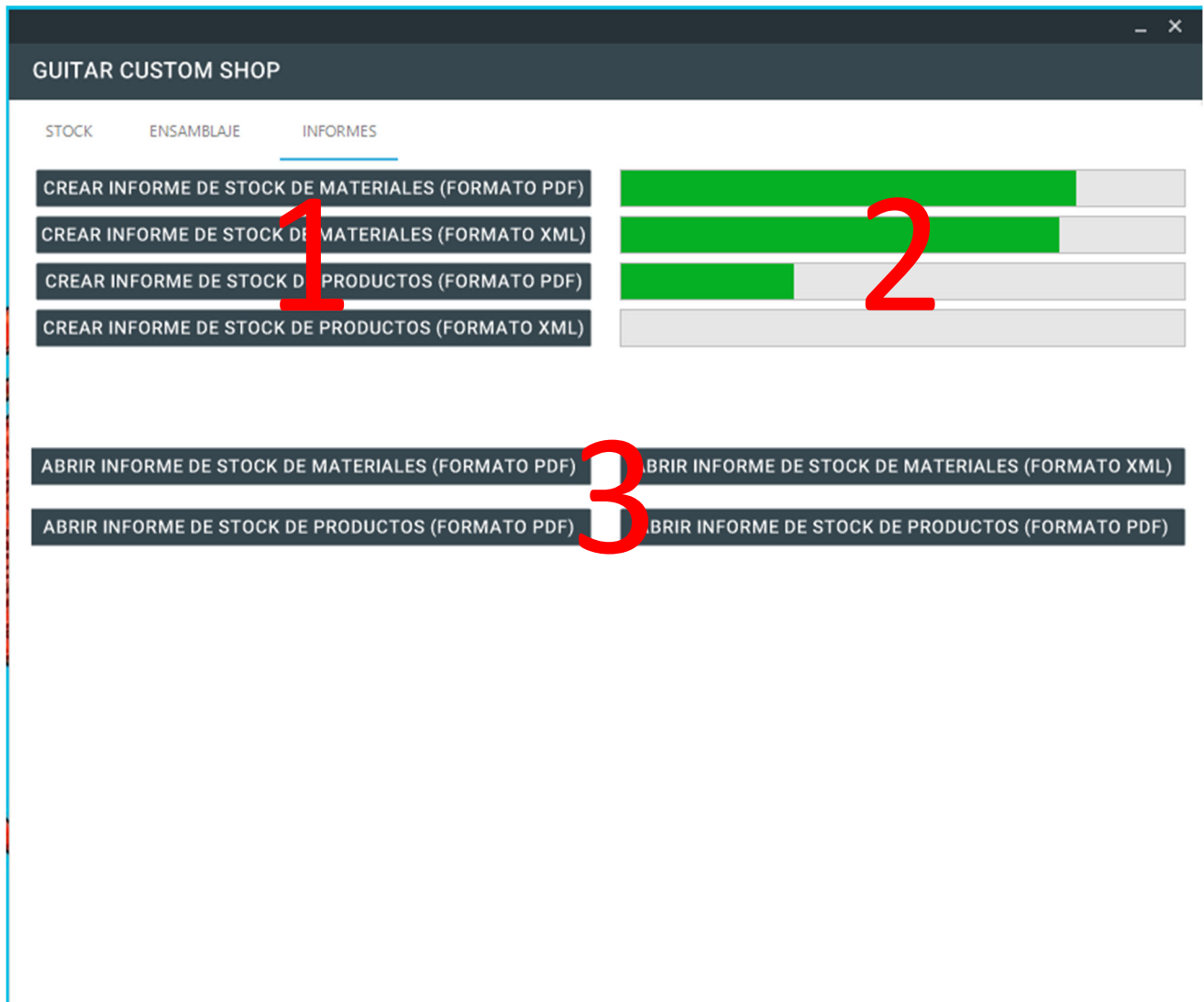
Model: Telecaster
Wood: Id: 4 | Tipo: Cedro | Fabricante: N/A | Ingreso: 11.07.21 |
Fender
Tuners: Id: 1 | Tipo: N/A | Fabricante: Grover | Ingreso: 11.07.21 |
Pickup: Id: 3 | Tipo: P90 | Fabricante: EMG | Ingreso: 11.07.21 |
Manufacture Date: 11.07.21
Serial Number: TE-15

Model: SG
Wood: Id: 11 | Tipo: Caoba | Fabricante: N/A | Ingreso: 10.07.21 |
Electronics: Id: 10 | Tipo: Gibson | Fabricante: N/A | Ingreso: 10.07.21 |
Tuners: Id: 17 | Tipo: N/A | Fabricante: Grover | Ingreso: 10.07.21 |
Pickup: Id: 12 | Tipo: Humbucker | Fabricante: SeymourDuncan | Ingreso: 10.07.21 |
Manufacture Date: 10.07.21
Serial Number: SG-12

- 1) Combobox en el que se elige el modelo de guitarra a construir.
- 2) Datagridviews en los que se discrimina por tipo de pieza y de los cuales hay que elegir un item de cada tipo de pieza para construir el producto.
- 3) Al presionar el botón fabricar se creara en la tabla Productos una entrada con la información de las piezas que conforman el producto y se eliminara dichas piezas de la tabla Piezas. En el caso que no se elija un modelo de guitarra, no se cuente con el stock suficiente de piezas, o no se haya elegido un tipo de cada pieza el programa se lo indicara al usuario por medio de una ventana emergente.
- 4) Richtextbox en la que se van a ir mostrando los productos fabricados.

VISTA INFORMES:

En esta vista el usuario puede crear informes y abrir los mismos para su uso.



- 1) En estos botones el usuario elige que formato le quiere dar a los informes(XML o PDF) y decide si quiere el informe de materiales o productos. Los informes se guardan por defecto en el Escritorio.
- 2) Barras de progreso que indican que se esta creando el informe correspondiente.
- 3) Botones para abrir los informes creados. En caso de no existir se le hará saber al usuario que primero debe crear el informe por medio de una ventana emergente.

IMPLEMENTACION DE LOS TEMAS VISTOS EN CLASE:

EXCEPCIONES: Implementado en las funciones:

- Clase frmMain:

- `private void frmMain_Load(object sender, EventArgs e)`
- `private void btnDeleteStock_Click(object sender, EventArgs e)`
- `private void btnCreate_Click(object sender, EventArgs e)`
- `private void btnPartsReportXml_Click(object sender, EventArgs e)`
- `private void btnProductsReportXml_Click(object sender, EventArgs e)`
- `private void btnProductsReportPdf_Click(object sender, EventArgs e)`
- `private void btnPartsReportPdf_Click(object sender, EventArgs e)`
- `private void openFile(string path)`

- Clase DAO:

- `public List<Part> GetAllParts()`
- `public List<Guitar> GetAllProducts()`
- `public bool SaveProduct(int id, string classType, string wood, string electronics, string pickups, string Tuners, string manufactureDate)`
- `public bool SavePiece(int id, string classType, string type, string manufacturer, string manufactureDate)`
- `public bool DeletePiece(int id)`

- Clase FilesConfig:

- `public bool XmlCreation<T>(string filePath, List<T> list)`
- `public bool PdfCreation(string info, string filePath)`

TEST UNITARIOS: Se implementaron tests unitarios evaluando los siguientes escenarios:

- Test para comprobar si todos los valores de una pieza son cargados al stock
- Test para comprobar la correcta apertura de la tabla Partes de la Base de Datos
- Test para comprobar la correcta apertura de la tabla Productos de la Base de Dato
- Test para comprobar el guardado en formato PDF
- Test para comprobar el guardado en formato XML

TIPOS GENERICOS: Implementado en la clase FilesConfig y sus métodos:

- `public class FilesConfig<T> : ISerializacion<T>`
 - `public bool XmlCreation<T>(string filePath, List<T> list)`
 - `public bool Serialize(T data, string path)`
 - `public bool Deserialize(string path, out T data)`

INTERFACES: Implementadas en las clases:

- `public class FilesConfig<T> : ISerializacion<T>`
- `public class Part : IParts`
- `public class DAO : ILogs`
- `public class Guitar : IGuitar`

ARCHIVOS Y SERIALIZACIÓN: A través de la clase FilesConfig y sus métodos se guarda la información de las tablas de la base de datos utilizada en el programa en archivos XML y PDF (gracias a la implementación del Nuget iTextSharp)

SQL: Por intermedio de la clase DAO se obtiene los datos de las tablas de la base de datos y luego además se hace las respectivas adiciones o sustracciones a dichas tablas por medio de los métodos implementados.

HILOS, EVENTOS Y DELEGADOS: Se implementaron en dos situaciones distintas:

- Un hilo que cada cierto tiempo refresca el RichTextBox con los productos terminados
- Hilos que implementan todos el mismo método para llenar la barra de progreso de creación de los informes pero cada uno para una barra en particular.

METODOS DE EXTENSION: Se utilizó en método TipoProducto de la clase Extensora, el cual extiende el tipo string para generar un nuevo string a partir de los dos primeros caracteres del string pasado como argumento.

Documentación:

Clase FrmMain:

```
public void Start():
/// <summary>
/// Refresca cada un segundo el thread con el evento refreshGuitarList();
/// </summary>

public void updateList():
/// <summary>
/// Vuelca la informacion de los productos existentes en el rich text box rtbGuitarsInfo
/// </summary>

private void refreshGrids():
/// <summary>
/// Actualiza los valores de los datagrids del form
/// </summary>

private void cmbPieceType_SelectedIndexChanged(object sender, EventArgs e):
/// <summary>
/// Maneja las posibles opciones que puede elegir el usuario en el form
/// de ingreso de stock a partir del tipo de pieza que pretende ingresar
/// </summary>

private void btnAddStock_Click(object sender, EventArgs e):
/// <summary>
/// Llama al metodo AddPart de la clase Factory el cual agrega un objeto a la lista de ///
stock de piezas
/// </summary>

private void btnDeleteStock_Click(object sender, EventArgs e):
/// <summary>
/// Llama a los metodos removePart y deleteDB de la clase factory
/// los cuales borra un objeto de la lista del stock de piezas y
/// de la base de datos respectivamente
/// </summary>

private void btnCreate_Click(object sender, EventArgs e):
/// <summary>
/// Llama al metodo AddGuitar de la clase Factory, el cual crea un producto a partir de
/// la lista de partes existentes
/// Llama a los metodos removePart y removeDB que eliminan las partes utilizadas para
/// fabricar el producto
/// </summary>

private void frmMain_FormClosing(object sender, FormClosingEventArgs e):
/// <summary>
/// Genera un cuadro de dialogo al querer cerrar el formulario, el cual
/// al ser aceptado cierra todos los hilos abiertos y cierra el formulario
/// </summary>

private void frmMain_FormClosing(object sender, FormClosingEventArgs:
/// <summary>
/// Genera un cuadro de dialogo para cerrar el formulario
/// </summary>

private void btnPartsReportXml_Click(object sender, EventArgs e):
private void btnProductsReportXml_Click(object sender, EventArgs e):
private void btnProductsReportPdf_Click(object sender, EventArgs e):
private void btnPartsReportPdf_Click(object sender, EventArgs e):
/// <summary>
/// El evento de estos botones genera el reporte de piezas y productos en formato xml y
/// pdf
/// </summary>
```

```

private void SaveDocumentProgress(ProgressBar bar):
/// <summary>
/// Animacion del ProgressBar del guardado de documentos
/// </summary>

private void openFile(string path):
/// <summary>
/// Abre el archivo en el path indicado por parametro
/// </summary>

private void openProductsPdf_Click(object sender, EventArgs e):
private void openPartsPdf_Click(object sender, EventArgs e):
private void openPartsXml_Click(object sender, EventArgs e):
private void openProductsXml_Click(object sender, EventArgs e):
/// <summary>
/// El evento de estos botones Abre el reporte de piezas y productos en formato xml y pdf
/// </summary>

```

Clase Factory:

```

public bool AddPart(string piece, string type, string manufacturer):
/// <summary>
/// A partir de los parametros que le damos genera un objeto en la lista de piezas con ///
dichas características.
/// Ademas llama al metodo SavePiece de la clase DAO que guarda la infomracion en la base
/// de datos
/// </summary>>

public void removePart(int index):
/// <summary>
/// Remueve un objeto de la lista de partes a partir del indice pasado
/// </summary>

public bool AddGuitar(string wood, string pickup, string electronic, string tuner, string
guitarType):
/// <summary>
/// Agrega un objeto a la lista de productos con los parametros obtenidos como
/// características.
/// Ademas llama al metodo SaveProduct de la clase DAO que lo guarda en la base de datos
/// </summary>

public void OpenDB():
/// <summary>
/// Carga las listas de partes y productos con la informacion de
/// la base de datos
/// </summary>

public void deleteDB(int id):
/// <summary>
/// Elimina de la base de datos el objeto con Id que
/// coincida con el valor pasado como parametro
/// </summary>

public bool CreateXml<T>(string filePath, List<T> list):
/// <summary>
/// Guarda en un archivo XML la lista de piezas en el PATH pasado como parametro
/// </summary>

public bool CreatePdf(string info, string path):
/// <summary>
/// Guarda en un archivo PDF la lista de piezas en el PATH pasado como parametro
/// </summary>

```

```

public string ProductsInfo():
/// <summary>
/// Crea un string a partir de un stringbuilder con la informacion
/// de todos los productos de la lista de productos
/// </summary>

public string PartsInfo():
/// <summary>
/// Crea un string a partir de un stringbuilder con la informacion
/// de todos las partes de la lista de partes
/// </summary>

```

Clase DAO:

```

public List<Part> GetAllParts():
/// <summary>
/// Crea una lista de Partes a partir de la informacion de la
/// tabla Piezas de la Base De Datos Stock
/// </summary>

public List<Guitar> GetAllProducts():
/// <summary>
/// Crea una lista de Productos a partir de la informacion de la
/// tabla Productos de la Base De Datos Stock
/// </summary>

public bool SaveProduct(int id, string classType, string wood, string electronics, string
pickups, string Tuners, string manufactureDate):
/// <summary>
/// Crea un item en la tabla Productos a partir de los
/// parametros pasados en la funcion
/// </summary>

public bool SavePiece(int id, string classType, string type, string manufacturer, string
manufactureDate):
/// <summary>
/// Crea un item en la tabla Piezas a partir de los
/// parametros pasados en la funcion
/// </summary>

public bool DeletePiece(int id):
/// <summary>
/// Elimina de la tabla Piezas el item con el id
/// que coincida con el parametro pasado
/// </summary>

```

Clase Extensora:

```

public static string TipoProducto(this string i):
/// <summary>
/// Extiende el tipo string para generar una identificacion
/// a partir de los dos primeros caracteres del string pasado como argumento
/// </summary>

```

Clase FilesConfig:

```

public bool XmlCreation<T>(string filePath, List<T> list):
/// <summary>
/// Crea un archivo XML la lista de piezas en el PATH pasado como parametro
/// </summary>

```



```

public bool PdfCreation(string info, string filePath):
/// <summary>
/// Crea en un archivo PDF la lista de piezas en el PATH pasado como parametro
/// Implementacion del nuget iTextSharp
/// </summary>

public bool Serialize(T data, string path):
/// <summary>
/// Guarda los datos que le pasamos como parametro en el PATH indicado como parametro
/// </summary>

public bool Deserialize(string path, out T data):
/// <summary>
/// Convierte los datos alojados en el xml que pasamos como parametro a la lista pasada
/// como parametro.
/// </summary>

```

Clase StockManagment Tests:

```

public void AddStock_Test():
/// <summary>
/// Test para comprobar si todos los valores de una pieza son cargados al stock
/// </summary>

public void ReadPartsDB_Test():
/// <summary>
/// Test para comprobar la correcta apertura de la tabla Partes de la Base de Datos
/// </summary>

public void ReadProductsDB_Test():
/// <summary>
/// Test para comprobar la correcta apertura de la tabla Productos de la Base de Dato
/// </summary>

public void SavePDF_Test():
/// <summary>
/// Test para comprobar el guardado en formato PDF
/// </summary>

public void SaveXML_Test():
/// <summary>
/// Test para comprobar el guardado en formato XML
/// </summary>

```

DIAGRAMAS DE CLASES:

Form:

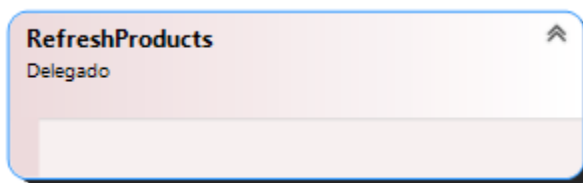
frmMain
Clase
→ MaterialForm

Campos

- btnAddStock : MaterialRaisedButton
- btnCreate : MaterialRaisedButton
- btnDeleteStock : MaterialRaisedButton
- btnPartsReportPdf : MaterialRaisedButton
- btnPartsReportXml : MaterialRaisedButton
- btnProductsReportPdf : MaterialRaisedButton
- btnProductsReportXml : MaterialRaisedButton
- classTypeDataGridViewTextBoxColumn : DataGridViewTextBoxColumn
- cmbCustomType : MetroSetComboBox
- cmbGuitarModel : MetroSetComboBox
- cmbManufacturer : MetroSetComboBox
- cmbPieceType : MetroSetComboBox
- components : IContainer
- dataGridViewTextBoxColumn10 : DataGridViewTextBoxColumn
- dataGridViewTextBoxColumn11 : DataGridViewTextBoxColumn
- dataGridViewTextBoxColumn12 : DataGridViewTextBoxColumn
- dataGridViewTextBoxColumn2 : DataGridViewTextBoxColumn
- dataGridViewTextBoxColumn3 : DataGridViewTextBoxColumn
- dataGridViewTextBoxColumn4 : DataGridViewTextBoxColumn
- dataGridViewTextBoxColumn6 : DataGridViewTextBoxColumn
- dataGridViewTextBoxColumn7 : DataGridViewTextBoxColumn
- dataGridViewTextBoxColumn8 : DataGridViewTextBoxColumn
- dgvElectronics : DataGridView
- dgvPickups : DataGridView
- dgvPieces : DataGridView
- dgvTuners : DataGridView
- dgvWood : DataGridView
- electronicsBindingSource : BindingSource
- entryDateDataGridViewTextBoxColumn : DataGridViewTextBoxColumn
- entryDateDataGridViewTextBoxColumn1 : DataGridViewTextBoxColumn
- idDataGridViewTextBoxColumn : DataGridViewTextBoxColumn
- lblElectronics : MetroSetLabel
- lblGuitarModel : MetroSetLabel
- lblManufacturer : MetroSetLabel
- lblPickups : MetroSetLabel
- lblTipoPieza : MetroSetLabel
- lblTuners : MetroSetLabel
- lblType : MetroSetLabel
- lblWood : MetroSetLabel
- manufacturerDataGridViewTextBoxColumn : DataGridViewTextBoxColumn
- manufacturerDataGridViewTextBoxColumn1 : DataGridViewTextBoxColumn
- miFabrica : Factory
- openPartsPdf : MaterialRaisedButton
- openPartsXml : MaterialRaisedButton
- openProductsPdf : MaterialRaisedButton
- openProductsXml : MaterialRaisedButton
- partBindingSource : BindingSource
- partsPathPdf : string
- partsPathXml : string
- pbPartPdf : ProgressBar
- pbPartXml : ProgressBar
- pbProductPdf : ProgressBar
- pbProductXml : ProgressBar
- pickupBindingSource : BindingSource
- pnlPieceProperties : Panel
- productsPathPdf : string
- productsPathXml : string
- refreshGuitarList : RefreshProducts
- refreshThread : Thread
- rtbGuitarsInfo : MetroSetRichTextBox
- tbAssembly : MetroSetTabPage
- tbMain : MetroSetTabControl
- tbReports : MetroSetTabPage
- tbStock : MetroSetTabPage
- tunersBindingSource : BindingSource
- typeDataGridViewTextBoxColumn : DataGridViewTextBoxColumn
- typeDataGridViewTextBoxColumn1 : DataGridViewTextBoxColumn
- woodBindingSource : BindingSource

Métodos

- btnAddStock_Click(object sender, EventArgs e) : void
- btnCreate_Click(object sender, EventArgs e) : void
- btnDeleteStock_Click(object sender, EventArgs e) : void
- btnPartsReportPdf_Click(object sender, EventArgs e) : void
- btnPartsReportXml_Click(object sender, EventArgs e) : void
- btnProductsReportPdf_Click(object sender, EventArgs e) : void
- btnProductsReportXml_Click(object sender, EventArgs e) : void
- cmbPieceType_SelectedIndexChanged(object sender, EventArgs e) : void
- Dispose(bool disposing) : void
- frmMain()
- frmMain_FormClosing(object sender, FormClosingEventArgs e) : void
- frmMain_Load(object sender, EventArgs e) : void
- InitializeComponent() : void
- openFile(string path) : void
- openPartsPdf_Click(object sender, EventArgs e) : void
- openPartsXml_Click(object sender, EventArgs e) : void
- openProductsPdf_Click(object sender, EventArgs e) : void
- openProductsXml_Click(object sender, EventArgs e) : void
- refreshGrids() : void
- SaveDocumentProgress(ProgressBar bar) : void
- Start() : void
- updateList() : void



Clase Part y Clases Hijas:

IParts

Part
Clase

Campos

classType : string
entryDate : string
id : int
manufacturer : string
type : string

Propiedades

ClassType { get; set; } : string
EntryDate { get; set; } : string
Id { get; set; } : int
Manufacturer { get; set; } : string
Type { get; set; } : string

Métodos

Data() : string
getClassType() : string
Part()
Part(int id, string classType, string type, string entryDate, string manufacturer)

IParts
Interfaz

Propiedades

ClassType { get; } : string
EntryDate { get; set; } : string
Manufacturer { get; set; } : string
Type { get; set; } : string

Métodos

getClassType() : string

Electronics

Clase
→ Part

Propiedades

Manufacturer { get; set; } : string
Type { get; set; } : string

Métodos

Data() : string
Electronics()
Electronics(int id, string classType, string type, string entryDate, string manufacturer)

Pickup

Clase
→ Part

Propiedades

Manufacturer { get; set; } : string
Type { get; set; } : string

Métodos

Data() : string
Pickup()
Pickup(int id, string classType, string type, string entryDate, string manufacturer)

Wood

Clase
→ Part

Propiedades

Manufacturer { get; set; } : string
Type { get; set; } : string

Métodos

Data() : string
Wood()
Wood(int id, string classType, string type, string entryDate, string manufacturer)

Tuners

Clase
→ Part

Propiedades

Manufacturer { get; set; } : string
Type { get; set; } : string

Métodos

Data() : string
Tuners()
Tuners(int id, string classType, string type, string entryDate, string manufacturer)

Clase Guitar y Clases Hijas:

IGuitar

Guitar

Clase

Campos

- classType : string
- electronics : string
- id : int
- manufactureDate : string
- pickups : string
- tuners : string
- wood : string

Propiedades

- ClassType { get; set; } : string
- Electronics { get; set; } : string
- Id { get; set; } : int
- ManufactureDate { get; set; } : string
- Pickups { get; set; } : string
- Tuners { get; set; } : string
- Wood { get; set; } : string

Métodos

- getClassType() : string
- Guitar()
- Guitar(int id, string classType, string wood, string pickups, string tuners, string electronics, string manufactureDate)

IGuitar

Interfaz

Propiedades

- Electronics { get; set; } : string
- ManufactureDate { get; set; } : string
- Pickups { get; set; } : string
- Tuners { get; set; } : string
- Wood { get; set; } : string

SG

Clase

→ Guitar

Propiedades

- Electronics { get; set; } : string
- Pickups { get; set; } : string
- Tuners { get; set; } : string
- Wood { get; set; } : string

Métodos

- SG()
- SG(int id, string classType, string wood, string pickups, string tuners, string electronics, string manufactureDate)

LesPaul

Clase

→ Guitar

Propiedades

- Electronics { get; set; } : string
- Pickups { get; set; } : string
- Tuners { get; set; } : string
- Wood { get; set; } : string

Métodos

- LesPaul()
- LesPaul(int id, string classType, string wood, string pickups, string tuners, string electronics, string manufactureDate)

Telecaster

Clase

→ Guitar

Propiedades

- Electronics { get; set; } : string
- Pickups { get; set; } : string
- Tuners { get; set; } : string
- Wood { get; set; } : string

Métodos

- Telecaster()
- Telecaster(int id, string classType, string wood, string pickups, string tuners, string electronics, string manufactureDate)

Clase Factory y Enumerados:

Factory
Clase

Campos

dao : DAO

fileManager : SerializeConfig<object>

guitarsList : List<Guitar>

partsList : List<Part>

Propiedades

GuitarsList { get; set; } : List<Guitar>

PartsList { get; set; } : List<Part>

Métodos

AddGuitar(string wood, string pickup, string electronic, string tuner, string guitarType) : bool

AddPart(string piece, string type, string manufacturer) : bool

CreatePdf(string info, string path) : bool

CreateXml<T>(string filePath, List<T> list) : bool

DeleteDB(int id) : void

Factory()

OpenDB() : void

PartsInfo() : string

ProductsInfo() : string

removePart(int index) : void

ElectronicType
Enum

Gibson
Fender

TunersMakers
Enum

Gotoh
Grover

WoodType
Enum

Caoba
Cedro
Nogal
Fresno


PickupsType
Enum

Humbucker
SingleCoil
P90

PickupsMaker
Enum

SeymourDuncan
DiMarzio
EMG

Clase DAO:

 ILogs

DAO
Clase

Campos

command : SqlCommand

connection : SqlConnection

connectionString : string

Métodos

DeletePiece(int id) : bool

GetAllParts() : List<Part>

GetAllProducts() : List<Guitar>

SavePiece(int id, string classType, string type, string manufacturer, string manufactureDate) : bool

SaveProduct(int id, string classType, string wood, string electronics, string pickups, string Tuners, string manufactureDate) : bool

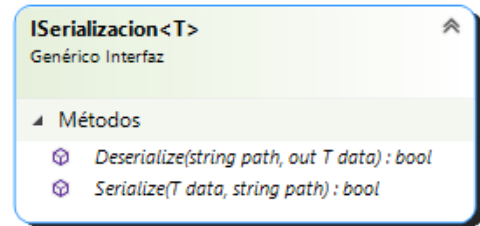
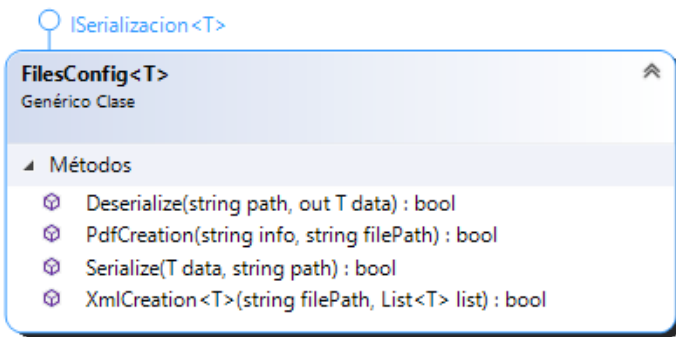
ILogs
Interfaz

Métodos

SavePiece(int id, string classType, string type, string manufacturer, string manufactureDate) : bool

SaveProduct(int id, string classType, string wood, string electronics, string pickups, string Tuners, string manufactureDate) : bool

Class FilesConfig:



Class Extensora:

