Universidad Católica de Santa María

FACULTAD DE CIENCIAS E INGENIERÍAS FÍSICAS Y FORMALES

PROGRAMA PROFESIONAL DE INGENIERÍA DE SISTEMAS

EXTENSIÓN PARA LA REALIZACIÓN DE BÚSQUEDAS DINÁMICAS Y GUÍA DE IMPLEMENTACIÓN PARA DESARROLLADORES DE GENEXUS EV2

TESIS PRESENTADA POR EL BACHILLER:

Sony Yvan Gallegos Quispe

PARA OPTAR EL TÍTULO PROFESIONAL DE:

Ingeniero de Sistemas

Arequipa – Perú

2012

Presentación

Sra. Directora del Programa Profesional de Ingeniería de Sistemas

Sres. Miembros del Jurado.

De conformidad con las disposiciones del Reglamento de Grados y Títulos del Programa Profesional de Ingeniería de Sistemas, pongo a vuestra consideración el presente trabajo de investigación titulado:

“Extensión Para La Realización De Búsquedas Dinámicas Y Guía De Implementación Para Desarrolladores De GeneXus Ev2.”

Sony Yvan Gallegos Quispe

Agradecimientos

A Dios padre todopoderoso por permitirme seguir en éste mundo,

A mi familia por estar siempre a mi lado,

A mi asesor el Dr. Guillermo Calderón Ruiz por su invaluable guía y apoyo,

Y un agradecimiento especial al Ing. Jorge Luis Martinez Muñoz por su apoyo para la finalización de este trabajo.

Dedicatoria

A mi madre por su amor y apoyo.

Tabla de contenido

Presentación 2
Agradecimientos 3
Dedicatoria 4
Tabla de contenido 5
Índice de tablas 6
Índice de figuras 7
Resumen 8
Abstract 9
Introducción 10
CAPÍTULO I 13
1.1 GUÍA DE IMPLEMENTACIÓN DE EXTENSIONES. ..... ¡ERROR! MARCADOR NO DEFINIDO.
1.1.1 Identificador único global *13*
1.1.2 Instalación Genexus X Platform Sdk *15*
1.1.3 Crear Un Proyecto De Extensiones *16*
1.1.4 Programando Mi Extensión *17*
1.1.5 Crear Objetos GeneXus *29*
1.1.6 Obtener tablas y campos *43*
CAPÍTULO II 45
2.1 EXTENSIÓN PARA CONSULTAS DINÁMICAS EN GENEXUS. 45
2.1.1 Análisis y Diseño *45*
2.2 DESCRIPCIÓN DE ARCHIVOS – CÓDIGO FUENTE DE LA EXTENSIÓN. 53
CAPÍTULO III 63
3.1 VALIDACIÓN 63
3.1.1 Variables Dependientes *63*
Anexos 73
ANEXO A: PROYECTO DE TESIS 74
ANEXO B: CÓDIGO FUENTE 96
ANEXO C: ENCUESTA 134

Índice de tablas

Tabla 1. Analistas – desarrolladores genexus encuestados..... 63

Índice de figuras

Figura 1. Creación del identificador único global con visual studio 2005 14
Figura 2. Creación del identificador único global desde página web 15
Figura 3. Crear proyecto genexus package paso 1 16
Figura 4. Crear proyecto genexus package paso 2 17
Figura 5. Mi primer menú integrado a la ide genexus ev2 21
Figura 6. Miprimeraventana 28
Figura7. Sdtejemplo 35
Figura 8. Importación de external object paso 1 39
Figura 9. Importación de external object paso 2 39
Figura 10. Importación de external object paso 3 40
Figura 11. External object importado 41
Figura 12. Propiedades del external object importado 42
Figura 13. Variable del external object en un web panel 43
Figura 14. Diagrama de caso de uso 49
Figura 15. Diagrama de secuencia iniciar extensión 50
Figura 16. Diagrama de secuencia definir consulta 50
Figura 17. Diagrama de secuencia programar con extensión 51
Figura 18. Diagrama de secuencia realizar consulta 51
Figura 19. Diagrama de clases 52
Figura 20. Árbol de archivos 53
Figura 21. Facilidad de instalación 65
Figura 22. Facilidad de implementación 66
Figura 23. Comprensión de la guía 66
Figura 24. Ejemplos de creación de objetos 67
Figura 25. Utilidad de la extensión 68

Resumen

GeneXus Ev2 es una herramienta para el desarrollo ágil de sistemas de información que gracias a la base de conocimientos creada por la experticia de los usuarios finales, convierte las tareas de gestión de base de datos y codificación de los programas en transparentes al equipo de desarrollo, ayudando así a la creación temprana de prototipos funcionales para verificar la correcta abstracción de los requerimientos solicitados por los usuarios. De modo que la gestión de base de datos es transparente al equipo de desarrollo, ésta ventaja se convierte en una limitante a la hora de querer implementar interfaces que ayuden a realizar consultas dinámicas. Es por ello que el presente trabajo de investigación desarrolla la extensión para agregar las funcionalidades de definir y ejecutar sentencias en el lenguaje de consulta estructurado en aplicaciones hechas por GeneXus Ev2, además documenta los conceptos básicos y ejemplos prácticos en la guía de implementación para desarrollar nuevas extensiones. Obteniendo como resultado la instalación sencilla de la extensión, la implementación de forma simple para consultas dinámicas, y la fácil comprensión de la guía de desarrollo de extensiones.

Abstract

GeneXus Ev2 is a tool for fast development of information systems thanks to the knowledge base created by the end-user expertise, makes the tasks of database management and coding programs transparent to the development team, thus helping creation of functional prototypes early to verify correct abstraction of the requirements requested by users. So the database management is transparent to the development team, this advantage becomes a limiting factor when trying to implement interfaces that help dynamic query. That is why this research develops the extension to add functionality to define and execute statements in structured query language made by GeneXus Ev2 applications, also documents the basic concepts and practical examples in the Implementation Guide for developing new extensions. Resulting in easy installation of the extension, the simple way to implement dynamic queries, and readily understandable guide extension development.

Introducción

GeneXus Ev2 es una poderosa herramienta para desarrollar sistemas de información, que gracias a su novedoso enfoque para abstraer los objetos del mundo real, logra capturar satisfactoriamente la esencia de cada uno de ellos, obteniendo así como resultado la base de conocimientos que se encarga de crear el sistema de base de datos y el código fuente necesario para la aplicación, también su sintaxis declarativa hace que la gestión del sistema de base de datos se haga de forma transparente al equipo de desarrollo, ayudando así a la creación temprana de prototipos funcionales para verificar la correcta abstracción de los requerimientos de los usuarios finales, permitiendo así tener un proceso incremental e iterativo de desarrollo de software.

Sin embargo presenta la restricción que no se puede implementar aplicaciones que ayuden al usuario final a realizar consultas a sus bases de datos en forma dinámica, debido a que la declaración de consultas mediante la sintaxis GeneXus no permite al equipo de desarrollo definir rutinas que ayuden a la realización de tales aplicaciones. Haciendo de las consultas tareas transparentes al usuario desarrollador y sus definiciones con condiciones previamente conocidas, como concecuencia la obtensión de datos por parte de las aplicaciones hechas por GeneXus Ev2 no puede ser dinámica.

Para solucionar tal restricción de GeneXus Ev2 se decidió desarrollar la extensión que permita a los usuarios desarrolladores poder implementar consultas que sean variables en tiempo de ejecución de las aplicaciones. Para cumplir con tal objetivo se tuvo que realizar las investigaciones pertinentes sobre la herramienta GeneXus Ev2 y el estado actual del tema

de desarrollo de extensiones, encontrándo al segundo poco conocido entre los profesionales que se dedican al diseño e implementación de aplicaciones GeneXus Ev2, es por ello que los propósitos de ésta investigación son (a) el desarrollo de la extensión para GeneXus Ev2 que agrega las funcionalidades de definir y ejecutar sentencias en el lenguaje de consulta estructurado y (b) documentar conceptos básicos, ejemplos prácticos que ayuden a la comprensión de los interesados en desarrollar nuevas extensiones.

El presente informe de tesis se divide en tres capítulos, el primer capítulo describe que es el identificador único global y para qué sirve, realiza la explicación de los archivos generados después de la instalación de la plataforma de desarrollo de extensiones, muestra ejemplos prácticos sobre la creación de los objetos GeneXus y por último enseña como recorrer las tablas y campos de la base de datos. El segundo capítulo trata sobre el desarrollo de la extensión, en el cuál se especifican y describen los requerimientos funcionales y no funcionales, se diagrama los casos de uso, de secuencia y de clases, luego se realiza la descripción de las clases que se generaron para la implementación y se especifica con exactitud el código fuente escrito. En el tercer capítulo se validan las distintas variables dependientes que se especificaron en el plan de tesis con la ayuda de la encuesta, la cual se aplicó a profesionales dedicadas al diseño e implementación de aplicaciones en GeneXus Ev2 que fueron recomendadas por la persona que se dedica a la distribución de productos GeneXus en todo el sur del país; la encuesta tiene por objetivo medir el grado de utilidad de la extensión, la facilidad de la instalación, el esfuerzo para la implementación de consultas dinámicas, la comprensión de la guía y cuán útiles son los ejemplos de creación de objetos GeneXus mediante código fuente, obteniendo resultados satisfactorios para cada punto

anterior mencionado, así entonces se concluyó que la herramienta es fácil de instalar, la implementación de consultas dinámicas se realiza con facilidad, la guía de implementación por parte de los profesionales dedicados al desarrollo de aplicaciones en GeneXus es fácil de comprender, los ejemplos sobre la creación de objetos GeneXus tiene gran utilidad así como el propósito de la extensión.

CAPÍTULO I

1. GUÍA DE IMPLEMENTACIÓN DE EXTENSIONES.

El presente capítulo está dividido en seis secciones. La sección uno trata del identificador único global que cada objeto en GeneXus debe tener para su respectiva identificación. La sección dos describe los elementos que son creados al instalar la plataforma para el desarrollo de extensiones. La sección tres trata sobre el inicio de proyectos de extensiones. La sección cuatro se refiere al ejemplo de creación de extensión, en el cual se describe paso a paso como se debe de implementar. La sección cinco toca el tema de creación de objetos GeneXus, mostrando ejemplos prácticos para su fácil compresión. La última y no por eso menos importante sección seis muestra un ejemplo práctico como obtener las tablas y campos de la base de datos de la aplicación.

1.1 Identificador único global.

El Identificador Único Global (GUID por sus siglas en inglés Globally Unique Identifier), es incorporado desde GeneXus X Ev1 donde todos los objetos tienen un GUID que es la clave única que identifica a cada uno de ellos (GuidGenerator, 2012).

1.1.1 ¿Para qué sirve el identificador único global?

Por ejemplo si se quiere exportar objetos desde cierta base de conocimiento (KB por sus siglas en inglés Knowledge Base) a otra, la GUID ayuda a diferenciar objetos que tienen el mismo nombre, y posteriormente si se decide importar los

objetos a la KB original los objetos podrán actualizarse identificándose por la GUID (Blog de Marcos Crispino, 2009).

1.1.2 Conseguir el identificar único global.

Hay dos formas de conseguir el identificador único global para objetos GeneXus, la primera es desde la IDE de Visual Studio: Menú Herramientas, Opción Crear GUID, la siguiente figura 1 muestra el cuadro de diálogo para obtener el identificador único global.

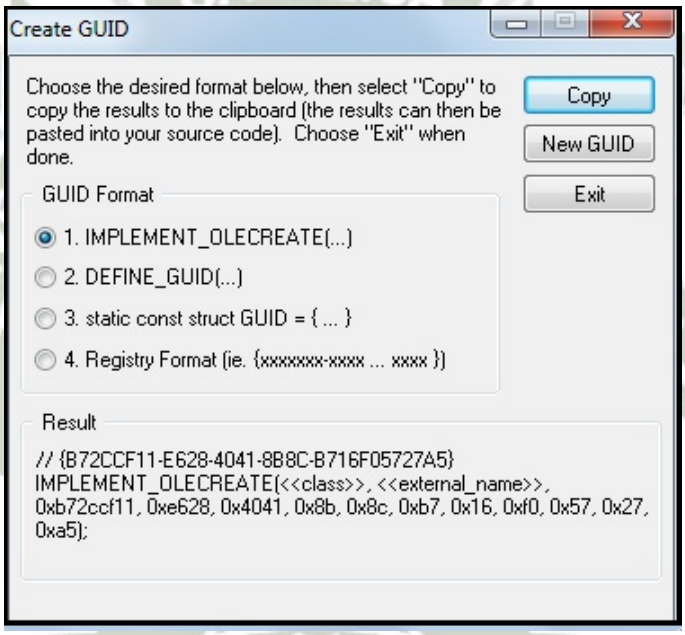


Figura 1. Creación del identificador único global con visual studio 2005. Fuente: Elaboración propia.

La segunda forma es mediante página web (GuidGenerator, 2012), presionando el botón “Generate some GUIDs!”, como se muestra en la figura 2:

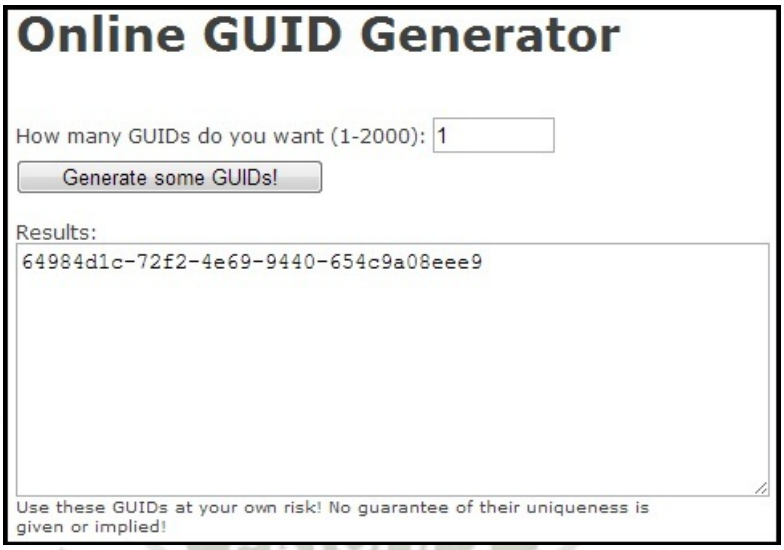


Figura 2. Creación del identificador único global desde página web. Fuente: Elaboración propia.

1.2 Instalación Genexus X Platform Sdk.

Para descargar el instalador visite el download center de GeneXus (GXtechnical, 2012).

Al instalar se crearán las siguientes carpetas en el directorio de instalación (CommunityWiki, 2006):

- Bin: Contiene las librerías que pueden ser referenciadas por las extensiones.
- PackageBuilder: Asistente integrado a Visual Studio 2005 o Visual C# 2005 Express Edition para crear proyectos para el desarrollo de extensiones.
- PatternBuilder: Asistente integrado a Visual Studio 2005 o Visual C# 2005 Express Edition para crear proyectos para el desarrollo de patterns.
- Patterns: Contiene las librerías que pueden ser referenciadas en proyectos de desarrollo de patterns.

- Samples: Conjunto de proyectos que muestran diferentes posibilidades de extensiones.
- Schemas: Contiene el archivo en formato de lenguaje de marcas extensible (XML por sus siglas en inglés eXtensible Markup Language) que posee las diferentes configuraciones que la extensión puede tomar al acoplarse al Entorno de Desarrollo Integrado (IDE) de GeneXus Ev2.

1.3 Crear Un Proyecto De Extensiones

Iniciar el Microsoft Visual C# 2005 Express Edition. Hacer click en el menú File, en el menú contextual click en Nuevo proyecto, aparecerá el siguiente cuadro de diálogo como se muestra en la figura 3:

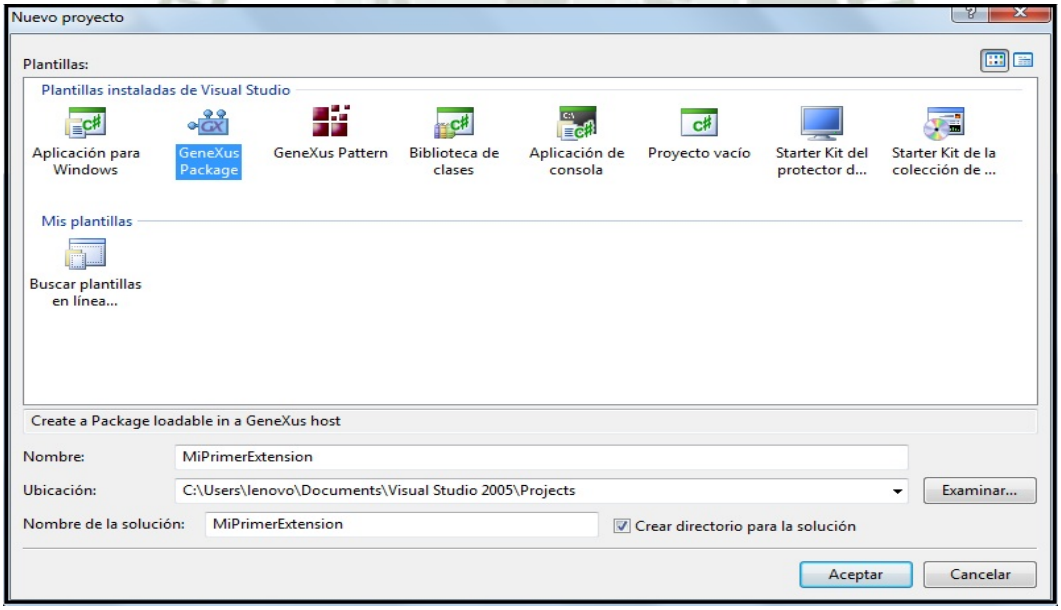


Figura 3. Crear proyecto extensión paso 1. Fuente: Elaboración propia.

Elegir la plantilla GeneXus Package, colocar nombre al proyecto de extensión y Aceptar, la siguiente figura 4 muestra el siguiente paso.

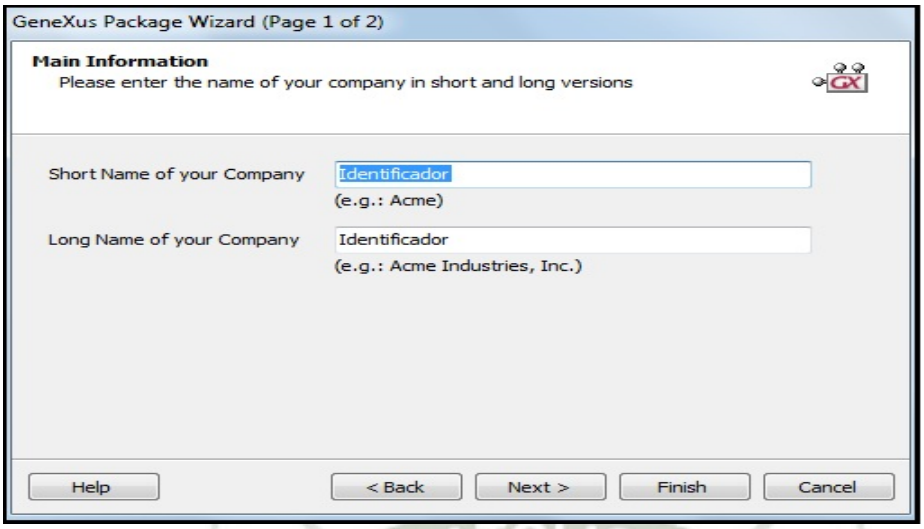


Figura 4. Crear proyecto extensión paso 2. Fuente: Elaboración propia.

Colocar algún identificador, éste aparecerá como el fabricante de la extensión, presionar finalizar.

1.4 Programando Mi Extensión

1.4.1 ¿Cómo crear un menú?

a. Para la creación de un menú agregar las siguientes referencias al proyecto (Marcos Crispino, 2008).

Artech.Architecture.UI.Framework

Artech.Common

Artech.Common.Framework

System.Drawing

b. En el archivo Package.cs agregar las siguientes directivas:

using Artech.Architecture.Common.Packages;

using Artech.Architecture.Common.Services;

using Artech.Architecture.UI.Framework.Packages;

using System.Runtime.InteropServices;

using System;

c. Asegurarse que la clase Package.cs herede de la clase AbstractPackageUI.

public class Package : AbstractPackageUI

d. Agregar en la clase Package.cs el siguiente atributo:

public static Guid guid = typeof(Package).GUID;

e. Crear la clase CommandKeys.cs

a. Agregar la siguiente directiva:

using Artech.Common.Framework.Commands;

b. Agregar el siguiente atributo:

private static CommandKey cmdMiComando =

new CommandKey(Package.guid, “MiPrimerMenu”);

c. Agregar la siguiente propiedad:

public static CommandKey MiPrimerMenu {

get { return cmdMiComando; }

}

f. Crear la clase CommandManager.cs

a. Agregar las siguientes directivas:

```
using Artech.Architecture.UI.Framework.Helper;
using Artech.Architecture.UI.Framework.Services;
using Artech.Common.Framework.Commands;
using System.Windows.Forms;
```

b. Asegurarse que la clase CommandManager.cs herede de la clase CommandDelegator:

```
class CommandManager:CommandDelegator
```

c. Crear los métodos siguientes:

```
public bool ExecMiComando(CommandData data){
        MessageBox.Show("Mi primer Menú!"); return true;
}
private bool QueryMiComando(CommandData data,
ref CommandStatus status) {
        status.State = CommandState.Disabled;
        if(UIServices.KB != null && UIServices.KB.CurrentKB != null){
            status.State = CommandState.Enabled;
```

```
        }

    return true;

    }
```

d. Crear el constructor y agregar el método AddCommand().

```
    public CommandManager() {

        AddCommand(CommandKeys.MiPrimerMenu,

        new ExecHandler(ExecMiComando),

        new QueryHandler(QueryMiComando));

    }
```

g. En el archivo Package.cs agregar la siguiente línea:

```
public override void Initialize(IGxServiceProvider services){

    base.Initialize(services);

    AddCommandTarget(new CommandManager());

}
```

h. Por último en el archivo GeneXusPackage.package agregar:

```
<Commands>

    <CommandDefinition id='MiPrimerMenu'></CommandDefinition>
```

```
</Commands>

<Menus>

    <Menu type='menubar'>

        <Popup id='MiPrimerMenu' name='MiPrimerMenu' insertBefore='Help'>

            <Command refid='MiPrimerMenu'/>

        </Popup>

    </Menu>

    </Menus>
```

Para ver el resultado (ver figura 5) generar la solución del proyecto y abrir directamente el GeneXus Ev2 o presionar F5 (Modo depuración) desde la IDE de Microsoft Visual C# 2005 Express Edition.

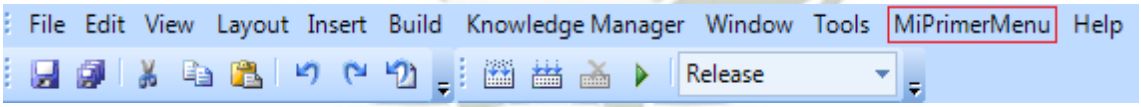


Figura 5. Mi primer menú integrado a la IDE GeneXus Ev2. Fuente: Elaboración propia.

1.4.2 ¿Cómo llamar una ventana desde el menú contextual?

Para el desarrollo de la extensión es necesario invocarlo desde la IDE de GeneXus Ev2. En éste caso se invocará a la ventana principal de la extensión desde

el menú contextual de *Other Tool Windows*, para ello ir al menú *View* y la opción *Other Tool Windows* del menú contextual.

a. Agregar las siguientes referencias al proyecto:

Artech.Architecture.Common

Artech.Architecture.UI.Framework

Artech.Common

Artech.Common.Framework

Artech.Common.Properties

Artech.FrameworkDE

Artech.Udm.Framework

System

System.Drawing

System.Windows.Forms

b. Crear la clase MiPrimeraVentana.cs

a. Agregar las siguientes directivas:

```
using System;
using System.Windows.Forms;
using System.Drawing;
using Artech.FrameworkDE;
using Artech.Architecture.UI.Framework.Packages;
using Artech.Architecture.UI.Framework.Services;
```

```
using Artech.Common.Framework.Commands;
using Artech.Common.Framework.Selection;
using System.Runtime.InteropServices;
```

b. Conseguir el identificador único global.

c. Agregar el identificador único global:

```
namespace Identificador.Packages.MiPrimeraExtension
{
    [Guid("64984d1c-72f2-4e69-9440-654c9a08eee9")]
    public class MiPrimeraVentana{}
}
```

d. Asegurarse que la clase herede de las siguientes clases: AbstractToolWindow, ISelectionListener.

```
public class MiPrimeraVentana : AbstractToolWindow,ISelectionListener{}
```

e. Agregar los siguientes atributos:

```
private Label mensaje;
public static Guid guid;
```

f. En el constructor de la clase agregar la siguiente línea:

```
public MiPrimeraVentana(){
    this.InitializeComponent();
    UIServices.TrackSelection.Subscribe(Guid.NewGuid(), this);
}
```

g. Agregar el siguiente constructor statico:

```
static MiPrimeraVentana(){
MiPrimeraVentana.guid = typeof(MiPrimeraVentana).GUID;
}
```

h. Agregar el método InitializeComponent:

```
public void InitializeComponent(){
this.mensaje = new Label();
this.mensaje.Name = “lblMensaje”;
this.mensaje.Text = “Mi primera ventana!”;
this.mensaje.Location = new Point(5, 20);
this.mensaje.Font = new Font(“Verdana”, 16F,
    System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
    ((byte)(0)));
this.mensaje.Size = new Size(100, 30);
this.Controls.Add(this.mensaje);
this.SuspendLayout();
this.ResumeLayout(false);
this.PerformLayout();
}
```

i. Por último agregar el siguiente método:

```
public bool OnSelectChange(ISelectionContainer sc){return true;}
```

c. Crear la clase CommandKeys.cs

a. Agregar la siguiente directiva:

using Artech.Common.Framework.Commands;

b. Agregar el siguiente atributo:

private static CommandKey cmdMiComando =

new CommandKey(MiPrimeraVentana.guid, “MiPrimeraVentana”);

c. Agregar la siguiente propiedad:

public static CommandKey PrimeraVentana {

get { return cmdMiComando; }

}

d. Crear la clase CommandManager.cs

a. Agregar las siguientes directivas:

using Artech.Architecture.UI.Framework.Helper;

using Artech.Architecture.UI.Framework.Services;

using Artech.Common.Framework.Commands;

b. Asegurarse que herede de la clase: CommandDelegator

class CommandManager:CommandDelegator{}

c. Agregar los siguientes métodos:

public bool ExecMiComando(CommandData data){

UIServices.ToolWindows.SelectToolWindow(MiPrimeraVentana.guid);

```
return true;
}

private bool QueryMiComando(CommandData data, ref CommandStatus
status){
        status.State = CommandState.Disabled;
        if (UIServices.KB != null && UIServices.KB.CurrentKB != null){
                status.State = CommandState.Enabled;
        }
        return true;
}
```

d. Agregar el constructor:

```
public CommandManager(){
                AddCommand(CommandKeys.PrimeraVentana,
                new ExecHandler(ExecMiComando),
                new QueryHandler(QueryMiComando));
        }
```

e. Editar la clase Package.cs

a. Agregar las siguientes directivas:

```
using Artech.Architecture.Common.Packages;
using Artech.Architecture.Common.Services;
using Artech.Architecture.UI.Framework.Packages;
```

```
using Artech.Architecture.UI.Framework.Services;
using Artech.Common.Framework.Selection;
using System.Runtime.InteropServices;
using Artech.Architecture.UI.Framework.Controls;
using System;
```

b. Asegurarse que la clase herede de: AbstractPackageUI

```
public class Package : AbstractPackageUI{}
```

c. Agregar el siguiente atributo:

```
private MiPrimeraVentana ventana;
```

d. Agregar la propiedad sobrecargado Name:

```
public override string Name{
        get { return “MiPrimeraExtension”; }
}
```

e. Agregar el método sobrecargado CreateToolWindow:

```
public override IToolWindow CreateToolWindow(Guid toolWindowId){
        if (toolWindowId.Equals(MiPrimeraVentana.guid)){
                if (ventana == null){
                        ventana = new MiPrimeraVentana();
                }
        return ventana;
        }
```

return base.CreateToolWindow(toolWindowId);

}

f. Editar el archivo GeneXusPackage.package

a. Agregar la siguientes líneas:

<ToolWindows>

<ToolWindow id =’64984d1c-72f2-4e69-9440-654c9a08eee9’ title=’MiPrimeraVentana’ mdi=’true’/>

</ToolWindows>

NOTA: El identificador único global que aparece en la clase MiPrimeraVentana.cs debe coincidir con el identificador único global del id del ToolWindow en el archivo GeneXusPackage.package.

Para ver el resultado en la IDE de GeneXus Ev2 (ver figura 6), vamos al *menú View, Other Tool Windows, opción MiPrimeraVentana.*

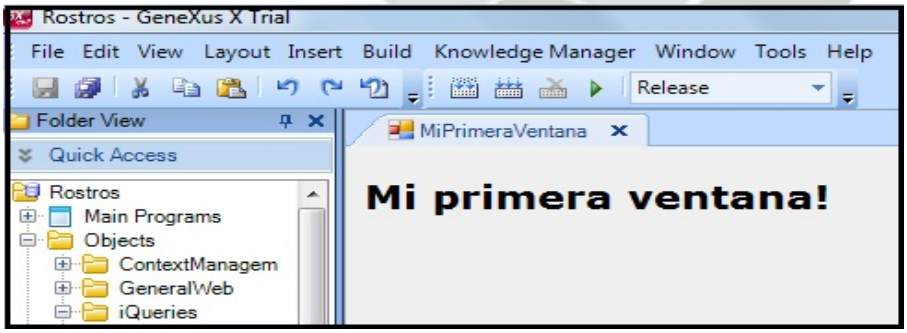


Figura 6. MiPrimeraVentana. Fuente: Elaboración propia.

1.5 Crear Objetos GeneXus

Para cumplir con los objetivos de la extensión en ocasiones se necesita crear objetos GeneXus mediante código fuente, en los siguientes ejemplos se muestran la creación de los objetos: folder, procedure, structured data type, dataprovider, dominio y external object.

1.5.1 Objeto Folder

Referenciar las siguientes librerias:

Artech.Architecture.Common.Objects

Artech.Architecture.UI.Framework.Services

Antes de crear cualquier objeto se necesita comprobar si la base de conocimientos está disponible, con el siguiente código:

```
if(UIServices.KB != null && UIServices.KB.CurrentModel != null){}
```

Así se comprueba que la KB y el Modelo están disponibles, entonces puede recién crearse objetos.

Para crear un folder en el árbol del Folder View de GeneXus:

Se comprueba si el nombre “iQueries” para el Folder existe:

```
Folder folder = folder = Folder.Get(UIServices.KB.CurrentModel, “iQueries”);
if (folder == null){}
```

Entonces se pregunta si el folder (en éste caso “iQueries”) no existe, se procede a crear el folder:

folder = new Folder(UIServices.KB.CurrentModel);

folder.Parent = Folder.GetRoot(UIServices.KB.CurrentModel);

folder.Name = “iQueries”;

folder.Save();

1.5.2 Objeto Procedure

Referenciar las siguientes librerias:

Artech.Architecture.Common.Objects

Artech.Architecture.UI.Framework.Services

Artech.Genexus.Common.Objects

Artech.Genexus.Common

En el paso anterior se creó el objeto folder, el siguiente procedure que será creado se colocará ahí:

Recuperar el folder “iQueries”:

Folder folder = Folder.Get(UIServices.KB.CurrentModel, “iQueries”);

Comprobar que el procedure no existe:

Procedure procedure = Procedure.Get(UIServices.KB.CurrentModel,

“MiProcedimiento”);

if(procedure == null){}

Se crea y coloca el procedure en el Folder:

```
procedure = new Procedure(UIServices.KB.CurrentModel);

procedure.Name = “MiProcedimiento”;

procedure.Parent = folder;

procedure.Save();
```

NOTA: Para que el objeto procedure aparezca directamente en el árbol de *folder view* de GeneXus Ev2, se establece la propiedad parent del procedure del siguiente modo:

```
procedure.Parent = Folder.GetRoot(UIServices.KB.CurrentModel);
```

a. Variables

Para agregar variables al objeto procedure anterior el siguiente código ayuda a hacerlo:

```
Variable variable = new Variable(procedure.Variables);
variable.Name = “query”;
variable.Type = eDBType.VARCHAR; //definimos el tipo
variable.Length = 9999;          //definimos el tamaño
procedure.Variables.Add(var);
procedure.Save();
```

b. Source

Para escribir directamente en el source del procedure, el siguiente código ayuda a hacerlo:

Utilizando la variable anteriormente creada:

```
string source = “&query = ‘Hola mundo con procedure’”;
procedure.ProcedurePart.Source = source;
procedure.Save();
```

c. Rules

Para agregar las rules de in u out en un objeto procedure el siguiente código ejemplifica como hacerlo:

```
procedure.Rules.Source = “parm(out:&variable);”;
procedure.Save();
```

En la regla se establece a la variable anteriormente creada que será un parámetro *out*, en caso de querer que sea una *rule* de *in* solo se debe sustituir el *out* por el *in*.

1.5.3 Structured Data Type

Referenciar las siguientes librerías:

```
Artech.Architecture.Common.Objects
Artech.Architecture.UI.Framework.Services
Artech.Genexus.Common.Objects
Artech.Genexus.Common.Parts.SDT
```

Recuperar el folder:

```
Folder folder = Folder.Get(UIServices.KB.CurrentModel, “iQueries”);
```

Comprobar si el objeto no existe:

```
SDT miSdt = SDT.Get(UIServices.KB.CurrentModel, “miSdt”);
if(miSdt == null){}
```

Crear y colocar el objeto en el folder:

```
miSdt = new SDT(UIServices.KB.CurrentModel);
miSdt.Parent = folder;
miSdt.Name = “miSdt”;
```

Agregar un nivel al objeto Structured Data Type y establecer la propiedad IsCollection a True:

```
SDTLevel sdtLevel = miSdt.SDTStructure.Root;
sdtLevel.Name = “Item”;
sdtLevel.Items.Clear();
sdtLevel.IsCollection = true;
sdtLevel.CollectionItemName = “Item”;
```

Crear y agregar *items* al *level* recién creado:

```
SDTItem item = new SDTItem(sdtLevel.SDTStructure);
item.Name = “item1”;
item.Type = eDBType.VARCHAR; //definimos el tipo
item.Length = 9999;          //definimos el tamaño
```

item.IsCollection = false;

sdtLevel.AddItem(item);

miSdt.Save();

1.5.4 DataProvider

Referenciar las siguientes librerias:

Artech.Architecture.Common.Objects

Artech.Architecture.UI.Framework.Services

Artech.Genexus.Common.Objects

Recupera el folder:

Folder folder = Folder.Get(UIServices.KB.CurrentModel, “iQueries”);

Comprobar si existe el DataProvider:

DataProvider dataProvider = DataProvider.Get(UIServices.KB.CurrentModel,

“MiDataProvider”);

if(dataProvider == null){}

Crear y colocar el DataProvider en el folder:

dataProvider = new DataProvider(UIServices.KB.CurrentModel);

dataProvider.Name = “MiDataProvider”;

dataProvider.Parent = folder;

dataProvider.Save();

a. Source

En éste ejemplo se debe suponer que se tiene el siguiente objeto *structured data type* como muestra la figura 7:

Name	Type	Description	Is Collection
jqSelectData		jq Select Data	☑
Item		Item	
Id	VarChar(40)	Id	☐
Descr	VarChar(255)	Descr	☐
Selected	Boolean	Selected	☐

Figura 7. SDTEjemplo. Fuente. Elaboración propia.

En la propiedad *source* del objeto *dataprovider* establecer la estructura del *structured data type*, de la siguiente forma:

dataProvider.DataProviderSource.Source = “jqSelectData\r\n{ Item\r\n { \r\n Id = ‘1’\r\n Descr = ‘Opción 1’ \r\n Selected = False \r\n } \r\n }”;

dataProvider.Save();

b. Propiedades

El objeto *dataprovider* cuenta con el método *setpropertyvalue* para establecer sus propiedades mediante código fuente, de la siguiente forma:

Propiedad: Expose as Web Service:

dataProvider.SetPropertyValue(Properties.DPRV.ExposeAsWebService, true);

Propiedad: CollectioName:

dataProvider.SetPropertyValue(Properties.DPRV.CollectionName, “jqSelectData”);

Propiedad: Output:

dataProvider.SetPropertyValue(Properties.DPRV.Output, new

KBObjectReference(jqSelectData.Key));

1.5.5 Dominio

El siguiente código ayuda a crear dominios:

Recuperar el folder:

Folder folder = Folder.Get(UIServices.KB.CurrentModel, “iQueries”);

Comprobar si existe el Dominio:

Domain miDominio = Domain.Get(UIServices.KB.CurrentModel, “MiDominio”);

if(miDominio == null){}

Crear y colocar el Dominio en el folder:

Domain miDominio = new Domain(UIServices.KB.CurrentModel);

miDominio.Name = “miDominio”;

miDominio.Parent = folder;

miDominio.Type = eDBType.VARCHAR;

miDominio.Length = 256;

miDominio.Save();

1.5.6 External Object

El objeto External Object (EO) permite al desarrollador agregar funcionalidad a su aplicación. El EO debe tener asociado alguna librería que respalde su comportamiento, la librería debe ser realizada en el mismo lenguaje que la aplicación principal está siendo compilada por GeneXus Ev2.

Por ejemplo para el caso que la aplicación esté siendo compilada en Java, entonces la librería debería ser un .jar. Del mismo modo para el caso que la aplicación esté siendo compilada en C#, la librería debe ser .dll. De éste modo no se producirá ningún conflicto al momento de compilar la aplicación completa.

Para crear un *external object* se debe tener un proyecto que nos genere .dll o .jar según sea el caso.

El siguiente ejemplo mostrará una clase llamada CLauncher, el cual está hecha en C#, y el nombre del ensamblado es iLauncher.

```
namespace iLauncher
{
    public class CLauncher
    {
        public string server;
        public string dbname;
        public string dbmsport;
        public string user;
```

```
        public string password;
        public string trusted;
        public string parent;
        public string child;
        public string ns;

        private DataSet dataSet;
        private SqlDataAdapter dataAdapter;
        public CLauncher(){}

        public String ExecuteQueryToXml(string query){

            //Lógica del método

        }
    }
}
```

En la lógica del método se puede definir el comportamiento del EO.

Agregar la librería a GeneXus Ev2, ir al *menú Tools, menú contextual Application Integration*, el cual muestra las siguientes opciones:

.Net Assembly Import

Java Class Import

WSDL Import

XML Schema Import

Para el ejemplo se elegirá la opción *.Net Assembly Import*, mostrará el siguiente cuadro de diálogo y buscar la ruta donde se encuentra la librería, como muestra la figura 8:

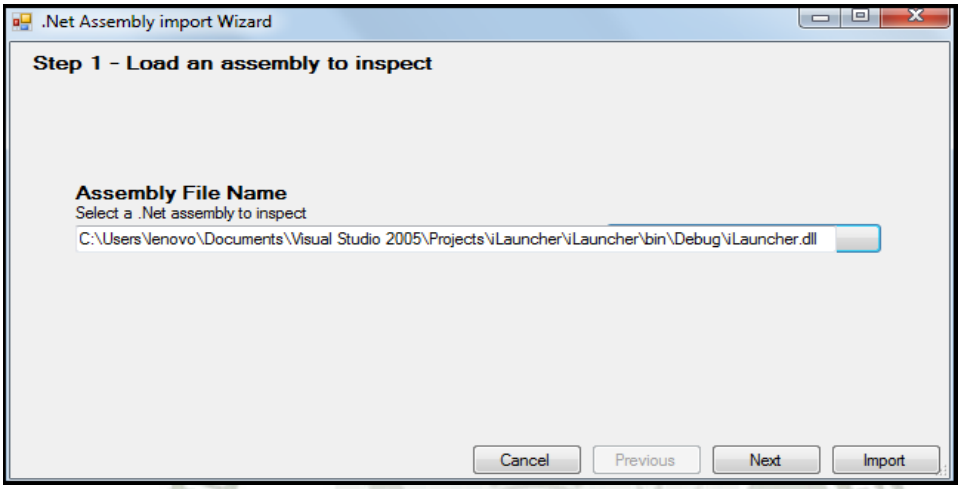


Figura 8. Importación de external object paso 1. Fuente: Elaboración propia.

Presionar *Next* y mostrará el siguiente cuadro de diálogo como se muestra en la figura 9:

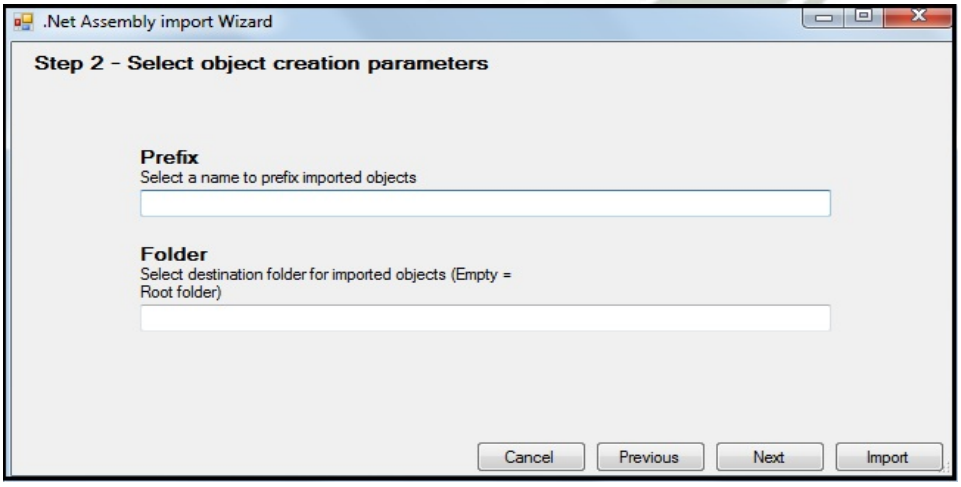


Figura 9. Importación de external object paso 2. Fuente: Elaboración propia.

No es necesario que se establezcan valores a los campos Prefix y Folder, presionar Next y se muestra el siguiente cuadro de diálogo como se ve en la figura 10:

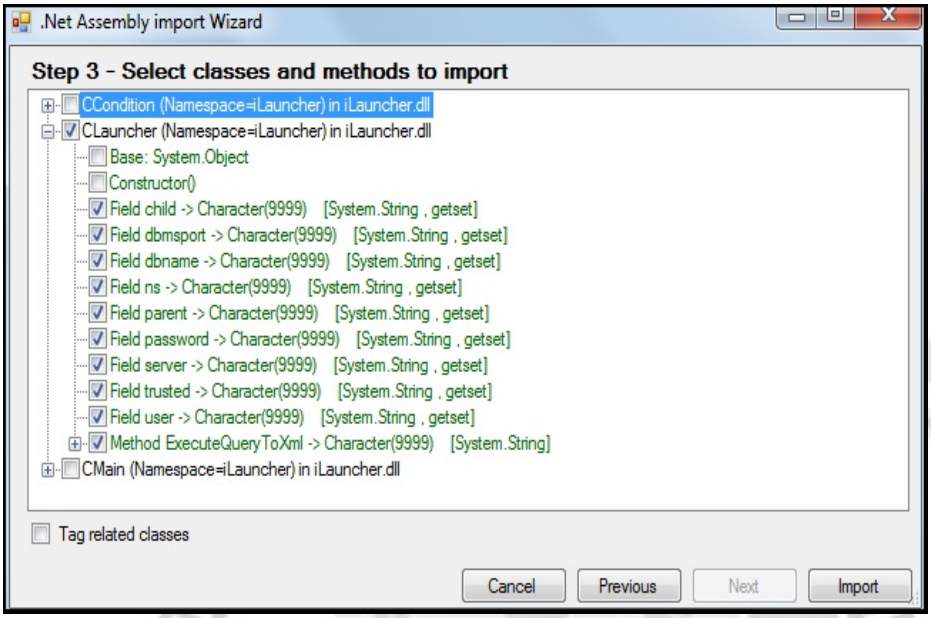


Figura 10. Importación de external object paso 3. Fuente: Elaboración propia.

En éste cuadro de diálogo muestra el contenido de la librería, se indica que la librería tiene tres clases definidas: *CCondition, CLauncher* y *CMain*. Puede elegirse cuál de ellos se importe en el *external object*, por cada clase se creará un *external object*, una vez elegido la clase puede elegirse los atributos y métodos a importar. Luego presionar *Import*.

NOTA: Para que los atributos o métodos se muestren tienen que ser declarados como *public* en la librería, caso contrario se le declaran como private.

La siguiente figura 11 muestra el *external object* creado de la clase *CLauncher*:

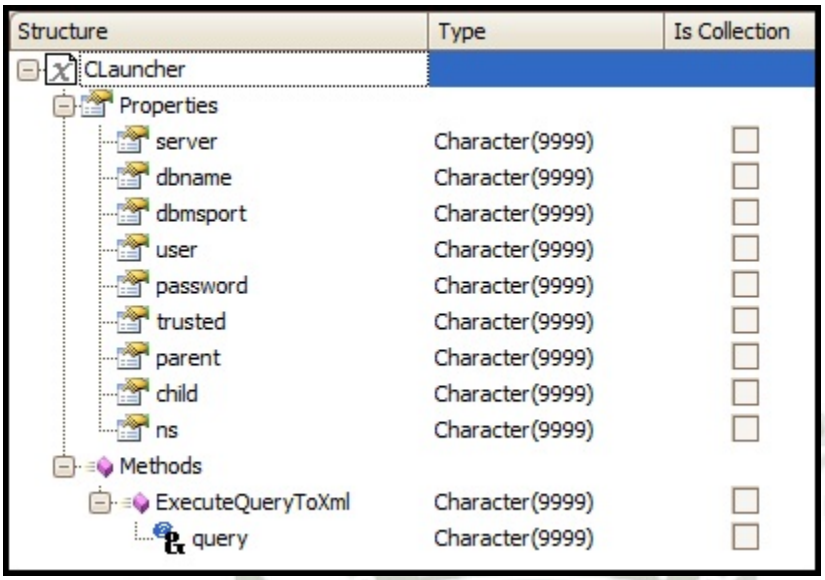


Figura 11. External object importado. Fuente: Elaboración propia.

Note que por cada atributo en la clase se creó una propiedad en el *external object* y por cada método se creó uno. Observe que los siguientes atributos no se importaron en el external object:

private DataSet dataSet;

private SqlDataAdapter dataAdapter;

Fíjese en la figura 12 las propiedades del *external object*, el *Name*, el *Namespace* y el *AssemblyName* son las mismas que la librería.

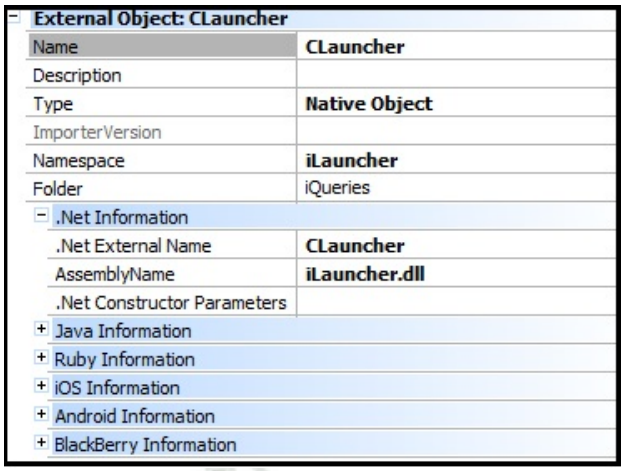


Figura 12: Propiedades del external object importado. Fuente: Elaboración propia.

NOTA: Una vez agregado el *external object* es necesario que la librería (en éste caso iLauncher.dll) se copie a la carpeta *bin* del proyecto:

Ejemplo: D:\Tesis\Proyecto\Rostros\CSharpModel\web\bin

Donde:

D:\Tesis\Proyecto: Es el directorio donde está ubicado el proyecto.
Rostros: Nombre del proyecto.

Luego de importar el *external object* es necesario que se haga un *Rebuild All* en el proyecto principal.

a. ¿Cómo utilizamos el External Object?

Se crea una variable del tipo de *external object* que se ah importado, en el objeto web panel, como se muestra en la figura 13:

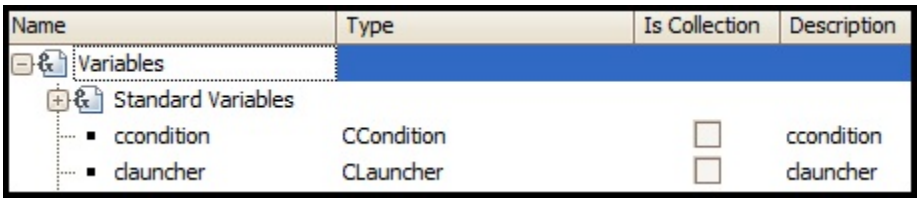


Figura 13: Variable del external object en un web panel. Fuente: Elaboración propia.

a. En la parte Events del Web Panel:

   Establecer valores para las propiedades de la nueva variable.

   &clauncher.parent = 'SdtQueryRos'

   &clauncher.child = 'SdtQueryRos.Item'

   &clauncher.ns = 'Rostros'

   &sqlQuery = 'select * from ros'

b. Invocar al método de la variable.

   &sqlResult = &clauncher.ExecuteQueryToXml(&sqlQuery)

1.6 Obtener tablas y campos

En el siguiente ejemplo se muestra como obtener las tablas y campos de la base de datos.

Obtenemos todas las tablas:

```
IEnumerator<Table> tables =
Table.GetAll(UIServices.KB.CurrentModel).GetEnumerator();
```

Recorremos todas las tablas:

```
while(tables.MoveNext()){
        Table table = tables.Current;
}
```

Obtenemos todos los campos:

```
BaseCollection<TableAttribute> atts = table.TableStructure.Attributes;
```

Recorremos todos los campos:

```
 foreach(TableAttribute att in atts){
        string nameTable = att.Table.Name;
        string nameAttribute = att.Name;
        string descriptionAtributte = att.Attribute.Description;
}
```

CAPÍTULO II

1. EXTENSIÓN PARA CONSULTAS DINÁMICAS EN GENEXUS.

El presente capítulo está dividido en dos secciones. La primera sección trata el análisis y diseño de la extensión, donde se describen los requerimientos tanto funcionales como no funcionales, los casos de uso y se muestran los diagramas de secuencia y de clases. La segunda sección describe las clases que se generaron en el desarrollo de la extensión y muestra el código fuente de la misma.

1.1.Análisis y Diseño

1.1.1. Requerimientos Funcionales

RQ – 01	Generar consulta
DESCRIPCIÓN: El Usuario podrá generar la consulta con las tablas y atributos que elija.	
Actor(es): Usuario (Desarrollador)	

RQ – 02	Ejecutar consulta en tiempo de diseño
DESCRIPCIÓN: El Usuario podrá ejecutar la consulta en tiempo de diseño.	
Actor(es): Usuario (Desarrollador)	

RQ – 03	Guardar consulta
DESCRIPCIÓN: El Usuario podrá guardar la consulta.	
Actor(es): Usuario (Desarrollador)	

RQ – 04	Ver consulta
DESCRIPCIÓN: El Usuario podrá visualizar las consultas que están almacenadas.	
Actor(es): Usuario (Desarrollador)	

RQ – 05	Limpiar objetos
DESCRIPCIÓN: El Usuario podrá deshacer todas las acciones realizadas, limpiando todos los objetos que se muestran en pantalla.	
Actor(es): Usuario (Desarrollador)	

RQ – 06	Ejecutar consulta en tiempo de ejecución
DESCRIPCIÓN: El Usuario podrá invocar la ejecución de la consulta en tiempo de ejecución de la aplicación.	
Actor(es): Usuario (Usuario Final)	

1.1.2. Requerimientos No Funcionales

- La invocación del método de ejecución de la consulta debe ser programada según la lógica del desarrollador.
- El resultado de la consulta debe ser devuelto en formato de lenguaje de marcas extensible (XML por sus siglas en inglés eXtensible Markup Language).

- Para visualizar la consulta en la ventana de la extensión debe ser guardada en un archivo .xml.
- La consulta debe ser guardada en el objeto *procedure* de GeneXus para obtenerla cuando se le requiera.
- El resultado de la consulta debe ser tomado y mostrado por el objeto GeneXus Structured Data Type.

1.1.3. Casos de Uso

a. CU 01 - Iniciar extensión

Nombre:	Iniciar Extensión
Descripción:	El desarrollador instala la extensión, importa la librería y luego hace Rebuild en la aplicación.
Actor(es):	Desarrollador
Pre-Condiciones:	Descromprimir la extensión. Iniciar GeneXus Ev2.
Eventos:	Copiar la librería en la carpeta bin del proyecto.
Post-Condiciones:	La extensión crea el directorio \iQueries\admin en la carpeta del proyecto principal. La extensión crea el archivo iQueriesConfig.xml en el directorio \iQueries. La extensión crea el objeto Folder iQueries dentro del Folder View en GeneXus Ev2. La extensión crea el objeto ProcedureiQueriesConfig en el folder iQueries.

b. CU 02 - Definir consulta

Nombre:	Definir Consulta
Descripción:	El desarrollador elige los elementos de la consulta, prueba y guarda la consulta.
Actor(es):	Desarrollador
Pre-Condiciones:	

Eventos:	
Post-Condiciones:	La extensión crea un archivo .xml en el directorio \iQueries\admin La extensión genera los siguientes objetosGeneXus: Procedure para almacenar la consulta. Structured Data Type para mostrar resultado de la consulta.

c. CU 03 - Programar con extensión

Nombre:	Programar con Extensión
Descripción:	El desarrollador crea los objetos necesarios para su programación en GeneXus Ev2, invoca el método de la extensión para la ejecución de la consulta y recupera el resultado en un objeto GeneXus.
Actor(es):	Desarrollador
Pre-Condiciones:	
Eventos:	Crear variables de los tipos de dato *CCondition y CLauncher*.
Post-Condiciones:	

d. CU 04 - Realizar consulta

Nombre:	Realizar Consulta
Descripción:	El usuario final elige los parámetros de búsqueda y solicita consultar.
Actor(es):	Usuario final
Pre-Condiciones:	
Eventos:	
Post-Condiciones:	

1.1.4. Diagrama de caso de uso

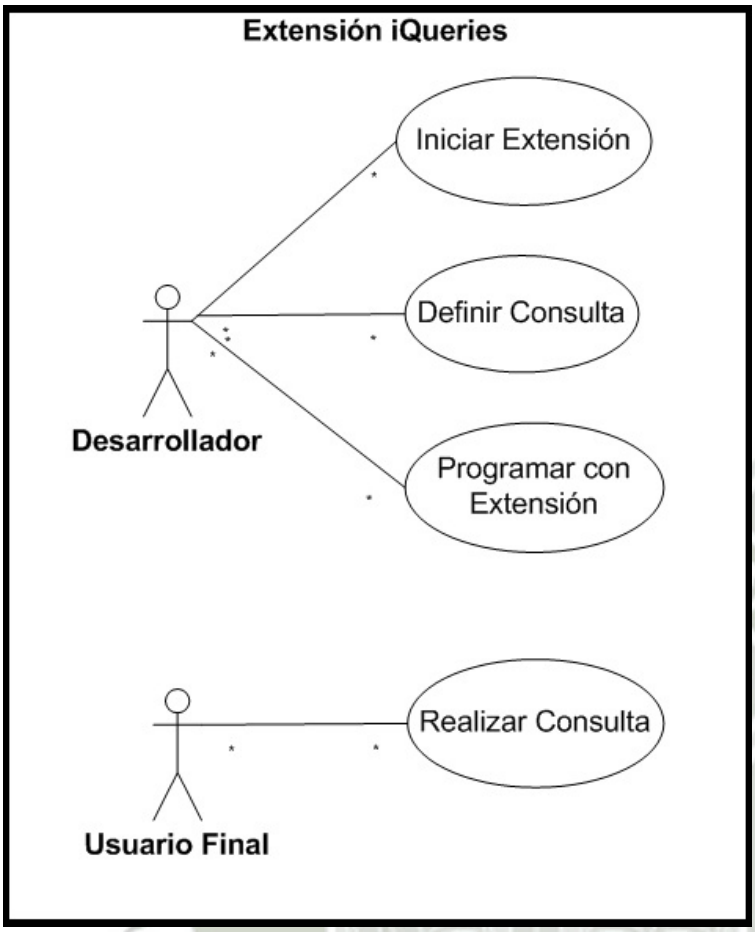


Figura 14. Diagrama de caso de uso. Fuente: Elaboración propia.

1.1.5. Diagramas de secuencia

a. Iniciar extensión

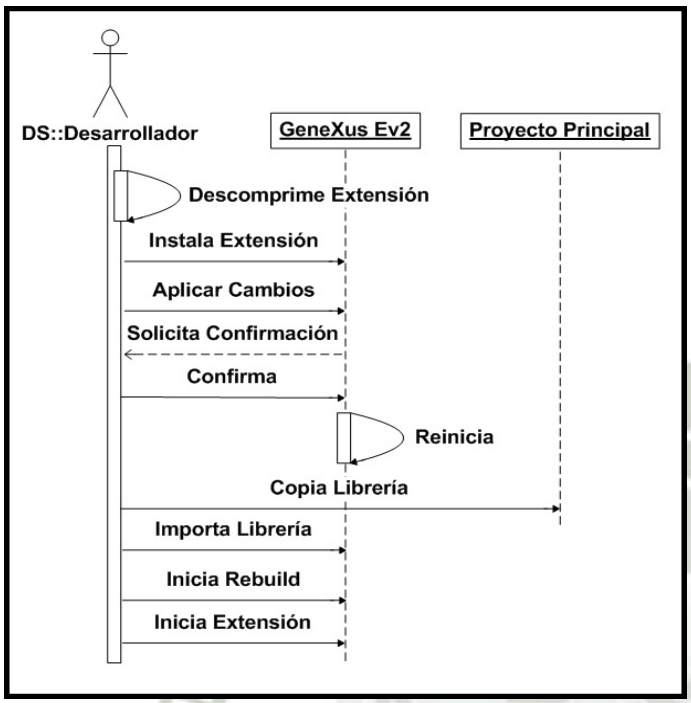


Figura 15. Diagrama de secuencia iniciar extensión. Fuente: Elaboración propia.

b. Definir consulta

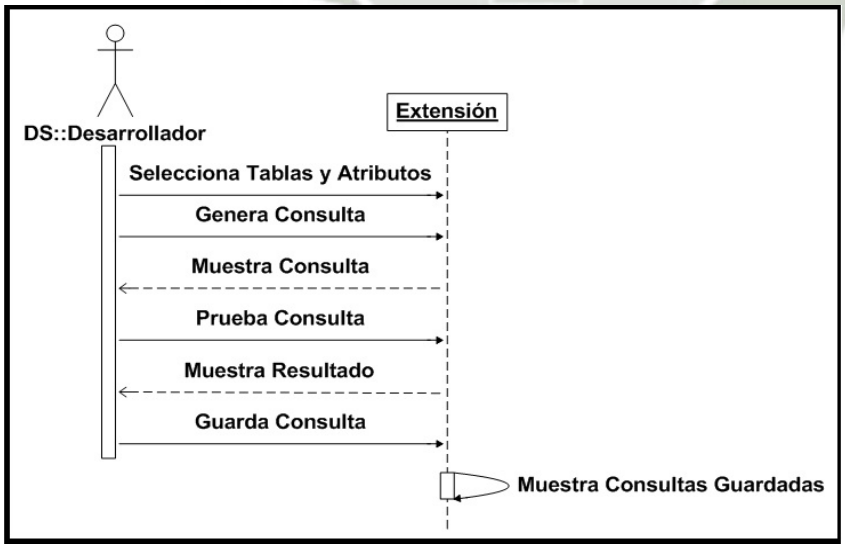


Figura 16. Diagrama de secuencia definir consulta. Fuente: Elaboración propia.

c. Programar con extensión

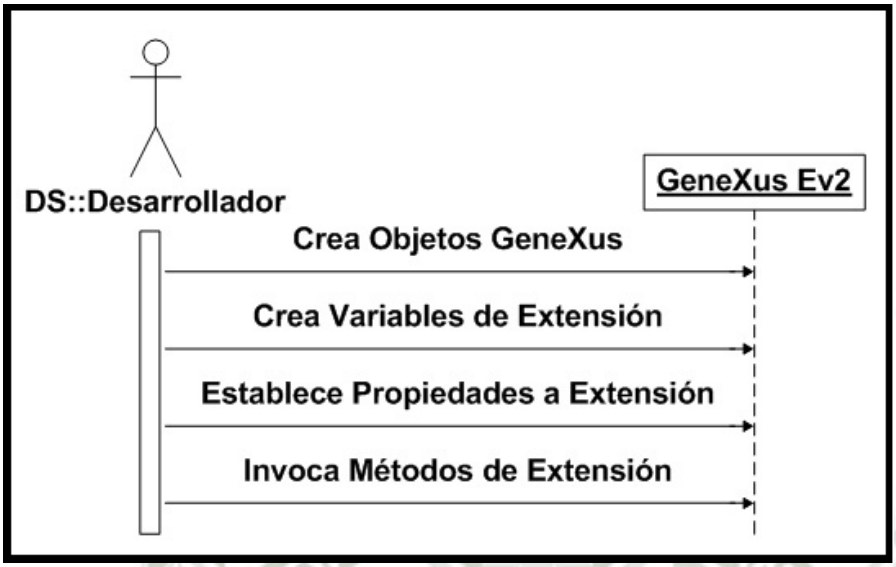


Figura 17. Diagrama de secuencia programar con extensión. Fuente: Elaboración propia.

d. Realizar consulta

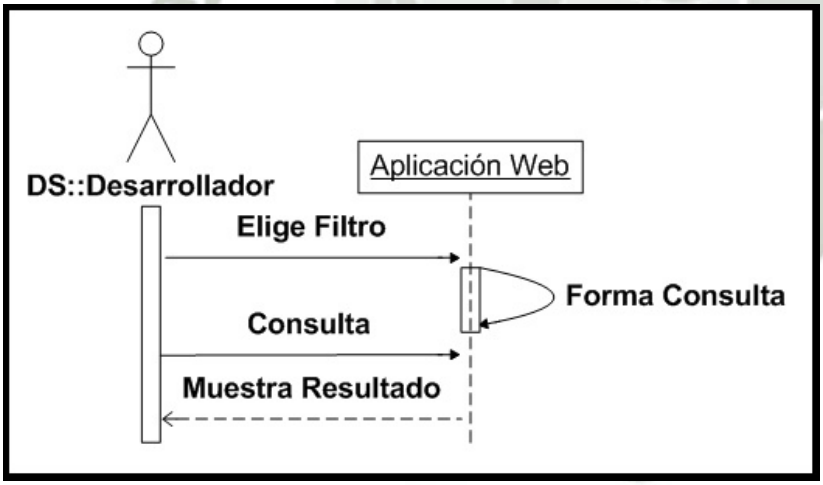


Figura 18. Diagrama de secuencia realizar consulta. Fuente: Elaboración propia.

1.1.6. Diagrama de clases

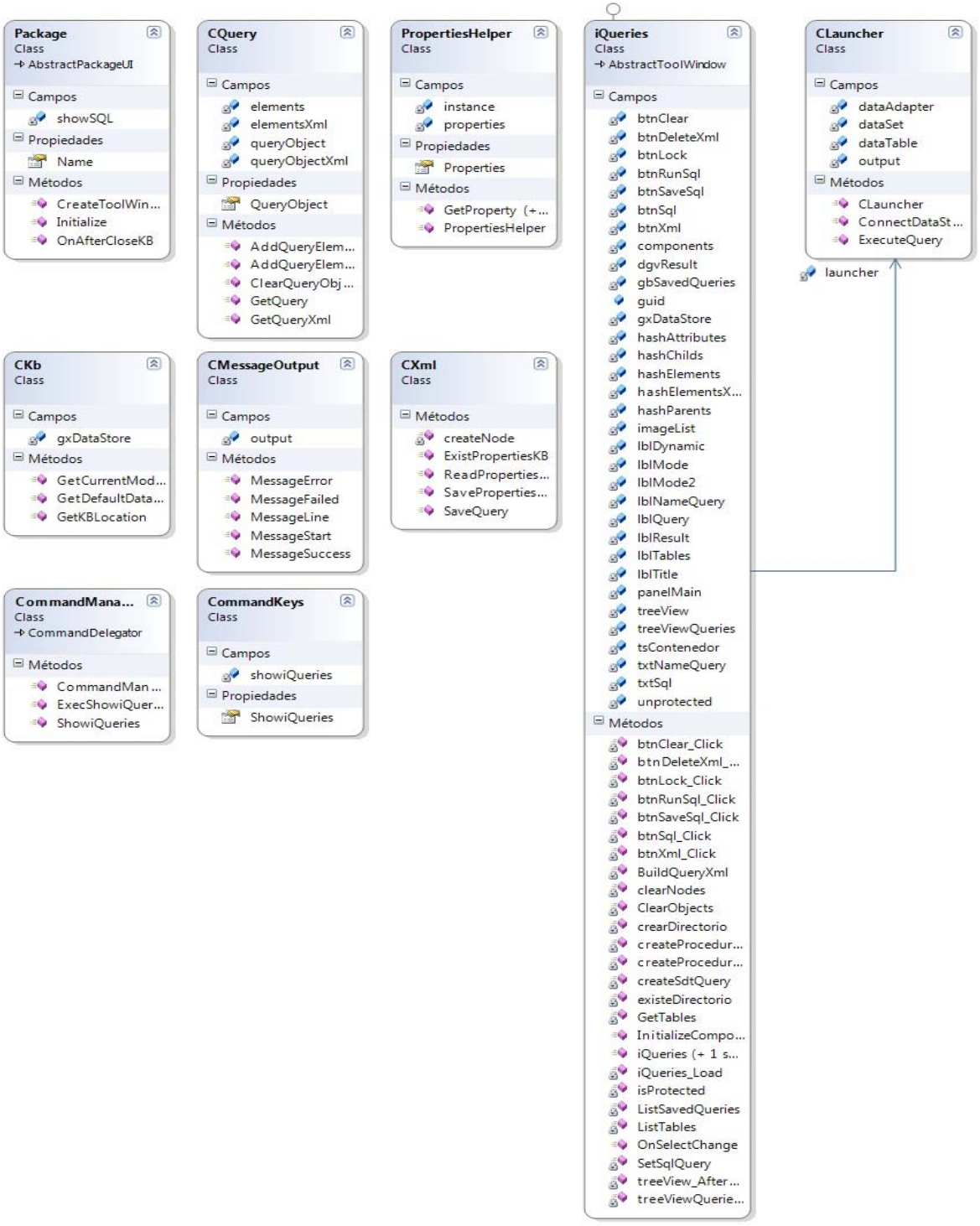


Figura 19. Diagrama de clases. Fuente: Elaboración propia.

1.2.Descripción de archivos – Código fuente de la extensión.

Para mayor especificación del código fuente, favor revisar el Anexo B. El siguiente árbol muestra los archivos que fueron creados para la creación de la extensión.

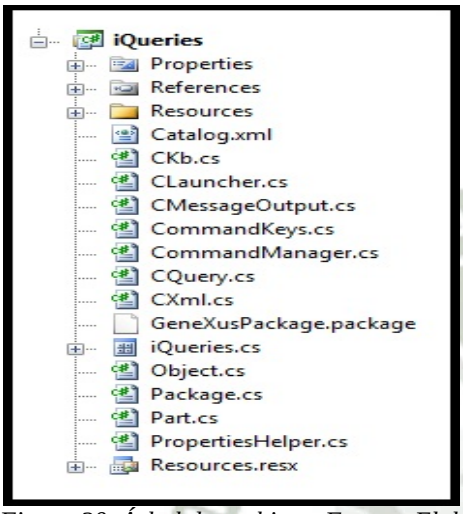


Figura 20. Árbol de archivos. Fuente: Elaboración propia.

a. GeneXusPackage.package.- Archivo de configuración para establecer el comportamiento de la extensión en la IDE de GeneXus Ev2.

b. Ckb.cs.- Clase utilitaria para obtener los valores del proyecto actual.

Nombre	Tipo	Parámetros	Tipo Dato	Miembro	Acceso	Descripción
gxDataStore	Atributo		GxDataStore	Static	Publico	Instancia dataStore
GetDefaultDataStore	Método		GxDataStore	Static	Publico	Devuelte el dataStore por defecto
GetCurrentModel	Método		KBModel	Static	Publico	Devuelve el modelo del proyecto
GetKBLocation	Método		String	Static	Publico	Devuelve la ruta del proyecto

c. CLauncher.cs.- Clase encargada de ejecutar la consulta generada.

Nombre	Tipo	Parámetros	Tipo Dato	Miembro	Acceso	Descripción
dataTable	Atributo		DataTable		Prívate	Contiene los datos del resultado de la consulta
dataAdapter	Atributo		SqlDataAdapter		Prívate	Obtiene los datos de la consulta y llena a datatable.
ConnectDataStore	Método		SqlConnection		Publico	Obtiene la conexión a la base de datos.
ExecuteQuery	Método	Query:String	DataTable		Publico	Ejecuta la consulta y devuelve los datos.

d. CMessageOutput.cs.- Clase utilitaria para mostrar mensajes en la venta output de GeneXus Ev2.

Nombre	Tipo	Parámetros	Tipo Dato	Miembro	Acceso	Descripción
Output	Atributo		IOutputService	Static	Public	Muestran mensajes en la ventana output de GeneXus Ev2.
MessageStart	Método	Message:String	Void	Static	Public	Muestra mensaje de bienvenida
MessageSuccess	Método	Message:String	Void	Static	Public	Muestra mensaje de éxito.
MessageFailed	Método	Message:String	Void	Static	Public	Muestra mensaje de fracaso.
MessageError	Método	Message:String	Void	Static	Public	Muestra mensaje de error.
MessageLine	Método	Message:String	Void	Static	Public	Agrega una línea de mensaje.

e. CommandKeys.cs.- Clase para iniciar el objeto extensión en la IDE de GeneXus Ev2.

Nombre	Tipo	Parámetros	Tipo Dato	Miembro	Acceso	Descripción
showiQueries	Atributo		CommandKey	Static	Prívate	Instancia el objeto extensión.
ShowiQueries	Propiedad		CommandKey	Static	Public	Devuelve el comando.

f. CQuery.cs.- Clase para gestionar la generación de la consulta.

Nombre	Tipo	Parámetros	Tipo Dato	Miembro	Acceso	Descripción
elements	Atributo		HashTable	Static	Prívate	Almacena los elementos de la consulta.
elementsXml	Atributo		HasTable	Static	Prívate	Almacena los elementos de la consulta guardada.
queryObject	Atributo		QueryObject	Static	Prívate	Almacena los elementos de la consulta y genera la consulta.
queryObjectXml	Atributo		QueryObject	Static	Prívate	Almacena los elementos de la consulta y genera la consulta guardada.
AddQueryElement	Método	queryElement:QueryElement	Void	Static	Public	Almacena los elementos de la consulta.
AddQueryElementXml	Método	queryElement:QueryElement	Void	Static	Public	Almacena los elementos de la consulta guardada.
GetQuery	Método	hashElements:HashTable	String	Static	Public	Devuelve la consulta generada.
GetQueryXml	Método	hashElements:HashTable	String	Static	Public	Devuelve la consulta guardada de nuevo generada.
ClearQueryObject	Método		Void	Static	Public	Limpia los elementos de la consulta.

g. CXml.cs.- Clase utilitaria para generar archivos .xml.

Nombre	Tipo	Parámetros	Tipo Dato	Miembro	Acceso	Descripción
SavePropertiesKB	Método	gxDataStore:GxDataStore	Void	Static	Public	Guarda la configuración del servidor en un archivo xml.
createNode	Método	xmlTextWriter:XmlTextWriter server:String dbname:String dbms_port:String user:String password:String trusted:String	Void	Static	Prívate	Método auxiliar a SavePropertiesKB.
ReadPropertiesKB	Método		String[]	Static	Public	Lee del archivo xml la configuración del servidor.
ExistPropertiesKB	Método		Bool	Static	Prívate	Comprueba si el archivo de configuración existe.
SaveQuery	Método	hashParents:HashTable hashChild:HashTable nameQuery:String query:String	Bool	Static	Public	Guarda la consulta generada en un archivo xml.

h. iQueries.cs.- Clase principal que tiene toda la lógica de la extensión.

Nombre	Tipo	Parámetros	Tipo Dato	Miembro	Acceso	Descripción
Guid	Atributo		Guid	Static	Public	Objeto para instanciar la extensipon en la IDE de GeneXus Ev2.

gxDataStore	Atributo		GxDataStore		Private	Instancia del dataStore por defecto.
Launcher	Atributo		CLauncher		Private	Objecto que ejecuta la consulta.
btnRunSql	Atributo		ToolStripButton		Private	Botón para ejecutar la consulta.
btnSaveSql	Atributo		ToolStripButton		Private	Botón para guarda la consulta.
btnSql	Atributo		ToolStripButton		Private	Botón para generar la consulta.
btnXml	Atributo		ToolStripButton		Private	Botón para obtener todas las consultas guardadas.
btnClear	Atributo		ToolStripButton		Private	Botón para limpiar todos los objetos en pantalla.
tsContenedor	Atributo		ToolStrip		Private	Objeto contenedor de los botones.
txtSql	Atributo		TextBox		Private	Texto para mostrar la consulta generada.
txtNameQuery	Atributo		TextBox		Private	Texto para colocar nombre a la consulta.
imageList	Atributo		ImageList		Private	Lista de imágenes para los controles.
Components	Atributo		IContainer		Private	Objeto contenedor para los objetos en pantalla.
treeView	Atributo		TreeView		Private	Objeto para mostrar las tablas y atributos de la base de datos.
gbSaverdQueries	Atributo		GroupBox		Private	Objeto contenedor del árbol que muestra las consultas guardadas.

dgvResult	Atributo		DataGridView		Private	Grilla que muestra el resultado de la consulta.
panelMain	Atributo		Panel		Private	Objeto contenedor de los objetos en pantalla.
treeViewQueries	Atributo		TreeView		Private	Objeto que muestra todas las consultas guardadas.
btnLock	Atributo		Button		Private	Botón para deshabilitar la edición de la consulta.
btnDeleteXml	Atributo		Button		Private	Botón para eliminar la consulta guardada seleccionada.
Unprotected	Atributo		Bool		Private	Flag que identifica que los objetos están deshabilitados.
lblTables	Atributo		Label		Private	Etiqueta para mostrar el texto Tables.
lblQuery	Atributo		Label		Private	Etiqueta para mostrar el texto Query:
lblResult	Atributo		Label		Private	Etiqueta para mostrar el texto Result.
lblNameQuery	Atributo		Label		Private	Etiqueta para mostrar el texto Name Query.
lblTitle	Atributo		Label		Private	Etiqueta para mostrar el texto iQueries:
lblDynamic	Atributo		Label		Private	Etiqueta para mostrar el texto Dynamic Queries.
lblMode	Atributo		Label		Private	Etiqueta para mostrar el texto Mode.

lblMode2	Atributo		Label		Private	Etiqueta para mostrar el texto Unprotected/Protected.
hashAttributes	Atributo		HashTable		Private	Estructura para almacenar todos los atributos de las tablas.
hashParents	Atributo		HashTable		Private	Estructura para almacenar las tablas elegidas.
hashChilds	Atributo		HashTable		Private	Estructura para almacenar los atributos elegidos.
hashElements	Atributo		HashTable		Private	Estructura para almacenar los elementos de la consulta.
hashElementsXml	Atributo		HashTable		Private	Estructura para almacenar los elementos de la consulta guardada.
iQueries	Constructor				Public	Instancia todos los objetos y suscribe el guid de la clase en los servicios de la IDE GeneXus Ev2.
iQueries	Constructor			Static	Private	Obtiene el id de la guid.
InitializeComponent	Método		Void		Public	Inicializa todos los objetos y establece sus propiedades.
iQueries_Load	Evento	Sender:object e:EventArgs	Void		Private	Crea y comprueba las carpetas de la extensión y el archivo de configuración de la base

						de datos en xml y procedure.
btnXml_Click	Evento	Sender:object e:EventArgs	Void		Private	Obtiene las consultas guardadas.
btnSql_Click	Evento	Sender:object e:EventArgs	Void		Private	Genera la consulta.
btnRunSql_Click	Evento	Sender:object e:EventArgs	Void		Private	Ejecuta la consulta
btnSaveSql_Click	Evento	Sender:object e:EventArgs	Void		Private	Guarda la consulta.
btnClear_Click	Evento	Sender:object e:EventArgs	Void		Private	Limpia los objetos en pantalla.
btnLock_Click	Evento	Sender:object e:EventArgs	Void		Private	Habilita y deshabilita los objetos para no editar la consulta.
treeView_AfterCheck	Evento	Sender:object e:EventArgs	Void		Private	Selecciona las tablas y atributos para la consulta.
btnDeleteXml_Click	Evento	Sender:object e:EventArgs	Void		Private	Elimina la consulta guardada seleccionada.
treeViewQueries_AfterSelect	Evento	Sender:object e:EventArgs	Void		Private	Muestra la consulta guardada.
isProtected	Método		Void		Private	Verifica si los objetos están deshabilitados.
clearNodes	Método		Void		Private	Limpia las tablas y atributos que fueron seleccionados.
SetSqlQuery	Método	sqlQuery:String	Void		Private	Establece el texto a txtsql con la consulta.
ClearObjects	Método		Void		Private	Limpia los objetos en pantalla.

OnSelectChange	Método	sc:ISelectionContainer	Void		Private	Retorna verdadero (método sobrecargado).
crearDirectorio	Método		Void		Private	Crea el directorio de la extensión en el directorio del proyecto principal.
existeDirectorio	Método		Bool		Private	Comprueba si el directorio ya ha sido creado.
createProcedureConfig	Método		Void		Private	Crea el objeto Procedure con la configuración del servidor.
createProcedureQuery	Método		Bool		Private	Crea el objeto Procedure que almacena la consulta generada.
createSdtQuery	Método		Bool		Private	Crea el objeto Structured Data Type con la estructura de la consulta.
GetTables	Método		Void		Private	Obtiene las tablas y atributos.
ListSavedQueries	Método		Void		Private	Obtiene las consultas guardadas.
ListTables	Método		Void		Private	Recupera todas las tablas y atributos de la base de datos.
BuildQueryXml	Método		Void		Private	Construye la consulta guardada.

i. PropertiesHelper.cs.- Clase utilitaria que obtiene las propiedades del servidor.

Nombre	Tipo	Parámetros	Tipo Dato	Miembro	Acceso	Descripción
Instance	Atributo		Object		Prívate	Objeto genérico.
PropertiesHelper	Constructor	Obj:object			Public	Instancia al atributo instance.
Properties	Atributo		PropertyDescriptorCollection		Prívate	
Properties	Propiedad		PropertyDescriptorCollection		Public	Obtiene el atributo properties
GetProperty	Método	propName:String	Object		Public	Obtiene los valores de una propiedad.
GetProperty<T>	Método Genérico	propName:String	T		Public	Obtiene la instancia de cualquier objeto.

CAPÍTULO III
VALIDACIÓN

1. Variables Dependientes

- Extensión para la realización de consultas dinámicas.
- Guía de implementación de extensiones.

Indicadores

- Fácil instalación de la extensión.
- Fácil implementación de consultas dinámicas con la extensión.
- Fácil comprensión de la guía de implementación.
- Ejemplos prácticos en la guía de implementación.
- Utilidad de la extensión.

Para la evaluación de los indicadores de las variables dependientes se elaboró una encuesta (ver Anexo C), en la tabla 1 se muestra a las personas que fueron encuestadas:

Tabla 1

Analistas – Desarrolladores GeneXus Encuestados

Nombre	Ocupación
Eduardo Jorge Jasaui Torres	Distribuidor GeneXus
Alfredo Zea García Calderón	Analista – Programador GeneXus
Fiorella Rivera Calderón	Analista – Programador GeneXus
Ana Belén Arista Huenonte	Analista – Programador GeneXus
Lilian Loayza	Analista – Programador GeneXus
Virgilio José Torres Otoya	Analista – Programador GeneXus
Juan Andres Quintanilla Calderón	Analista – Programador GeneXus
Jean Kharlo Pinto Espejo	Analista – Programador GeneXus

Mersalí Araujo Lara	Analista – Programador GeneXus
Nathaly Duschanna Palza Monrroy	Analista – Programador GeneXus
Christian Alain Revilla Arroyo	Analista – Programador GeneXus
Milagros Gorvenia Arenas	Analista – Programador GeneXus
Dennis René Arenas del Carpio	Analista – Programador GeneXus
Paola Rocío Salas Valencia	Analista – Programador GeneXus
Beatriz Irene Angelino Mendoza	Analista – Programador GeneXus
Yordan Pol Yampi Enciso	Analista – Programador GeneXus

Fuente: Elaboración Propia.

La realización de las encuestas se hizo de forma manual, y se escogieron a las personas anteriormente mencionadas porque tienen experiencia en el análisis y desarrollo de sistemas de información con la herramienta GeneXus.

Se realizaron los siguientes pasos para la toma de encuesta:

- Se hizo una breve introducción del problema y de la solución propuesta.
- Se mostró un video tutorial que explica la instalación y uso de la extensión.
- Para su prueba se dio acceso a una laptop con la herramienta trial de GeneXus y extensión instaladas.
- Se dio acceso al documento de la guía de implementación.
- Se explicaron los ejemplos de la guía de implementación.
- Se tomó la encuesta.

A continuación se mostrarán los resultados de los indicadores medidos por la encuesta:

a. Fácil instalación de la extensión.- Para comprobar la facilidad de la instalación se realizó la siguiente pregunta:

   ¿Qué le pareció la instalación de la extensión?

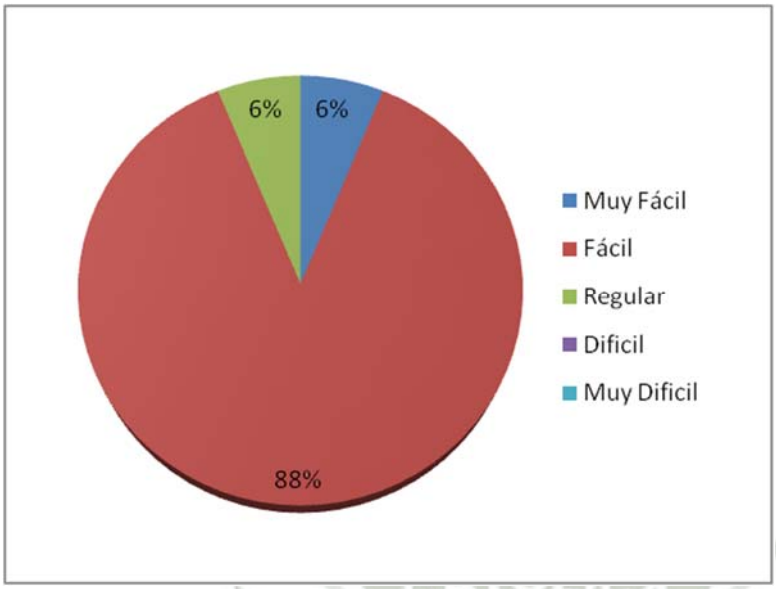


Figura 21. Facilidad de instalación. Fuente: Elaboración propia.

Interpretación: El 6% de los encuestados cree que la instalación es muy fácil, mientras que el 88% cree que es fácil y el 6% lo cree en un nivel intermedio.

b. Fácil implementación de consultas dinámicas con la extensión.- Para comprobar la facilidad de implementación de consultas con la extensión se realizó la siguiente pregunta:

   ¿Qué le pareció la implementación para realizar consultas dinámicas con la extensión?

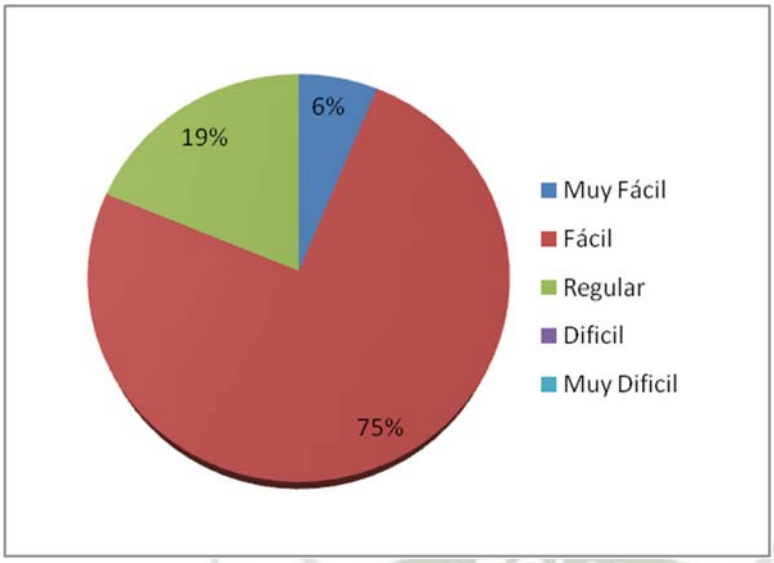


Figura 22. Facilidad de implementación. Fuente: Elaboración propia.

Interpretación: el 6% de los encuestados cree que la implementación de consultas dinámicas con la extensión es muy fácil, mientras que un 75% cree que es fácil y el 19% en un nivel intermedio.

c. Fácil comprensión de la guía de implementación.- Para comprobar la facilidad de comprensión de la guía de implementación se realizó la siguiente pregunta:

¿Cómo le pareció la comprensión de la guía de extensiones?

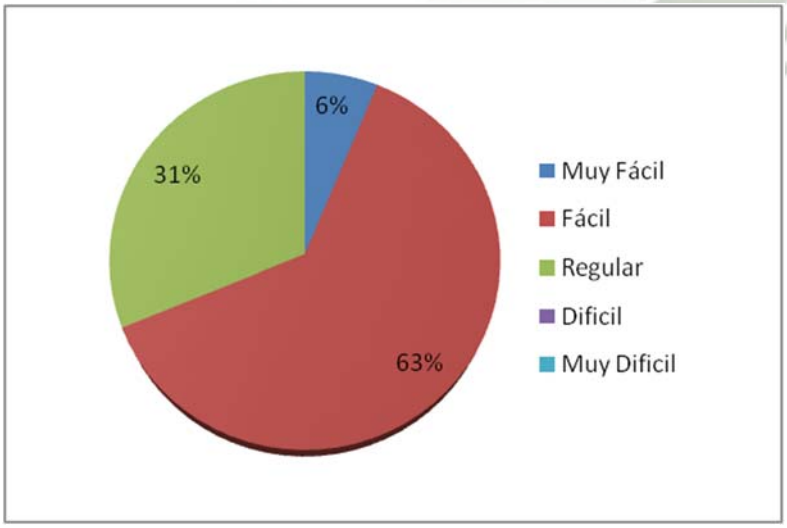


Figura 23. Comprensión de la guía. Fuente: Elaboración propia.

Interpretación: Los encuestados creen en un 6% que la comprensión es muy fácil, mientras que el 63% cree que es fácil y un 31% lo cree en un nivel intermedio.

d. Ejemplos prácticos en la guía de implementación.- Para comprobar la practicidad de los ejemplos mostrados en la guía de implementación se realizó la siguiente pregunta:

¿Qué le pareció los ejemplos de creación de objetos GeneXus en la guía de extensiones?

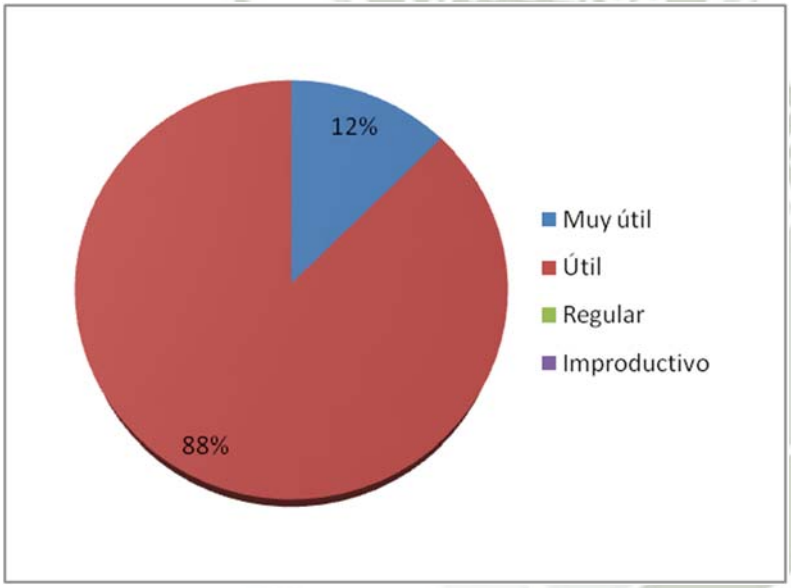


Figura 24. Ejemplos de creación de objetos. Fuente: Elaboración propia.

Interpretación: El 12% de los encuestados cree que son muy útiles los ejemplos de la guía de extensiones, mientras que un 88% cree son útiles.

e. Utilidad de la extensión.- Para comprobar la utilidad de la extensión para la implementación de consultas dinámicas en aplicaciones GeneXus, se consulto la opinión de los encuestados con la siguiente pregunta:

¿Cómo califica en grado de utilidad a la extensión?

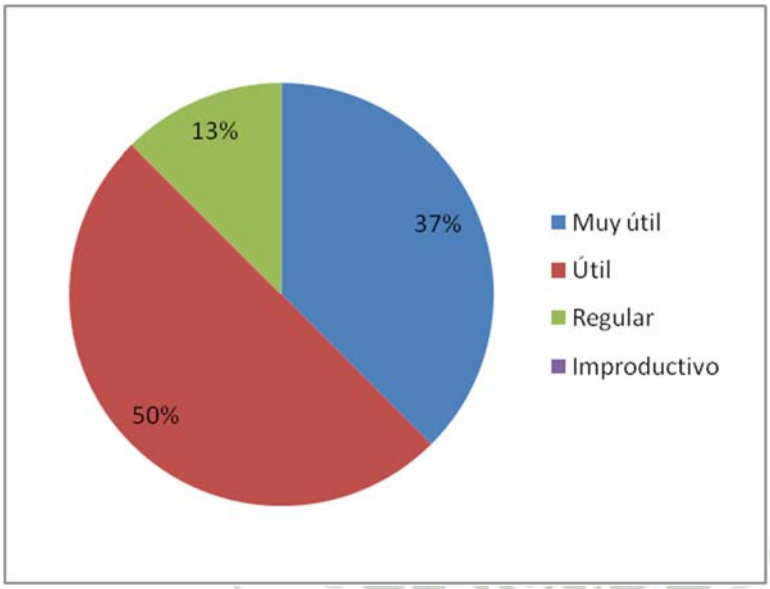


Figura 25. Utilidad de la extensión. Fuente: Elaboración propia.

Interpretación: El 37% de los encuestados cree que la utilidad de la extensión para la realización de consultas dinámicas en GeneXus es muy útil, mientras que el 50% cree que es útil y un 13% lo cree en un nivel intermedio.

CONCLUSIONES

PRIMERA la instalación de la extensión es fácil de realizar por las personas que se ocupan del análisis y desarrollo de sistemas de información en GeneXus Ev2.

SEGUNDA la extensión ayuda a implementar con facilidad consultas dinámicas en aplicaciones hechas por la herramienta GeneXus Ev2

TERCERA la guía de implementación de extensiones ayudará a personas interesadas en desarrollar su propia extensión debido a su fácil comprensión y sus ejemplos prácticos

QUINTA la extensión ayudará a desarrolladores interesados en implementar consultas dinámicas en aplicaciones GeneXus debido a su gran utilidad.

SEXTA la programación de extensiones para la herramienta GeneXus Ev2 es bastante similar a la de aplicaciones convencionales.

SÉPTIMA la utilización de las librerías GeneXus para la implementación de nuevas extensiones no es complicada.

Trabajos Futuros

1 Implementar la librería para la ejecución de la consulta en los lenguajes Java y Ruby, si la aplicación principal está siendo compilada por GeneXus en cualquier de esos lenguajes.

2 Mejorar el tiempo de respuesta de la consulta con grandes cantidades de información.

3 Agregar detalles de paginación de recuperación de la información al momento de realizar la consulta.

Referencias Bibliográficas

- Blog de Marcos Crispino. (2009). *Identificación de objetos en GeneXus X*. Recuperado de http://blog.marcoscrispino.com/2009/03/identificacion-de-objetos-en-genexusx.html

- CommunityWiki (2006). *GeneXus X/GeneXus Platform SDK, GeneXus Paltform Software Development Kit*. Recuperado de: http://wiki.gxtechnical.com/commwiki/servlet/hwiki?GeneXus+X%2FGeneXus+Platform+SDK,

- CommunityWiki (2008). *How to Implement a Default Provider for KBObject Parts*. Recuperado de: http://wiki.gxtechnical.com/commwiki/servlet/hwiki?GeneXus+X%2FGeneXus+Platform+SDK,

- CommunityWiki (2012). *Category: GeneXus Extensions*. Recuperado de: http://wiki.gxtechnical.com/commwiki/servlet/hwiki?Category%3AGeneXus+Extensions,

- CommunityWiki (2012). *Category: GeneXus Extensions Documentation*. Recuperado de: http://wiki.gxtechnical.com/commwiki/servlet/hwiki?Category%3AGeneXus+Extensions+Documentation,

- GuidGenerator. (s.f.). *Online GUID Generator*. Recuperado de: http://www.guidgenerator.com/online-guid-generator.aspx

- GXtechnical (2012). *GeneXus X Platform SDK*. Recuperado de: http://www2.gxtechnical.com/portal/hgxpp001.aspx?15,8,8,O,,0,,%202707

- Marcos Crispino (2008). *Taller de GeneXus Extensions*. Recuperado de: http://www.slideshare.net/mcrispino/taller-gx-extensions-presentation

- Marcos Crispino (2008). *Taller de GeneXus Extensions Reunión del Grupo de Usuarios GeneXus Montevideo Noviembre de 2008*. Recuperado de: http://buhoonline.com/wikiiys/servlet/hwiki?file%3ATaller+de+Genexus+Extensions

Anexos

Anexo A: Proyecto de Tesis

1 Planteamiento de la investigación.

1.1 Planteamiento del problema.

GeneXus Ev2 es una herramienta para el desarrollo ágil de sistemas de información, muy utilizado en varios continentes como: Latinoamérica, Europa Occidental y Asia. GeneXus Ev2 crea aplicaciones los cuales pueden ser mudados a diferentes plataformas de forma automática, y permite la mantención eficiente frente a los cambios de la realidad de negocio de la empresa en cuestión.

GeneXus Ev2 permite el desarrollo incremental de sistemas de información debido a la base de conocimientos abstraída de la experticia de los usuarios finales por el equipo de desarrollo, para generar automáticamente el sistema de base de datos y el código fuente de los programas, permitiendo así a la gente de desarrollo tomarse más tiempo en comprender los problemas de los usuarios.

GeneXus Ev2 ofrece para el desarrollo de sus aplicaciones una sintaxis declarativa, por el cual los usuarios desarrolladores pueden definir qué datos obtener y de donde obtener, más no como obtener, convirtiéndose en ventaja para crear y mantener en forma automática el sistema de base de datos, resultando éstas tareas transparentes a la gente de desarrollo, en consecuencia limita a éstos últimos no poder definir y ejecutar consultas en el lenguaje de consulta estructurado en sus aplicaciones, con el fin de satisfacer requerimientos que necesiten realizar consultas dinámicas.

Los sistemas de información por su naturaleza deben ofrecer a los usuarios finales, si éstos lo requieren, consultas dinámicas para la obtención de datos que éstos necesiten, por consiguiente al referirse que serán consultas que estarán formándose variablemente no se

podría tener establecido todas las posibles combinaciones de los parámetros de búsqueda que el usuario final requiera. Es por ello que ésta investigación se centra en desarrollar la extensión para la herramienta GeneXus Ev2 que permita la comunicación entre los programas y el sistema de base de datos (ambos generados por GeneXus Ev2) mediante lenguaje de consulta estructurado, y también puesto que el tema de creación de extensiones entre los desarrolladores que utilizan la herramienta es relativamente nuevo y no se cuenta con documentación formal, se pretende realizar una guía de implementación para ayudar a los interesados en desarrollar más extensiones.

1.2 Objetivos de la investigación.

a. General

Desarrollar la extensión para la realización de consultas dinámicas y guía de implementación para los desarrolladores en GeneXus Ev2.

b. Específicos

- La extensión debe permitir al desarrollador su fácil instalación en el Entorno de Desarrollo Integrado de GeneXus Ev2.
- La extensión debe brindar al desarrollador una solución fácil de implementar consultas dinámicas.
- La extensión debe utilizar objetos de la herramienta GeneXus Ev2 en su propuesta de solución.
- La guía de implementación debe ofrecer ejemplos prácticos para su fácil compresión.

1.3 Preguntas de la investigación.

La presente investigación pretende implementar la extensión para construir consultas dinámicas y realizar una guía de implementación para el desarrollo de extensiones para los profesionales que se dedican al diseño e implementación de aplicaciones en GeneXus Ev2. Se pretende asegurar que los usuarios finales de un sistema de información puedan realizar consultas más flexibles a su base de datos y que la gente de desarrollo tenga una guía de partida para desarrollar sus propias extensiones, agregando así nuevas funcionalidades. Entonces de acuerdo a lo dicho anteriormente se formula la siguiente pregunta: ¿Cómo se podría implementar una extensión que agregue la funcionalidad para realizar consultas dinámicas en aplicaciones hechas por GeneXus Ev2?

Por consiguiente se desprenden las siguientes interrogantes:

- ¿Qué herramienta permite el desarrollo de extensiones para GeneXus Ev2?
- ¿Cuáles son los pasos a seguir para el desarrollo de extensiones en GeneXus Ev2?
- ¿GeneXus Ev2 permite la ejecución de consultas o se tiene que programar una rutina y en qué lenguaje?
- ¿Si en el caso que GeneXus Ev2 no permita ejecutar consultas de qué modo la extensión devolvería los resultados a la aplicación?
- ¿Cómo formaría el usuario final sus consultas?

1.4 Línea y sub-línea de investigación.

La presente investigación se encuentra en la línea de sistemas de información y bases de datos y en las sub-líneas sistemas de información y gestión de bases de datos debido a que se intenta resolver la viabilidad de poder realizar consultas dinámicas en sistemas de información hechas en la herramienta de GeneXus Ev2.

1.5 Solución propuesta.

a. Justificación e importancia.

En el actual mercado el uso de la herramienta GeneXus Ev2 cada vez se hace más popular entre las empresa que desarrollan su propio sistema de información. GeneXus Ev2 es una potente herramienta para el diseño e implementación, creando así prototipos funcionales en tiempos cortos que ayudan a la corrección de los requerimientos de los usuarios finales. Los dos puntos fuertes de GeneXus Ev2 son (a) crear sistemas de información sin importar la plataforma en el cual correrá la aplicación ya que su sintaxis declarativa puede compilarse a varios lenguajes estructurados, (b) la generación automática de sistemas de base de datos y código fuente de programas.

De acuerdo a lo mencionado anteriormente las tareas que están comprendidas dentro de la gestión del sistema de base de datos se convierten en automáticas y transparentes para la gente del equipo de desarrollo. Los sistemas de información deben ofrecer a los usuarios finales consultas lo bastante flexibles para ajustarse a sus parámetros de búsqueda, es aquí donde surge la siguiente pregunta: ¿Si la gestión de base de datos es transparente a la gente del equipo desarrollo, entonces como éstos podrían generar consultas dinámicas para que el usuario las utilice? La presente investigación trata de resolver la pregunta formulada asegurando la implementación de la extensión que agregue la funcionalidad de generar

consultas dinámicas para aplicaciones desarrolladas en la herramienta GeneXus Ev2, además de crear la guía de implementación de extensiones para los desarrolladores interesados que utilizan dicha herramienta, ésta guía les ayudará a iniciarse en la creación de extensiones, debido que éste tema no es muy conocido entre la gente que se dedica al desarrollo en GeneXus ev2 y no se cuenta con la suficiente documentación.

Podemos agregar que él que realiza la presente investigación tiene experiencia en el desarrollo de aplicaciones en GeneXus Ev2, posee un certificado en conocer la herramienta, además de contar con la asesoría necesaria, con los suficientes conocimientos y el tiempo suficiente para implementar la extensión, como así también desarrollar la guía de implementación de extensiones para los desarrolladores que tengan el interés de crear nuevas extensiones para la herramienta.

Por último como dato importante se muestra la siguiente tabla 1A (los datos fueron proporcionados por el Ing. Eduardo Jasaui Torres, Distribuidor de productos GeneXus en el Sur del Perú), que lista las empresas que utilizan la herramienta GeneXus Ev2 en la ciudad de Arequipa, el número de desarrolladores con los que cuentan y cuántos de ellos desarrollaron extensiones.

Tabla 1A
Empresas que utilizan GeneXus Ev2 en la ciudad de Arequipa

Empresa	No. Desarrolladores	No. Desarrolladores que implementaron extensiones
Franky Ricky	7	0
Papelera Panamericana S.A.	4	0
Caja Municipal Arequipa	38	0
Incalpaca TPX S.A.	11	0
GxData	4	0
Total	64	0

Fuente: Elaboración propia.

b. Descripción de la solución

La implementación de la extensión proporcionará una herramienta que permita la flexibilidad para construir consultas dinámicamente y ejecutarlas de manera simple y sencilla, con esto se ayudará a la reducción de tiempo de desarrollo a los requerimientos de sistemas de información que soliciten flexibilidad al momento de hacer consultas a su sistema de base de datos. También al proponer la guía de implementación para ayudar a los interesados en crear propias extensiones, convirtiéndose así el desarrollo de extensiones un tema más familiar, reduciendo el tiempo de investigación por parte de la gente de desarrollo. Además se tiene el conocimiento que GeneXus Ev2 posee una arquitectura abierta para agrega nuevas funcionalidades.

c. Alcances y Limitaciones

i. Alcances

- La extensión brindará una solución alterna sin licencias a la propuesta por GxQuery.
- La extensión proporcionará una interface para la definición, ejecución, visualización e eliminación de consultas.
- La extensión proporcionará un método de ingreso de los parámetros de búsqueda.
- La extensión permitirá la ejecución de la consulta en tiempo de ejecución de la aplicación.
- La Guía de implementación mostrará conceptos básicos, instalación de las librerías que se necesitan para desarrollar extensiones, ejemplos de obtención de información de las tablas de la base de datos y creación de objetos de herramienta GeneXus Ev2.

ii. Limitaciones

- La extensión depende del lenguaje que se genera la aplicación, esto es que parte de la misma será desarrollado en un lenguaje que pueda diferir del lenguaje que se decida compilar la aplicación hecha por GeneXus Ev2.
- Para la aplicación de la extensión se utilizará la versión trial de GeneXus Ev2, la cual sólo permite la creación de 140 objetos.
- No se cuenta con ejemplos prácticos en GeneXus Extensions Software Development Kit, que utilicen funciones que interesen al desarrollo de la extensión.
- Se necesita conexión a internet para la ejecución de las aplicaciones en la versión trial GeneXus Ev2.

2 Fundamentos teóricos

2.1 Estado del arte.

GxQuery es una herramienta licenciada desarrollada por la empresa uruguaya Artech, nace como respuesta a la necesidad de los usuarios finales de poder realizar consultas dinámicas en sus bases de datos operacionales. Funciona como un complemento de Excel y también a través de una interface web que permite una guía bastante útil al momento de confeccionar reportes. Actualmente se encuentra en su versión GxQuery 3.0 Upgrade #5 (GeneXus, 2012).

2.2 Bases teóricas de la investigación.

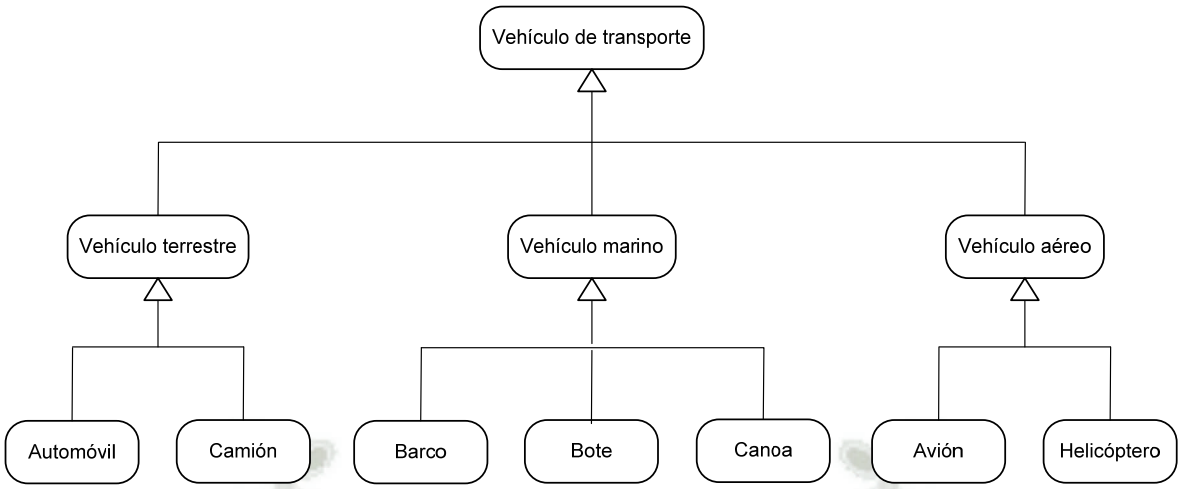
a. Programación orientada a objetos (POO).

Es una filosofía cuyo objetivo principal es la abstracción de realidades para modelar las entidades que conforman un sistema de información. Dicho de otro modo es el paradigma que abstrae los objetos del mundo real para clasificarlos en clases, definir sus atributos y los comportamientos que éstos toman en el contexto de su realidad.

i. Clases en POO.

Son niveles de abstracción que agrupan características y comportamientos de los objetos del mundo real.

Por Ejemplo:



En la figura anterior se muestra 3 niveles de abstracción de clases, como primer nivel se tiene la clase automóvil, camión, barco, bote, canoa, avión y helicóptero, al agruparlas por sus características similares se pueden clasificar en vehículos terrestres, marinos y aéreos, por último pueden generalizarse a todas ellas como la clase de vehículos de transporte.

ii. Objetos, atributos, métodos y estados.

- Objetos.- Son las instancias o representaciones físicas de las clases o entidades de la realidad modelada.
- Atributos.- Son las características que definen a los objetos de la misma clase.
- Métodos.- Son los comportamientos que se definen para los objetos de la misma clase.
- Estados.- Son los valores que toman los atributos de los objetos en un determinado tiempo.

Ejemplo:

Clase Automóvil
Atributos
- No. Llantas
- No. Puertas
- No. Pasajeros
- Color
- Modelo
- Año
- Fabricante

Métodos
- Encender
- Acelerar
- Frenar
- Apagar

Estados

	No.Llantas	No.Puertas	No.Pasajeros	Color	Modelo	Año	Fabricante
Auto1	4	4	3	Rojo	Ford Focus	2010	Ford
Auto2	4	4	3	Negro	Ford Focus	2010	Ford
Auto3	4	2	1	Blanco	Ford Start Concept	2012	Ford
Auto4	4	2	1	Azul	Ford Start Concept	2012	Ford

b. Modelo de datos.

El modelo de datos es una herramienta que permite abstraer ciertas realidades o mini-mundos para obtener como resultado estructuras de base de datos y restricciones que el mini-mundo implica. Permite manipular los datos contenidos en las estructuras de base de datos. Al modelo de datos lo definen como: “Un conjunto de conceptos, reglas y convenciones bien definidos que nos permiten aplicar una serie de abstracciones a fin de describir y manipular los datos de un cierto mundo real que deseamos almacenar en la base de datos” (De Miguel, Piattini y Castaño, 2000, p.23).

Para definir de forma precisa las propiedades del modelo de datos, podemos distinguirlas en dos grupos: (a) estáticas y (b) dinámicas.

a. Estáticas: son las propiedades que permanecen invariables en el tiempo, en éste grupo se encuentran las estructuras.

b. Dinámicas: son los datos que se encuentran almacenadas dentro de las estructuras de la base de datos que van variando en el transcurso del tiempo gracias a las operaciones que se les aplican.

Para definir las estructuras y las operaciones en las bases de datos existen dos lenguajes: (a) lenguaje de definición de datos (LDD) y (b) el lenguaje de manipulación de datos (LMD). Las cuales respectivamente sirven para el mantenimiento de las estructuras de las base de datos y de los datos que se almacenan en ellas.

El lenguaje de definición de datos (LDD).- Permite definir las estructuras de las relaciones (tablas) y las restricciones que existen entre ellas, el siguiente ejemplo lo muestra:

```
Tabla colección
CREATE TABLE[Col]
(
                    [ColCod]CHAR(3) NOTNULL,
                    [ColDes]CHAR(35)NOTNULL,
                    PRIMARYKEY([ColCod])
)

Tabla cliente
CREATETABLE[Cli]
(
                    [CliCod]CHAR(6) NOTNULL,
                    [CliRazCom]CHAR(35) NOTNULL,
                    PRIMARYKEY([CliCod])
)

Tabla unidad de medida
CREATETABLE[UniMed]
(
                        [UniMedCod]CHAR(2)NOTNULL,
                        [UniMedDes]CHAR(35)NOTNULL,
                        PRIMARYKEY([UniMedCod])
)

Tabla producto
CREATETABLE[Prd]
(
                        [MnvCod]CHAR(2)      NOTNULL,
                        [PrdCod]CHAR(28)NOTNULL,
```

```
[PrdDes]CHAR(35)NOTNULL,
[ColCod]CHAR(3)NOTNULL,
[UniMedCod]CHAR(2)NOTNULL,
[CliCod]CHAR(6)NOTNULL,
PRIMARYKEY([MnvCod],[PrdCod])
)
```

Índices

Índice de producto a cliente.

```
CREATENONCLUSTEREDINDEX[IPRD1]
ON[Prd]([CliCod])
```

Índice de producto a unidad de medida.

```
CREATENONCLUSTEREDINDEX[IPRD2]
ON[Prd]([UniMedCod])
```

Índice de producto a colección.

```
CREATENONCLUSTEREDINDEX[IPRD3]
ON[Prd]([ColCod])
```

Restricciones

Restricción en la tabla producto para el campo colcod.

```
ALTERTABLE[Prd]
ADDCONSTRAINT[IPRD3]  FOREIGNKEY([ColCod])
REFERENCES[Col]([ColCod])
```

Restricción en la tabla producto para el campo unimecod.

```
ALTERTABLE[Prd]
ADDCONSTRAINT[IPRD2]
FOREIGNKEY([UniMedCod])
REFERENCES[UniMed]([UniMedCod])
```

Restricción en la tabla producto para el campo clicod.

```
ALTERTABLE[Prd]
ADDCONSTRAINT[IPRD1]
FOREIGNKEY([CliCod])
REFERENCES[Cli]([CliCod])
```

El ejemplo anterior muestra la definición de las estructuras y restricciones en las relaciones (tablas) en una base de datos de prendas que maneja productos, clientes, unidad de medida y colección de las mismas, en el cual cada producto tiene asociado un cliente, una unidad de medida (tallas) y la colección (temporada invierno, otoño, primavera o verano) a la que corresponde.

El lenguaje de manipulación de datos (LMD).- Permite realizar operaciones a los datos almacenados en las estructuras. Las siguientes sentencias lo muestran:

```
SELECT[ColCod] AS ColCod, [ColDes] AS ColDes FROM[Col]
SELECT[CliRazCom] AS CliRazCom, [CliCod] AS CliCod FROM[Cli]
SELECT[UniMedDes] AS UniMedDes, [UniMedCod] AS UniMedCod FROM[UniMed]
SELECT T1.[MnvCod] AS Macronivel, T1.[PrdCod] AS PrdCod, T1.[PrdDes] AS Descripcion, T1.[CliCod]
AS CliCod, T1.[UniMedCod] AS UniMedCod, T1.[ColCod] AS ColCod, T2.[ColDes] AS ColDes,
T3.[UniMedDes] AS UniMedDes, T4.[CliRazCom] AS CliRazCom,
FROM[Prd] T1
INNER JOIN[Col] T2 ON T2.[ColCod] = T1.[ColCod]
INNER JOIN[UniMed] T3 ON T3.[UniMedCod] = T1.[UniMedCod]
INNER JOIN[Cli] T4 ON T4.[CliCod] = T1.[CliCod]
```

Los anteriores ejemplos muestran el modo de manipular los datos mostrando las colecciones, los clientes, las unidades de medida y los productos que existen en la base de datos.

El modelo de datos proporciona facilidades para definir ambos aspectos, propiedades estáticas (LDD) y dinámicas (LMD).

c. Modelo Relacional.

El modelo relacional considera a la base de datos como un conjunto de relaciones (tablas), donde cada relación o tabla está compuesta por filas y columnas. Cada fila dentro de una relación que contiene datos se le denomina registro o tupla, por consiguiente el modelo relacional se fundamenta en que cada relación es un conjunto de datos.

Conceptos básicos

Tabla.- El modelo relacional representa a una relación como una tabla bidimensional. Considerando el ejemplo anterior una relación o tabla en el modelo relacional sería de la siguiente forma:

Colección	
ColCod	ColDes
001	Invierno
002	Otoño
003	Primavera
004	Verano

Atributos.- Son las columnas de la relación o tabla:

- ColCod: código de la colección.
- ColDes: descripción de la colección.

Esquemas.- Los esquemas en las bases de datos van representados por el nombre de las relaciones y los atributos que éstas tienen. En el diseño de una base de datos puede ir una o más esquemas, al conjunto total de ellas se le denominan esquema relacional de la base de datos.

Colección (ColCod, ColDes)

Tupla.- Cada fila dentro de una relación que tiene valores para cada columna (atributo), se le denomina tupla.

(‘001’, Invierno)

(‘002’, Otoño)

(‘003’, Primavera)

(‘004’, Verano)

Dominios.- Cada columna o atributo debe ser considerado como atómico, vale decir que cada una de ellas no debe subdividirse en una estructura más pequeña, siendo ellas los elementos más pequeños en una base de datos relacional.

d. Lenguaje de consulta estructurado.

El lenguaje de consulta estructurado (SQL por sus siglas en inglés Structured Query Language) es un lenguaje de alto nivel estandarizado que sirve para obtener datos desde una base de datos, definiendo que se quiere obtener y donde obtenerlo, más no el cómo obtenerlo, por eso se dice que SQL es un lenguaje declarativo.

El lenguaje de consulta estructurado es un lenguaje para gestionar las bases de datos relacionales, sirve para la creación y mantenimiento de las mismas, para ello debemos diferenciar dos tipos de lenguajes que sirven para objetivos distintos: (a) el lenguaje de definición de datos (LDD) y (b) el lenguaje de manipulación de datos (LMD). El primero sirve para la creación y mantenimiento de la base de datos gestionando así las relaciones, atributos e índices, el segundo sirve para gestionar los datos: consultar, insertar, actualizar, eliminar, agrupar y ordenar.

La sintaxis del lenguaje de consulta estructurado está formado por varios elementos como son: comandos, cláusulas, operadores y funciones de agregado. Por ello el LDD y el LMD cuentan con los siguientes comandos, cláusulas, operadores y funciones de agregado:

Comando	Descripción
CREATE	Utilizado para crear nuevas tablas, campos e índices.
DROP	Empleado para eliminar tablas e índices.
ALTER	Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.

Comando	Descripción
SELECT	Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado.
INSERT	Utilizado para cargar lotes de datos en la base de datos en una única operación.
UPDATE	Utilizado para modificar los valores de los campos y registros especificados.
DELETE	Utilizado para eliminar registros de una tabla de una base de datos.

Cláusula	Descripción
FROM	Utilizada para especificar la tabla de la cual se van a seleccionar los registros.
WHERE	Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar.
GROUP BY	Utilizada para separar los registros seleccionados en grupos específicos.
HAVING	Utilizada para expresar la condición que debe satisfacer cada grupo.
ORDER BY	Utilizada para ordenar los registros seleccionados de acuerdo con un orden específico.

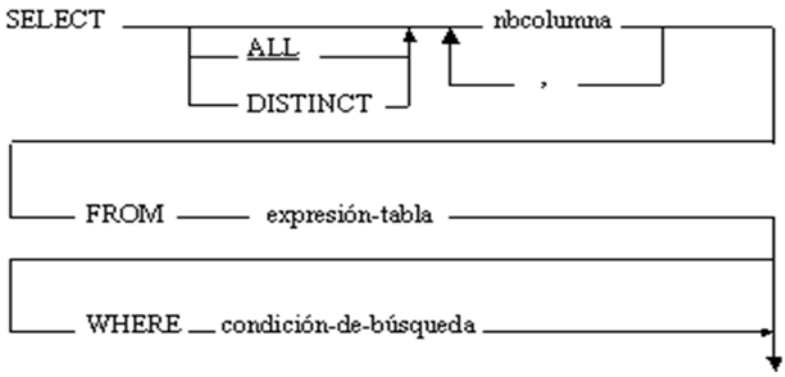
Operador	Uso
AND	Es el “y” lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Es el “o” lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT	Negación lógica. Devuelve el valor contrario de la expresión.

Operador	Uso
<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor ó Igual que
>=	Mayor ó Igual que
=	Igual que
BETWEEN	Utilizado para especificar un intervalo de valores.
LIKE	Utilizado en la comparación de un modelo.
IN	Especificado para especificar registros de una base de datos.

Función	Descripción
AVG	Utilizado para calcular el promedio de los valores de un campo determinado.
COUNT	Utilizada para devolver el número de registros de la selección.
SUM	Utilizada para devolver la suma de todos los valores de un campo determinado
MAX	Utilizada para devolver el valor más alto de un campo especificado.
MIN	Utilizada para devolver el valor más bajo de un campo especificado.

1. Formación de consultas

Para la formación de consultas debe empezarse siempre con cualquier de los comandos LMD y siguiéndole alguna cláusula, la siguiente figura muestra el diagrama sintáctico para la formación de una consulta:



Como se puede ver en la figura para formar una consulta se debe empezar por la palabra reservada SELECT, luego poner ALL o DICTINCT (o nada), seguido de los atributos separadas por comas, estableciendo luego de donde se obtendrán los datos con la palabra reservada FROM seguido del nombre de la relación o tabla, por último opcionalmente pueden establecerse las condiciones (las cuales deben hacer uso de los operadores de comparación) de la consulta con la cláusula WHERE, separadas por los operadores AND u OR.

Ejemplo:

Teniendo la siguiente relación:

Alumnos

Código	Nombre	Apellido
2006241401	Fernando	Linares
2010341502	Patricia	Gonzales
2009345602	Ángela	Retamozo

SELECT codigo, nombre, apellido FROM alumnos WHERE codigo = ‘2006241401’

El resultado será el siguiente:

Codigo	Nombre	Apellido
2006241401	Fernando	Linares

SELECT código, nombre, apellido FROM alumnos WHERE apellido = ‘Gonzales’ AND apellido = ‘Retamozo’

Codigo	Nombre	Apellido
2010341502	Patricia	Gonzales
2009345602	Ángela	Retamozo

Para el segundo ejemplo también se podría escribir de la siguiente forma:

SELECT código, nombre, apellido FROM alumnos WHERE apellido IN (‘Gonzales’, ‘Retamozo’)

Fíjese que se cambia el operador AND por el operador IN. El resultado es el mismo:

Codigo	Nombre	Apellido
2010341502	Patricia	Gonzales
2009345602	Ángela	Retamozo

ii. Combinación Interna

Es el producto cartesiano de dos relaciones, es la combinación de cada registro de la primera relación con cada registro de la segunda, indicando las condiciones que el producto cartesiano debe cumplir para que se dé un resultado. Por ejemplo:

Apellido	IDDepartamento
Gallegos	31
Zúñiga	33
Ramirez	33
Sheen	34
Zolano	34
Landa	36

NombreDepartamento	IDDepartamento
Ventas	31
Sistemas	33
Diseño	34
Producción	35

Existen dos formas de combinar las relaciones empleados y departamentos:

Unión interna implícita.

```
SELECT   apellido, nombredepartamento
FROM     empleados, departamentos
WHERE    empleados.iddepartamento =  departamentos.iddepartamento
```

Unión interna explícita

```
SELECT   apellido, nombredepartamento
FROM     empleados
JOIN     departamentos
ON       empleados.iddepartamento = departamentos.iddepartamento
```

Fíjese que la primera consulta establece las relaciones separada por comas y utiliza la cláusula where estableciendo la condición con el operador = en atributos que se encuentran en las distintas relaciones pero que tienen el mismo significado. Mientras tanto la segunda consulta usa la sentencia join indicando la primera relación a su izquierda y la segunda a su derecha, estableciendo también que la condición debe darse por un campo que tenga el mismo significado en ambas relaciones.

3. Metodología

3.1. Tipo y nivel de investigación.

a. Tipo de investigación.

La presente investigación es aplicada porque se emplearán conceptos como: programación orientada a objetos, modelo de datos, modelo relacional y el lenguaje de consulta estructurado, además se utilizarán como referencia los ejemplos ofrecidos en GeneXus Extensions SDK para obtener el producto que se desea.

b. Nivel de investigación

Exploratorio y Experimental, la actual investigación nace a partir de la necesidad de realizar consultas dinámicas en aplicaciones generadas por la herramienta GeneXus Ev2, como una solución alterna a la dada por GxQuery se decide implementar la extensión para realizar dicha tarea. Encontrando el inconveniente que no hay una documentación formal o es muy poco estudiado el tema de desarrollo de extensiones en GeneXus Ev2, el cual hace que ésta solución sea una de las muchas que se pudieran dar en el futuro.

3.2. Variables

Independientes

- Arquitectura abierta para la extensibilidad de GeneXus Ev2.
- Librerías de GeneXus para desarrollar extensiones (Plataforma GeneXus Extensions Software Development Kit).

Dependientes

- Extensión para la realización de consultas dinámicas.
- Guía de implementación de extensiones.

Indicadores

- Fácil instalación de la extensión.
- Fácil implementación de consultas dinámicas con la extensión.
- Fácil comprensión de la guía de implementación.
- Ejemplos prácticos en la guía de implementación.

Referencias bibliográficas

- CommunityWiki (2012). *Genexus X/GeneXus Platform SDK, GeneXus Paltform Software Development Kit*. Recuperado de: http://wiki.gxtechnical.com/commwiki/servlet/hwiki?GeneXus+X%2FGeneXus+Platform+SDK,

- Genexus (2012). *GXquery es una herramienta que se integra con GeneXus y que permite crear reportes desde bases de datos operacionales*. Recuperado de: http://www.genexus.com/productos/gxquery?es

- Genexus (2012). *Liberación de GXplorer y GXquery*. Recuperado de: http://www.genexus.com/noticias/leer-noticia/liberacion-de-gxplorer-y-gxquery?es

- GxData (2012). *GXDATA Tecnologías de la Información EIRL*. Recuperado de: http://www.gxdata.com.pe/

- GXtecnical (2007). *GXplorer y GXquery Io RC*. Recuperado de: http://www2.gxtechnical.com/portal/hgxpp001.aspx?15,7,3,O,S,0,PAG;CONC;131;3;D;9369;1;PAG;,

- GXtechnical (2012). *GeneXus Query – Guia rápida de uso*. Recuperado de: http://ftpusa.artech.com.uy/files/1353467878/gxq40startsp.pdf

Anexo B: Código fuente

1. Descripción de clases.

Se describen las siguientes clases:

Nombre	Descripción
Ckb	Clase utilitaria para obtener los valores del proyecto actual.
CLauncher	Clase encargada de ejecutar la consulta generada.
CMessageOutput	Clase utilitaria para mostrar mensajes en la ventana output de GeneXus Ev2.
CommandKeys	Clase que contiene los comandos para iniciar el objeto extensión en la IDE de GeneXus Ev2.
CommandManager	Clase para iniciar el objeto extensión en la IDE de GeneXus Ev2.
CQuery	Clase para gestionar la generación de la consulta.
CXml	Clase utilitaria para generar archivos .xml.
iQueries	Clase principal que tiene toda la lógica de la extensión.
PropertiesHelper	Clase utilitaria que obtiene las propiedades del servidor.

2. Código fuente de clases.

a. Clase CKb.cs

```
using Artech.Genexus.Common.Entities;
using Artech.Architecture.UI.Framework.Services;
using Artech.Genexus.Common;
using Artech.Architecture.Common.Objects;
using System;
using System.Collections.Generic;

namespace IvanGQ.iSQL
{
public class CKb
{
    private static GxDataStore gxDataStore;

    public static GxDataStore GetDefaultDataStore()
    {
        IEnumerator<GxDataStore> enumerator =
        UIServices.KB.CurrentKB.WorkingModel.GetAs<GxModel>()
        .DataStores.GetEnumerator();
        gxDataStore = null;
        try
        {
            using (enumerator)
            {
```

```
                        while (enumerator.MoveNext())
                        {
                                GxDataStore current = enumerator.Current;
                                if (!current.DataStoreCategory.IsDefault)
                                {
                                        continue;
                                }
                                gxDataStore = current;
                                return gxDataStore;
                        }
                        return gxDataStore;
                }
        }
        catch (Exception ex)
        {
                CMessageOutput.MessageError(ex.Message);
        }
        return gxDataStore;
}
public static KBModel GetCurrentModel() {
        return UIServices.KB.CurrentModel;
}
public static String GetKBLocation() {
        return UIServices.KB.CurrentKB.Location;
}
}
}
```

b. CLauncher.cs

```
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System;
using Artech.Genexus.Common.Entities;
using Artech.Architecture.UI.Framework.Services;
using Artech.Genexus.Common;
using Artech.Architecture.Common.Services;

namespace IvanGQ.iSQL
{
class CLauncher
{
        private IOutputService output;
        private DataSet dataSet;
        private DataTable dataTable;
        private SqlDataAdapter dataAdapter;

        public CLauncher() {
                output = CommonServices.Output;
        }
```

```
public SqlConnection ConnectDataStore() {
        SqlConnection sqlConnection = null;
        try {
                String[] properties = CXml.ReadPropertiesKB();
                string server = properties[0];
                string dbname = properties[1];
                string dbmsport = properties[2];
                string user = properties[3];
                string password = properties[4];
                string trusted = properties[5];
                string stringConnection = "";
                if (trusted.ToUpper() != "NO")
                {
                        string[] strArrays = new string[5];
                        strArrays[0] = "Initial Catalog=";
                        strArrays[1] = dbname;
                        strArrays[2] = ";Data Source=";
                        strArrays[3] = server;
                        strArrays[4] = ";Integrated Security=yes";
                        stringConnection = string.Concat(strArrays);
                }
                else
                {
                        string[] strArrays1 = new string[8];
                        strArrays1[0] = "User ID=";
                        strArrays1[1] = user;
                        strArrays1[2] = ";Password=";
                        strArrays1[3] = password;
                        strArrays1[4] = ";Initial Catalog=";
                        strArrays1[5] = dbname;
                        strArrays1[6] = ";Data Source=";
                        strArrays1[7] = server;
                        stringConnection = string.Concat(strArrays1);
                }

                if (!(dbname == "") || !(server == ""))
                {
                        try
                        {
                                sqlConnection = new
                                SqlConnection(stringConnection);
                        }
                        catch (SqlException sqlException)
                        {
                                output.AddErrorLine(sqlException.Message);
                        }
                }
        }catch(Exception ex){
                CMessageOutput.MessageError("Aqui 2" + ex.Message);
        }

        return sqlConnection;
}

public DataTable ExecuteQuery(string query) {
        SqlConnection sqlConnection = this.ConnectDataStore();
```

```
                try {
                        sqlConnection.Open();
                }catch(Exception ex){
                        CMessageOutput.MessageFailed("No se pudo conectar
                        al servidor: " + ex.Message);
                }
                if (sqlConnection.State == ConnectionState.Open)
                {
                        this.dataSet = new DataSet();
                        this.dataTable = new DataTable();
                        this.dataAdapter = new SqlDataAdapter(query,
                        sqlConnection);
                        this.dataAdapter.Fill(this.dataTable);
                        int columns = this.dataTable.Columns.Count;
                        DataTable tableAux = new DataTable();
                        for (int index = 0; index < columns; index++)
                        {
                                DataColumn columnAux = this.dataTable.Columns[index];
                                DataColumn column =
                                new DataColumn(columnAux.ColumnName.Trim(),
                                columnAux.DataType);
                                tableAux.Columns.Add(column);
                        }

                        foreach (DataRow row in this.dataTable.Rows)
                        {
                                object[] newRow = new object[columns];
                                for (int index = 0; index < columns; index++)
                                {
                                        newRow[index] =
                                        row.ItemArray[index].ToString().Trim();
                                }
                                try {
                                        tableAux.Rows.Add(newRow);
                                }catch(ArgumentException ex){
                                        CMessageOutput.MessageFailed("No se puede
                                        obtener imágenes:" + ex.Message);
                                }
                        }

                        this.dataTable.Clear();
                        this.dataTable = tableAux.Copy();
                        new SqlCommandBuilder(this.dataAdapter);
                }
                else
                {
                        output.AddErrorLine("No se pudo obtener
                        la conexión.");
                }

                return this.dataTable;
        }
}
}
```

c. CMessageOutput.cs

```
using Artech.Architecture.Common.Services;

namespace IvanGQ.iSQL
{
public class CMessageOutput
{
        private static IOutputService output;

        public static void MessageStart(string message) {
                output = CommonServices.Output;
                output.StartSection(message, true);
        }

        public static void MessageSuccess(string message) {
                output = CommonServices.Output;
                output.EndSection(message, true);
        }

        public static void MessageFailed(string message)
        {
                output = CommonServices.Output;
                output.EndSection(message, false);
        }

        public static void MessageError(string message) {
                output = CommonServices.Output;
                output.AddErrorLine(message);
        }

        public static void MessageLine(string message) {
                output = CommonServices.Output;
                output.AddLine(message);
        }
}
}
```

d. CommandKeys.cs

```
using Artech.Common.Framework.Commands;

namespace IvanGQ.iSQL
{
public class CommandKeys
{
        private static CommandKey showiQueries = new
        CommandKey(iQueries.guid, "iQueries");

        public static CommandKey ShowiQueries
        {
                get { return CommandKeys.showiQueries; }
        }
}
}
```

e. CommandManager.cs

```
using Artech.Common.Framework.Commands;
using Artech.Architecture.UI.Framework.Helper;
using Artech.Architecture.UI.Framework.Services;
using System.Windows.Forms;

namespace IvanGQ.iSQL
{
internal class CommandManager: CommandDelegator
{
        public CommandManager(){
                this.AddCommand(CommandKeys.ShowiQueries,
                new ExecHandler(ExecShowiQueries),
                new QueryHandler(ShowiQueries));
        }

        public bool ExecShowiQueries(CommandData commandData) {
                CMessageOutput.MessageStart("iQueries Started");
                UIServices.ToolWindows.SelectToolWindow(iQueries.guid);
                return true;
        }

        public bool ShowiQueries(CommandData commandData,
        ref CommandStatus status)
        {
                CMessageOutput.MessageStart("iQueries Started");
                status.Enable(true);
                return true;
        }
}
}
```

f. CQuery.cs

```
using Artech.GXplorer.Common.Objects;
using Artech.Architecture.Common.Objects;
using Artech.Architecture.UI.Framework.Services;
using Artech.GXplorer.Common.Parts;
using Artech.Genexus.Common.Objects;
using System.Collections;

namespace IvanGQ.iSQL
{
class CQuery
{
        private static Hashtable elements = new
        Hashtable();
        private static Hashtable elementsXml = new
        Hashtable();
        private static QueryObject queryObject = new
        QueryObject(UIServices.KB.CurrentModel);
        private static QueryObject queryObjectXml = new
        QueryObject(UIServices.KB.CurrentModel);
        public static void AddQueryElement(QueryElement queryElement) {
                if (!elements.ContainsKey(queryElement.Name))
```

```
		{
			elements.Add(queryElement.Name, queryElement);
		}
		else {
			elements.Remove(queryElement.Name);
			elements.Add(queryElement.Name, queryElement);
		}
	}
	public static void AddQueryElementXml(QueryElement queryElement) {
		if (!elementsXml.ContainsKey(queryElement.Name))
		{
			elementsXml.Add(queryElement.Name, queryElement);
		}
		else
		{
			elementsXml.Remove(queryElement.Name);
			elementsXml.Add(queryElement.Name, queryElement);
		}
	}
	public static string GetQuery(Hashtable hashElements) {
		Hashtable aux = new Hashtable();
		foreach (DictionaryEntry entry in elements)
		{
			if(hashElements.ContainsKey(entry.Key)){
				aux.Add(entry.Key, entry.Value);
			}
		}
		elements = aux;
		queryObject = new QueryObject(UIServices.KB.CurrentModel);
		//queryObject.QueryStructurePart.Elements.Clear();
		foreach(DictionaryEntry entry in elements){
			QueryElement element = (QueryElement)entry.Value;
			queryObject.QueryStructurePart.Elements.Add(element);
		}
		string sql = queryObject.QuerySQLSentencePart.GetSQLSentence;
		return sql;
	}
	public static string GetQueryXml(Hashtable hashElements) {
		Hashtable aux = new Hashtable();
		foreach (DictionaryEntry entry in elementsXml)
		{
			if (hashElements.ContainsKey(entry.Key))
			{
				aux.Add(entry.Key, entry.Value);
			}
		}
		elementsXml = aux;
		queryObjectXml = new QueryObject(UIServices.KB.CurrentModel);
		foreach (DictionaryEntry entry in elementsXml)
		{
			QueryElement element = (QueryElement)entry.Value;
			queryObjectXml.QueryStructurePart.Elements.Add(element);
		}
		string sql = queryObjectXml.QuerySQLSentencePart.GetSQLSentence;
		return sql;
	}
```

```
        public static QueryObject QueryObject {
            get {
                return queryObject;
            }
        }
        public static void ClearQueryObject() {
            queryObject.QueryStructurePart.Elements.Clear();
        }
    }
    }
```

g. CXml.cs

```
using Artech.Genexus.Common.Entities;
using Artech.Architecture.UI.Framework.Services;
using Artech.Genexus.Common.Objects;
using Artech.Genexus.Common.Parts;
using System.ComponentModel;
using System.Xml;
using System;
using System.IO;
using System.Windows.Forms;
using System.Collections;
using System.Collections.Generic;

namespace IvanGQ.iSQL
{
public class CXml
{
        public static void SavePropertiesKB(GxDataStore gxDataStore)
        {
            PropertiesHelper propertiesHelper = new
            PropertiesHelper(gxDataStore);
            string server = propertiesHelper.GetProperty<string>
            ("CS_SERVER");
            string dbname = propertiesHelper.GetProperty<string>
            ("CS_DBNAME");
            string dbms_port = propertiesHelper.GetProperty<string>
            ("DBMS_PORT");
            string user = propertiesHelper.GetProperty<string>
            ("USER_ID");
            string password = propertiesHelper.GetProperty<string>
            ("USER_PASSWORD");
            string trusted = propertiesHelper.GetProperty<string>
            ("TRUSTED_CONNECTION");

            string currentPath = UIServices.KB.CurrentKB.Location;
            string newPath = System.IO.Path.Combine(currentPath,
            "iQueries");
            //System.IO.Directory.CreateDirectory(newPath);
            newPath = System.IO.Path.Combine(newPath,
            "iQueriesConfig.xml");

            if (!System.IO.File.Exists(newPath))
            {
                XmlTextWriter xmlTextWriter = new XmlTextWriter(newPath,
```

```
                System.Text.Encoding.UTF8);
                xmlTextWriter.WriteStartDocument(true);
                xmlTextWriter.Formatting = Formatting.Indented;
                xmlTextWriter.Indentation = 2;
                xmlTextWriter.WriteStartElement("iQueriesConfig");
                createNode(xmlTextWriter, server, dbname, dbms_port, user, password,
                trusted);
                xmlTextWriter.WriteEndElement();
                xmlTextWriter.WriteEndDocument();
                xmlTextWriter.Close();
                CMessageOutput.MessageSuccess("El archivo de configuración
                iQueriesConfig.xml se ah creado exitósamente...");
        }
}
private static void createNode(XmlTextWriter xmlTextWriter,
string server,string dbname,string dbms_port,string user,
string password,string trusted) {
        xmlTextWriter.WriteStartElement("Config");

        xmlTextWriter.WriteStartElement("server");
        xmlTextWriter.WriteString(server);
        xmlTextWriter.WriteEndElement();

        xmlTextWriter.WriteStartElement("dbname");
        xmlTextWriter.WriteString(dbname);
        xmlTextWriter.WriteEndElement();

        xmlTextWriter.WriteStartElement("dbmsport");
        xmlTextWriter.WriteString(dbms_port);
        xmlTextWriter.WriteEndElement();

        xmlTextWriter.WriteStartElement("user");
        xmlTextWriter.WriteString(user);
        xmlTextWriter.WriteEndElement();

        xmlTextWriter.WriteStartElement("password");
        xmlTextWriter.WriteString(password);
        xmlTextWriter.WriteEndElement();

        xmlTextWriter.WriteStartElement("trusted");
        xmlTextWriter.WriteString(trusted);
        xmlTextWriter.WriteEndElement();

        xmlTextWriter.WriteEndElement();
}
public static String[] ReadPropertiesKB() {
        XmlDataDocument xmlDataDocument = new XmlDataDocument();
        XmlNodeList xmlNodeList;
        String[] properties = new String[6];
        string currentPath = UIServices.KB.CurrentKB.Location;
        string newPath = System.IO.Path.Combine(currentPath,
        "iQueries");
        newPath = System.IO.Path.Combine(newPath, "iQueriesConfig.xml");
        CMessageOutput.MessageLine(newPath);
        FileStream fs = new FileStream(newPath, FileMode.Open,
        FileAccess.Read);
```

```
            xmlDataDocument.Load(fs);
            xmlNodeList = xmlDataDocument.GetElementsByTagName("Config");
            try {
                  properties[0] = xmlNodeList[0].ChildNodes.Item(0)
                  .InnerText.Trim();
                  properties[1] = xmlNodeList[0].ChildNodes.Item(1)
                  .InnerText.Trim();
                  properties[2] = xmlNodeList[0].ChildNodes.Item(2)
                  .InnerText.Trim();
                  properties[3] = xmlNodeList[0].ChildNodes.Item(3)
                  .InnerText.Trim();
                  properties[4] = xmlNodeList[0].ChildNodes.Item(4)
                  .InnerText.Trim();
                  properties[5] = xmlNodeList[0].ChildNodes.Item(5)
                  .InnerText.Trim();
            }catch(Exception ex){
                  CMessageOutput.MessageError("Aqui " + ex.Message);
            }

            fs.Close();
            return properties;
      }
      public static bool ExistPropertiesKB() {
            string currentPath = UIServices.KB.CurrentKB.Location;
            string newPath = System.IO.Path.Combine(currentPath, "iQueries");
            newPath = System.IO.Path.Combine(newPath, "iQueriesConfig.xml");
            if (!System.IO.File.Exists(newPath))
            {
                  return false;
            }
            else
            {
                  return true;
            }
      }
      public static bool SaveQuery(Hashtable hashParents,
      Hashtable hashChilds, string nameQuery, string query) {
            string currentPath = UIServices.KB.CurrentKB.Location;
            string newPath = System.IO.Path.Combine(currentPath, "iQueries");
            string newPath1 = System.IO.Path.Combine(newPath, "admin");
            if (!System.IO.Directory.Exists(newPath1)) {
                  System.IO.Directory.CreateDirectory(newPath1);
            }
            bool success = false;
            newPath1 = System.IO.Path.Combine(newPath1, nameQuery + ".xml");

            if (!System.IO.File.Exists(newPath1))
            {
                  XmlTextWriter xmlTextWriter = new XmlTextWriter(newPath1,
                  System.Text.Encoding.UTF8);
                  xmlTextWriter.WriteStartDocument(true);
                  xmlTextWriter.Formatting = Formatting.Indented;
                  xmlTextWriter.Indentation = 2;
                  xmlTextWriter.WriteStartElement("iQueries");
                  xmlTextWriter.WriteAttributeString("xmlns", null, null,
                  UIServices.KB.CurrentKB.Name);
```

```
                    List<string> strs = new List<string>();
                    IEnumerator<Table> tables = Table.GetAll(UIServices.KB.CurrentModel)
                    .GetEnumerator();
                    while (tables.MoveNext())
                    {
                        Table table = (Table)tables.Current;
                        if (hashParents.ContainsKey(table.Name))
                        {
                            xmlTextWriter.WriteStartElement(table.Name);
                            xmlTextWriter.WriteAttributeString("xmlns", null, null,
                            UIServices.KB.CurrentKB.Name);

                            IEnumerator<TableAttribute> atts =
                            table.TableStructure.Attributes.GetEnumerator();
                            string nameParent = table.Name;
                            while (atts.MoveNext())
                            {
                                Artech.Genexus.Common.Objects.Attribute att
                                = (Artech.Genexus.Common.Objects.
                                Attribute)atts.Current;
                                if (hashChilds.ContainsKey(nameParent +
                                att.Name))
                                {
                                xmlTextWriter.WriteStartElement(att.Name);
                                    xmlTextWriter.WriteString(att.Name);
                                    xmlTextWriter.WriteEndElement();
                                }
                            }
                            xmlTextWriter.WriteEndElement();
                        }
                    }

                    xmlTextWriter.WriteEndElement();
                    xmlTextWriter.WriteEndDocument();
                    xmlTextWriter.Namespaces = true;
                    xmlTextWriter.Close();
                    success = true;
                }
                else
                {
                    CMessageOutput.MessageFailed("El nombre de archivo " +
                    nameQuery + " ya existe.");
                }
                return success;
            }
        }
        }
```

h. iQueries.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
```

```
using System.IO;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
using System.Text;
using System.Windows.Forms;
using System.Xml;
using Artech.Architecture.Common.Services;
using Artech.Architecture.Common.Collections;
using Artech.Architecture.Common.Objects;
using Artech.Architecture.Common.Threading;
using Artech.Architecture.UI.Framework.Packages;
using Artech.Architecture.UI.Framework.Services;
using Artech.Architecture.UI.Framework;
using Artech.Common.Collections;
using Artech.Common.Framework.Selection;
using Artech.Common.Helpers.Threading;
using Artech.FrameworkDE;
using Artech.Genexus.Common.Entities;
using Artech.Genexus.Common.ModelParts;
using Artech.Genexus.Common.Objects;
using Artech.GXplorer.Common.Objects;
using Artech.ReverseEngineering.Data.Runtime;
using Artech.ReverseEngineering.Data.Database;
using Artech.Genexus.Common;
using Artech.Genexus.Common.Parts;
using Artech.ReverseEngineering.Data;
using Artech.GXplorer.Common.DataAccess;
using Artech.GXplorer.Common;
using Artech.GXplorer.Common.Parts;
using Artech.Genexus.Common.Parts.SDT;

namespace IvanGQ.iSQL
{
[Guid("CFC19729-6C1E-4474-A5DE-D76144BA7C3E")]

public class iQueries: AbstractToolWindow,
ISelectionListener
{
        public static Guid guid;

        private GxDataStore gxDataStore;

        private CLauncher launcher;

        private ToolStripButton btnRunSql;
        private ToolStrip tsContenedor;
        private ToolStripButton btnSaveSql;
        private ToolStripButton btnSql;
        private ToolStripButton btnXml;
        private ToolStripButton btnClear;

        private TextBox txtSql;
        private TextBox txtNameQuery;

        private ImageList imageList;
        private IContainer components;
```

```
private TreeView treeView;
private GroupBox gbSavedQueries;
private DataGridView dgvResult;
private Panel panelMain;
private TreeView treeViewQueries;
private Button btnLock;
private Button btnDeleteXml;
private bool unprotected;

private Label lblTables;
private Label lblQuery;
private Label lblResult;
private Label lblNameQuery;
private Label lblTitle;
private Label lblDynamic;
private Label lblMode;
private Label lblMode2;

private Hashtable hashAttributes;
private Hashtable hashParents;
private Hashtable hashChilds;
private Hashtable hashElements;
private Hashtable hashElementsXml;


public iQueries()
{
        this.InitializeComponent();
        UIServices.TrackSelection.Subscribe(
        Guid.NewGuid(), this);
}
static iQueries()
{
        iQueries.guid = typeof(iQueries).GUID;
}

public void InitializeComponent()
{
        this.components = new System.ComponentModel.
        Container();
        System.ComponentModel.ComponentResourceManager
        resources = new System.ComponentModel.
        ComponentResourceManager(
        typeof(iQueries));
        System.Windows.Forms.DataGridViewCellStyle
        dataGridViewCellStyle1 =
        new System.Windows.Forms.
        DataGridViewCellStyle();
        this.btnRunSql = new
        System.Windows.Forms.ToolStripButton();
        this.btnXml = new
        System.Windows.Forms.ToolStripButton();
        this.txtSql = new
        System.Windows.Forms.TextBox();
        this.tsContenedor = new
        System.Windows.Forms.ToolStrip();
```

```
this.btnSql = new
System.Windows.Forms.ToolStripButton();
this.btnSaveSql = new
System.Windows.Forms.ToolStripButton();
this.btnClear = new
System.Windows.Forms.ToolStripButton();
this.imageList = new
System.Windows.Forms.ImageList(
this.components);
this.treeView = new
System.Windows.Forms.TreeView();
this.lblTables = new
System.Windows.Forms.Label();
this.lblQuery = new
System.Windows.Forms.Label();
this.lblResult = new
System.Windows.Forms.Label();
this.lblNameQuery = new System.Windows.Forms.Label();
this.txtNameQuery = new
System.Windows.Forms.TextBox();
this.lblTitle = new
System.Windows.Forms.Label();
this.lblDynamic = new
System.Windows.Forms.Label();
this.panelMain = new
System.Windows.Forms.Panel();
this.dgvResult = new
System.Windows.Forms.DataGridView();
this.gbSavedQueries = new
System.Windows.Forms.GroupBox();
this.btnDeleteXml = new
System.Windows.Forms.Button();
this.treeViewQueries =
new System.Windows.Forms.TreeView();
this.lblMode = new
System.Windows.Forms.Label();
this.lblMode2 = new
System.Windows.Forms.Label();
this.btnLock = new
System.Windows.Forms.Button();
this.tsContenedor.SuspendLayout();
this.panelMain.SuspendLayout();
((System.ComponentModel.ISupportInitialize)(
this.dgvResult)).
BeginInit();
this.gbSavedQueries.SuspendLayout();
this.SuspendLayout();
//
// btnRunSql
//
this.btnRunSql.Image = Resources.RunSql2;
this.btnRunSql.ImageTransparentColor =
System.Drawing.Color.Magenta;
this.btnRunSql.Name = "btnRunSql";
this.btnRunSql.Size = new System.Drawing.
Size(23, 22);
```

```
this.btnRunSql.TextImageRelation =
System.Windows.Forms.
TextImageRelation.TextBeforeImage;
this.btnRunSql.ToolTipText = "Run the sql";
this.btnRunSql.Click += new
System.EventHandler(this.btnRunSql_Click);
//
// btnXml
//
this.btnXml.Image = Resources.Folder2;
this.btnXml.ImageTransparentColor =
System.Drawing.Color.Magenta;
this.btnXml.Name = "btnXml";
this.btnXml.Size = new System.Drawing.Size(23, 22);
this.btnXml.TextImageRelation = System.Windows.Forms.
TextImageRelation.TextBeforeImage;
this.btnXml.ToolTipText = "Gest xml files";
this.btnXml.Click += new
System.EventHandler(this.btnXml_Click);
//
// txtSql
//
this.txtSql.AcceptsTab = true;
this.txtSql.AllowDrop = true;
this.txtSql.BackColor = System.Drawing.Color.White;
this.txtSql.Font = new
System.Drawing.Font("Courier New", 9.75F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point,
((byte)(0)));
this.txtSql.Location = new
System.Drawing.Point(198, 29);
this.txtSql.Multiline = true;
this.txtSql.Name = "txtSql";
this.txtSql.ReadOnly = true;
this.txtSql.ScrollBars =
System.Windows.Forms.ScrollBars.Vertical;
this.txtSql.Size = new
System.Drawing.Size(403, 177);
this.txtSql.TabIndex = 1;
this.txtSql.TabStop = false;
//
// tsContenedor
//
this.tsContenedor.Items.AddRange(
new System.Windows.Forms.
ToolStripItem[] {
this.btnSql,
this.btnRunSql,
this.btnSaveSql,
this.btnXml,
this.btnClear});
this.tsContenedor.Location = new
System.Drawing.Point(0, 0);
this.tsContenedor.Name = "tsContenedor";
this.tsContenedor.Size = new
```

```
System.Drawing.Size(1125, 25);
this.tsContenedor.TabIndex = 3;
//
// btnSql
//
this.btnSql.Image = Resources.Sql;
this.btnSql.ImageTransparentColor =
System.Drawing.Color.Magenta;
this.btnSql.Name = "btnSql";
this.btnSql.Size = new
System.Drawing.Size(23, 22);
this.btnSql.TextImageRelation =
System.Windows.Forms.
TextImageRelation.TextBeforeImage;
this.btnSql.ToolTipText = "Constructs the query";
this.btnSql.Click += new
System.EventHandler(this.btnSql_Click);
//
// btnSaveSql
//
this.btnSaveSql.DisplayStyle =
System.Windows.Forms.
ToolStripItemDisplayStyle.Image;
this.btnSaveSql.Image = Resources.SaveSql;
this.btnSaveSql.ImageTransparentColor =
System.Drawing.Color.Magenta;
this.btnSaveSql.Name = "btnSaveSql";
this.btnSaveSql.Size = new
System.Drawing.Size(23, 22);
this.btnSaveSql.ToolTipText = "Save the query";
this.btnSaveSql.Click += new
System.EventHandler(this.btnSaveSql_Click);
//
// btnClear
//
this.btnClear.DisplayStyle =
System.Windows.Forms.
ToolStripItemDisplayStyle.Image;
this.btnClear.Image = Resources.Clear;
this.btnClear.ImageTransparentColor =
System.Drawing.Color.Magenta;
this.btnClear.Name = "btnClear";
this.btnClear.Size = new
System.Drawing.Size(23, 22);
this.btnClear.Text = "toolStripButton1";
this.btnClear.ToolTipText = "Clean all objects";
this.btnClear.Click += new
System.EventHandler(this.btnClear_Click);
//
// imageList
//
this.imageList.ImageStream =
((System.Windows.Forms.ImageListStreamer)
(resources.GetObject("imageList.ImageStream")));
this.imageList.TransparentColor =
System.Drawing.Color.Transparent;
```

```
this.imageList.Images.SetKeyName(0,
"Table.GIF");
this.imageList.Images.SetKeyName(1,
"attribute.ico");
this.imageList.Images.SetKeyName(2,
"iconDone.png");
this.imageList.Images.SetKeyName(3,
"Xml.png");
this.imageList.Images.SetKeyName(4,
"Lock.png");
this.imageList.Images.SetKeyName(5,
"Unlock.png");
this.imageList.Images.SetKeyName(6,
"Delete.png");
//
// treeView
//
this.treeView.CheckBoxes = true;
this.treeView.ImageIndex = 0;
this.treeView.ImageList = this.imageList;
this.treeView.Location =
new System.Drawing.Point(19, 25);
this.treeView.Name = "treeView";
this.treeView.SelectedImageIndex = 2;
this.treeView.Size = new
System.Drawing.Size(162, 409);
this.treeView.TabIndex = 10;
this.treeView.AfterCheck +=
new System.Windows.Forms.
TreeViewEventHandler(this.treeView_AfterCheck);
//
// lblTables
//
this.lblTables.AutoSize = true;
this.lblTables.Font = new
System.Drawing.Font("Verdana", 9.75F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point,
((byte)(0)));
this.lblTables.Location = new
System.Drawing.Point(16, 6);
this.lblTables.Name = "lblTables";
this.lblTables.Size = new
System.Drawing.Size(57, 16);
this.lblTables.TabIndex = 11;
this.lblTables.Text = "Tables:";
//
// lblQuery
//
this.lblQuery.AutoSize = true;
this.lblQuery.Font = new
System.Drawing.Font("Verdana", 9.75F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point,
((byte)(0)));
this.lblQuery.Location = new
```

```
System.Drawing.Point(198, 7);
this.lblQuery.Name = "lblQuery";
this.lblQuery.Size = new
System.Drawing.Size(53, 16);
this.lblQuery.TabIndex = 12;
this.lblQuery.Text = "Query:";
//
// lblResult
//
this.lblResult.AutoSize = true;
this.lblResult.Font = new System.
Drawing.Font("Verdana", 9.75F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point,
((byte)(0)));
this.lblResult.Location = new System.
Drawing.Point(198, 209);
this.lblResult.Name = "lblResult";
this.lblResult.Size = new System.
Drawing.Size(54, 16);
this.lblResult.TabIndex = 13;
this.lblResult.Text = "Result:";
//
// lblNameQuery
//
this.lblNameQuery.AutoSize = true;
this.lblNameQuery.Font = new System.
Drawing.Font("Verdana", 9.75F,
System.Drawing.FontStyle.Italic,
System.Drawing.GraphicsUnit.Point,
((byte)(0)));
this.lblNameQuery.Location = new
System.Drawing.Point(357, 7);
this.lblNameQuery.Name = "lblNameQuery";
this.lblNameQuery.Size = new
System.Drawing.Size(95, 16);
this.lblNameQuery.TabIndex = 14;
this.lblNameQuery.Text = "Name query:";
//
// txtNameQuery
//
this.txtNameQuery.Location = new
System.Drawing.Point(458, 7);
this.txtNameQuery.Name = "txtNameQuery";
this.txtNameQuery.Size = new
System.Drawing.Size(143, 20);
this.txtNameQuery.TabIndex = 15;
//
// lblTitle
//
this.lblTitle.AutoSize = true;
this.lblTitle.Font = new
System.Drawing.Font("Verdana", 12F,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point,
((byte)(0)));
```

```
this.lblTitle.Location = new
System.Drawing.Point(15, 35);
this.lblTitle.Name = "lblTitle";
this.lblTitle.Size = new
System.Drawing.Size(86, 18);
this.lblTitle.TabIndex = 16;
this.lblTitle.Text = "iQueries:";
//
// lblDynamic
//
this.lblDynamic.AutoSize = true;
this.lblDynamic.Font = new
System.Drawing.Font("Verdana", 12F,
System.Drawing.FontStyle.Italic,
System.Drawing.GraphicsUnit.Point,
((byte)(0)));
this.lblDynamic.Location = new
System.Drawing.Point(107, 35);
this.lblDynamic.Name = "lblDynamic";
this.lblDynamic.Size = new
System.Drawing.Size(147, 18);
this.lblDynamic.TabIndex = 17;
this.lblDynamic.Text = "Dynamic Queries";
//
// panelMain
//
this.panelMain.Controls.Add(this.dgvResult);
this.panelMain.Controls.Add(this.gbSavedQueries);
this.panelMain.Controls.Add(this.treeView);
this.panelMain.Controls.Add(this.lblTables);
this.panelMain.Controls.Add(this.txtNameQuery);
this.panelMain.Controls.Add(this.lblResult);
this.panelMain.Controls.Add(this.lblNameQuery);
this.panelMain.Controls.Add(this.txtSql);
this.panelMain.Controls.Add(this.lblQuery);
this.panelMain.Location = new
System.Drawing.Point(3, 85);
this.panelMain.Name = "panelMain";
this.panelMain.Size = new
System.Drawing.Size(839, 447);
this.panelMain.TabIndex = 18;
//
// dgvResult
//
this.dgvResult.AllowUserToAddRows = false;
this.dgvResult.AllowUserToDeleteRows = false;
this.dgvResult.AllowUserToOrderColumns = true;
this.dgvResult.BackgroundColor =
System.Drawing.SystemColors.Control;
this.dgvResult.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
dataGridViewCellStyle1.Alignment =
System.Windows.Forms.
DataGridViewContentAlignment.MiddleLeft;
dataGridViewCellStyle1.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
```

```
dataGridViewCellStyle1.Font =
new System.Drawing.Font("Arial Narrow", 8.25F,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point,
((byte)(0)));
dataGridViewCellStyle1.ForeColor =
System.Drawing.Color.White;
dataGridViewCellStyle1.SelectionBackColor =
System.Drawing.SystemColors.Highlight;
dataGridViewCellStyle1.SelectionForeColor =
System.Drawing.SystemColors.HighlightText;
dataGridViewCellStyle1.WrapMode =
System.Windows.Forms.DataGridViewTriState.True;
this.dgvResult.ColumnHeadersDefaultCellStyle =
dataGridViewCellStyle1;
this.dgvResult.ColumnHeadersHeightSizeMode =
System.Windows.Forms.
DataGridViewColumnHeadersHeightSizeMode.
AutoSize;
this.dgvResult.GridColor =
System.Drawing.SystemColors.Control;
this.dgvResult.Location = new
System.Drawing.Point(201, 228);
this.dgvResult.Name = "dgvResult";
this.dgvResult.ReadOnly = true;
this.dgvResult.RowHeadersVisible = false;
this.dgvResult.Size =
new System.Drawing.Size(400, 206);
this.dgvResult.TabIndex = 18;
//
// gbSavedQueries
//
this.gbSavedQueries.Controls.Add(
this.btnDeleteXml);
this.gbSavedQueries.Controls.Add(
this.treeViewQueries);
this.gbSavedQueries.Font = new
System.Drawing.Font("Verdana", 9.75F,
System.Drawing.FontStyle.Italic,
System.Drawing.GraphicsUnit.Point,
((byte)(0)));
this.gbSavedQueries.Location =
new System.Drawing.Point(619, 7);
this.gbSavedQueries.Name = "gbSavedQueries";
this.gbSavedQueries.Size =
new System.Drawing.Size(205, 427);
this.gbSavedQueries.TabIndex = 17;
this.gbSavedQueries.TabStop = false;
this.gbSavedQueries.Text =
"Saved Queries";
//
// btnDeleteXml
//
this.btnDeleteXml.ImageIndex = 6;
this.btnDeleteXml.ImageList = this.imageList;
this.btnDeleteXml.Location =
```

```
new System.Drawing.Point(110, 0);
this.btnDeleteXml.Name = "btnDeleteXml";
this.btnDeleteXml.Size =
new System.Drawing.Size(16, 16);
this.btnDeleteXml.TabIndex = 1;
this.btnDeleteXml.UseVisualStyleBackColor = true;
this.btnDeleteXml.Click +=
new System.EventHandler(
this.btnDeleteXml_Click);
//
// treeViewQueries
//
this.treeViewQueries.Dock =
System.Windows.Forms.DockStyle.Fill;
this.treeViewQueries.ImageIndex = 3;
this.treeViewQueries.ImageList =
this.imageList;
this.treeViewQueries.Location =
new System.Drawing.Point(3, 19);
this.treeViewQueries.Name = "treeViewQueries";
this.treeViewQueries.SelectedImageIndex = 3;
this.treeViewQueries.Size =
new System.Drawing.Size(199, 405);
this.treeViewQueries.TabIndex = 0;
this.treeViewQueries.AfterSelect +=
new System.Windows.Forms.
TreeViewEventHandler(
this.treeViewQueries_AfterSelect);
//
// lblMode
//
this.lblMode.AutoSize = true;
this.lblMode.Font = new
System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Italic,
System.Drawing.GraphicsUnit.Point,
((byte)(0)));
this.lblMode.Location =
new System.Drawing.Point(19, 60);
this.lblMode.Name = "lblMode";
this.lblMode.Size =
new System.Drawing.Size(42, 15);
this.lblMode.TabIndex = 19;
this.lblMode.Text = "Mode:";
//
// lblMode2
//
this.lblMode2.AutoSize = true;
this.lblMode2.Font = new System.Drawing.Font(
"Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Italic,
System.Drawing.GraphicsUnit.Point,
((byte)(0)));
this.lblMode2.ForeColor =
System.Drawing.Color.Blue;
this.lblMode2.Location =
```

```
            new System.Drawing.Point(67, 62);
            this.lblMode2.Name = "lblMode2";
            this.lblMode2.Size =
            new System.Drawing.Size(66, 13);
            this.lblMode2.TabIndex = 20;
            this.lblMode2.Text = "Unprotected";
            //
            // btnLock
            //
            this.btnLock.BackColor =
            System.Drawing.SystemColors.Control;
            this.btnLock.FlatStyle =
            System.Windows.Forms.FlatStyle.Popup;
            this.btnLock.ImageIndex = 5;
            this.btnLock.ImageList = this.imageList;
            this.btnLock.Location =
            new System.Drawing.Point(140, 56);
            this.btnLock.Name = "btnLock";
            this.btnLock.Size =
            new System.Drawing.Size(26, 23);
            this.btnLock.TabIndex = 21;
            this.btnLock.UseVisualStyleBackColor = false;
            this.btnLock.Click += new System.EventHandler(
            this.btnLock_Click);
            //
            // iQueries
            //
            this.BackColor = System.Drawing.SystemColors.Control;
            this.Controls.Add(this.btnLock);
            this.Controls.Add(this.lblMode2);
            this.Controls.Add(this.lblMode);
            this.Controls.Add(this.lblTitle);
            this.Controls.Add(this.lblDynamic);
            this.Controls.Add(this.panelMain);
            this.Controls.Add(this.tsContenedor);
            this.Name = "iQueries";
            this.Size = new System.Drawing.Size(1125, 706);
            this.Load += new System.EventHandler
            (this.iQueries_Load);
            this.tsContenedor.ResumeLayout(false);
            this.tsContenedor.PerformLayout();
            this.panelMain.ResumeLayout(false);
            this.panelMain.PerformLayout();
            ((System.ComponentModel.ISupportInitialize)
            (this.dgvResult)).EndInit();
            this.gbSavedQueries.ResumeLayout(false);
            this.ResumeLayout(false);
            this.PerformLayout();

}
private void iQueries_Load(object sender, EventArgs e)
{
            if (UIServices.KB != null &&
            UIServices.KB.CurrentModel != null)
            {
                        CMessageOutput.MessageStart("iQueries Started");
```

```
                    Folder folder = Folder.Get(UIServices.KB.CurrentModel,
                    "iQueries");
                    if (folder == null)
                    {
                            folder = new Folder(UIServices.KB.CurrentModel);
                            folder.Parent = Folder.GetRoot(UIServices.KB.
                            CurrentModel);
                            folder.Name = "iQueries";
                            folder.Save();
                            this.createProcedureConfig();
                    }
                    if (!this.existeDirectorio())
                    {
                            this.crearDirectorio();
                            CXml.SavePropertiesKB(CKb.GetDefaultDataStore());
                    }
                    this.ClearObjects();
                    this.GetTables();
            }
    }
    private void btnXml_Click(object sender, EventArgs e)
    {
            this.ListSavedQueries();
            if (this.treeViewQueries.Nodes.Count > 0)
            {
                    this.ClearObjects();
                    this.ListTables();
                    this.ListSavedQueries();
                    this.unprotected = true;
                    this.isProtected();
                    this.treeViewQueries.Focus();
                    CMessageOutput.MessageSuccess("Se han recuperado
                    todas las consultas exitósamente...");
            }
            else
            {
                    CMessageOutput.MessageLine("No se ah guardado
                    ninguna consulta...");
            }
    }
    private void btnSql_Click(object sender, EventArgs e)
    {
            if (hashParents.Count > 0)
            {
                    this.SetSqlQuery("");
                    List<string> strs = new List<string>();

                    IEnumerator<Table> tables = Table.GetAll(
                    UIServices.KB.CurrentModel)
                    .GetEnumerator();

                    while (tables.MoveNext())
                    {
                            Table table = (Table)tables.Current;
                            if (hashParents.ContainsKey(table.Name))
                            {
```

```
                                IEnumerator<TableAttribute> atts
                                = table.TableStructure.
                                Attributes.GetEnumerator();
                                string nameParent = table.Name;
                                while (atts.MoveNext())
                                {
                                        Artech.Genexus.Common.Objects.Attribute
                                        att =
                                        (Artech.Genexus.Common.Objects.
                                        Attribute)atts.Current;
                                        if (hashChilds.ContainsKey(nameParent +
                                        att.Name))
                                        {
                                                if (strs.IndexOf(att.Name) != -1)
                                                {
                                                        continue;
                                                }
                                                strs.Add(att.Name);
                                                QueryElement queryElement =
                                                new QueryElement(
                                                CQuery.QueryObject.
                                                QueryStructurePart,
                                                att, true);
                                                CQuery.AddQueryElement(
                                                queryElement);
                                        }
                                }
                        }
                }

                string getSQLSentence = CQuery.GetQuery(hashElements);
                if (string.IsNullOrEmpty(getSQLSentence))
                {
                        CMessageOutput.MessageFailed("Las relaciones de
                        las tablas no están bien hechas");
                }
                this.SetSqlQuery(getSQLSentence);
        }
        else
        {
                CMessageOutput.MessageFailed("Debe seleccionar
                atributos para formar la consulta...");
        }
}
private void btnRunSql_Click(object sender,
EventArgs e)
{
        if (txtSql.Text.Length > 0)
        {
                launcher = new CLauncher();
                this.dgvResult.DataSource =
                launcher.ExecuteQuery(txtSql.Text);
        }
```

```
            else
            {
                CMessageOutput.MessageFailed("Debe generar
                la consulta...");
            }
}
private void btnSaveSql_Click(object sender, EventArgs e)
{
            if (this.txtNameQuery.Text.Length > 0 &&
            this.txtSql.Text.Length > 0)
            {
                if (hashParents.Count == 0 || hashChilds.Count == 0)
                {
                    CMessageOutput.MessageFailed("No hay ninguna
                    consulta que guardar...");
                }
                else
                {
                    if (CXml.SaveQuery(hashParents, hashChilds,
                    this.txtNameQuery.Text,
                    this.txtSql.Text))
                    {
                        if (this.createSdtQuery())
                        {
                            if (this.createProcedureQuery())
                            {
                                this.treeViewQueries.Nodes.Clear();
                                this.ListSavedQueries();
                                CMessageOutput.MessageSuccess("La
                                consulta se ah guardado
                                exitósamente...");
                            }
                        }
                    }
                }
            }
            else
            {
                CMessageOutput.MessageFailed("Necesita ingresar
                un nombre a la consulta o
                necesita generar la consulta...");
            }

}
private void btnClear_Click(object sender, EventArgs e)
{
            this.ClearObjects();
            this.ListTables();
            this.treeView.Focus();
            this.unprotected = false;
            this.isProtected();
}
```

```
private void btnLock_Click(object sender, EventArgs e)
{
	if (this.unprotected)
	{
		this.unprotected = false;
	}
	else
	{
		this.unprotected = true;
	}
	this.txtSql.Text = "";
	this.txtNameQuery.Text = "";
	this.dgvResult.DataSource = new DataTable();
	this.isProtected();
}
private void treeView_AfterCheck(object sender,
TreeViewEventArgs e)
{
	if (e.Action != TreeViewAction.Unknown)
	{
		if (e.Node.Parent == null)
		{
			if (hashParents.ContainsKey(e.Node.Name))
			{
				hashParents.Remove(e.Node.Name);
				foreach (TreeNode node in e.Node.Nodes)
				{
					node.Checked = false;
					hashChilds.Remove(node.Parent.Name +
					node.Name);
					hashElements.Remove(node.Name);
				}
			}
			else
			{
				hashParents.Add(e.Node.Name, e.Node.Name);
				foreach (TreeNode node in e.Node.Nodes)
				{
					node.Checked = true;
					hashChilds.Add(node.Parent.Name +
					node.Name, node.Name);
					hashElements.Remove(node.Name);
					hashElements.Add(node.Name, node.Name);
				}
			}
		}
		else
		{
			if (!hashParents.ContainsKey(e.Node.Parent.Name))
			{
				e.Node.Parent.Checked = true;
				hashParents.Add(e.Node.Parent.Name,
				e.Node.Parent.Name);
				e.Node.Checked = true;
				hashChilds.Add(e.Node.Parent.Name + e.Node.Name,
				e.Node.Name);
```

```
                        hashElements.Remove(e.Node.Name);
                        hashElements.Add(e.Node.Name, e.Node.Name);
                    }
                    else
                    {
                        if (hashChilds.ContainsKey(e.Node.Parent.Name +
                        e.Node.Name))
                        {
                            e.Node.Checked = false;
                            hashChilds.Remove(e.Node.Parent.Name +
                            e.Node.Name);
                            hashElements.Remove(e.Node.Name);
                            TreeNode parent = e.Node.Parent;
                            int count = 0;
                            foreach (TreeNode node in parent.Nodes)
                            {
                                if (node.Checked)
                                {
                                    count += 1;
                                }
                            }
                            if (count == 0)
                            {
                                e.Node.Parent.Checked = false;
                                hashParents.Remove(parent.Name);
                            }
                        }
                        else
                        {
                            e.Node.Checked = true;
                            hashChilds.Add(e.Node.Parent.Name +
                            e.Node.Name, e.Node.Name);
                            hashElements.Remove(e.Node.Name);
                            hashElements.Add(e.Node.Name,
                            e.Node.Name);
                        }
                    }
                }
            }
        }
        private void btnDeleteXml_Click(object sender, EventArgs e)
        {
            if (this.treeViewQueries.SelectedNode != null)
            {
                string nameFile = this.treeViewQueries.
                SelectedNode.Name;
                string path = CKb.GetKBLocation();
                path = Path.Combine(path, "iQueries");
                path = Path.Combine(path, "admin");
                path = Path.Combine(path, nameFile);
                File.Delete(path);
                CMessageOutput.MessageSuccess("El archivo " +
                nameFile + " se eliminó correctamente...");
                this.ListSavedQueries();
                this.treeViewQueries.Focus();
            }
```

```
            else
            {
                CMessageOutput.MessageFailed("Necesita elegir
                un archivo XML...");
            }
}
private void treeViewQueries_AfterSelect(object sender,
TreeViewEventArgs e)
{
            this.unprotected = true;
            this.isProtected();
            hashElementsXml.Clear();
            string path = CKb.GetKBLocation();
            path = Path.Combine(path, "iQueries");
            path = Path.Combine(path, "admin");
            path = Path.Combine(path, e.Node.Name);

            XmlDataDocument xmlDataDocument = new XmlDataDocument();
            XmlNodeList level0;
            Hashtable localParents = new Hashtable();
            Hashtable localChilds = new Hashtable();

            CMessageOutput.MessageLine(path);
            FileStream fs = new FileStream(path, FileMode.Open,
            FileAccess.Read);
            try
            {
                xmlDataDocument.Load(fs);
                if (xmlDataDocument.HasChildNodes)
                {
                    level0 = xmlDataDocument.ChildNodes;
                    foreach (XmlNode node in level0)
                    {
                        XmlNodeList level1 = node.ChildNodes;
                        foreach (XmlNode node2 in level1)
                        {
                            XmlNodeList level2 = node2.ChildNodes;
                            localParents.Add(node2.Name, node2.Name);
                            foreach (XmlNode node3 in level2)
                            {
                                localChilds.Add(node2.Name +
                                node3.Name, node3.Name);
                                if (!hashElementsXml.ContainsKey
                                node3.Name))
                                {
                                    hashElementsXml.Add(
                                    node3.Name,
                                    node3.Name);
                                }
                            }
                        }
                    }
                    fs.Close();
                    TreeNodeCollection nodes = treeView.Nodes;
                    foreach (TreeNode parent in nodes)
                    {
```

```
                                parent.Checked = false;
                                parent.Collapse();
                                TreeNodeCollection childs = parent.Nodes;
                                foreach (TreeNode child in childs)
                                {
                                        child.Checked = false;
                                }
                        }
                        foreach (TreeNode parent in nodes)
                        {
                                if (localParents.ContainsKey(parent.Name))
                                {
                                        parent.Checked = true;
                                        parent.Expand();
                                        TreeNodeCollection childs = parent.Nodes;
                                        foreach (TreeNode child in childs)
                                        {
                                                if (localChilds.ContainsKey(
                                                parent.Name +
                                                child.Name))
                                                {
                                                        child.Checked = true;
                                                }
                                        }
                                }
                        }
                        this.BuildQueryXml(localParents, localChilds);
                }
                else
                {
                        CMessageOutput.MessageFailed("El documento
                        está vacio");
                }
        }
        catch (FileLoadException ex)
        {
                CMessageOutput.MessageFailed("El documento está
                vacio - " + ex.Message);
        }
}
private void isProtected()
{
        if (this.unprotected)
        {
                this.lblMode2.ForeColor = Color.Red;
                this.lblMode2.Text = "Protected";
                this.btnLock.ImageIndex = 4;
                this.treeView.Enabled = false;
                this.txtSql.ReadOnly = true;
                this.txtNameQuery.Enabled = false;
                this.btnLock.Enabled = true;
        }
        else
        {
                this.lblMode2.ForeColor = Color.Blue;
                this.lblMode2.Text = "Unprotected";
```

```
                this.btnLock.ImageIndex = 5;
                this.treeView.Enabled = true;
                this.txtSql.ReadOnly = false;
                this.txtNameQuery.Enabled = true;
                this.btnLock.Enabled = false;
                this.clearNodes();
        }
}
private void clearNodes()
{
        TreeNodeCollection parents = treeView.Nodes;
        foreach (TreeNode parent in parents)
        {
                parent.Collapse();
                parent.Checked = false;
                foreach (TreeNode node in parent.Nodes)
                {
                        node.Checked = false;
                }
        }
}
private void SetSqlQuery(string sqlQuery)
{
        this.txtSql.Text = sqlQuery;
        this.txtSql.Refresh();
}
private void ClearObjects()
{
        this.treeView.Nodes.Clear();
        this.txtNameQuery.Text = "";
        this.txtSql.Text = "";
        this.dgvResult.DataSource = new DataTable();
        this.treeViewQueries.Nodes.Clear();
        this.btnLock.Enabled = true;
        this.lblMode2.Text = "Unprotected";
}
public bool OnSelectChange(ISelectionContainer sc)
{
        return true;
}

private void crearDirectorio()
{
        string path = CKb.GetKBLocation();
        path = Path.Combine(path, "iQueries");
        if (!Directory.Exists(path))
        {
                Directory.CreateDirectory(path);
                path = Path.Combine(path, "admin");
                Directory.CreateDirectory(path);
        }
}
private bool existeDirectorio()
{
        string path = CKb.GetKBLocation();
        path = Path.Combine(path, "iQueries");
```

```
            if (!Directory.Exists(path))
            {
                return false;
            }
            else
            {
                return true;
            }
}
private void createProcedureConfig()
{
            if (Procedure.Get(UIServices.KB.CurrentModel,
            "iQueriesConfig") == null)
            {
                PropertiesHelper propertiesHelper = new PropertiesHelper(
                CKb.GetDefaultDataStore());
                string server = propertiesHelper.GetProperty<string>
                ("CS_SERVER");
                string dbname = propertiesHelper.GetProperty<string>
                ("CS_DBNAME");
                string dbms_port = propertiesHelper.GetProperty<string>
                ("DBMS_PORT");
                string user = propertiesHelper.GetProperty<string>
                ("USER_ID");
                string password = propertiesHelper.GetProperty<string>
                ("USER_PASSWORD");
                string trusted = propertiesHelper.GetProperty<string>
                ("TRUSTED_CONNECTION");

                Folder folder = Folder.Get(UIServices.KB.CurrentModel,
                "iQueries");
                Procedure procConfig = new Procedure(
                CKb.GetCurrentModel());
                procConfig.Name = "iQueriesConfig";
                procConfig.Parent = folder;
                procConfig.Save();

                Variable Server = new Variable(procConfig.Variables);
                Server.Name = "server";
                Server.Type = eDBType.VARCHAR;
                Server.Length = 200;
                procConfig.Variables.Add(Server);
                procConfig.Save();

                Variable Dbname = new Variable(procConfig.Variables);
                Dbname.Name = "dbname";
                Dbname.Type = eDBType.VARCHAR;
                Dbname.Length = 50;
                procConfig.Variables.Add(Dbname);
                procConfig.Save();

                Variable Dbms_port = new Variable(procConfig.Variables);
                Dbms_port.Name = "dbmsport";
                Dbms_port.Type = eDBType.VARCHAR;
                Dbms_port.Length = 10;
                procConfig.Variables.Add(Dbms_port);
```

```
                procConfig.Save();

                Variable User = new Variable(procConfig.Variables);
                User.Name = "user";
                User.Type = eDBType.VARCHAR;
                User.Length = 60;
                procConfig.Variables.Add(User);
                procConfig.Save();

                Variable Password = new Variable(procConfig.Variables);
                Password.Name = "password";
                Password.Type = eDBType.VARCHAR;
                Password.Length = 60;
                procConfig.Variables.Add(Password);
                procConfig.Save();

                Variable Trusted = new Variable(procConfig.Variables);
                Trusted.Name = "trusted";
                Trusted.Type = eDBType.VARCHAR;
                Trusted.Length = 2;
                procConfig.Variables.Add(Trusted);
                procConfig.Save();

                string[] source = new string[6];
                source[0] = "&server = '" + server + "'\r\n";
                source[1] = "&dbname = '" + dbname + "'\r\n";
                source[2] = "&dbmsport = '" + dbms_port + "'\r\n";
                source[3] = "&user = '" + user + "'\r\n";
                source[4] = "&password = '" + password + "'\r\n";
                source[5] = "&trusted = '" + trusted + "'\r\n";
                procConfig.ProcedurePart.Source = string.Concat(source);
                procConfig.Rules.Source = "parm(out:&server,
                out:&dbname,out:&dbmsport,out:&user,out:&password,
                out:&trusted);";
                procConfig.Save();
                CMessageOutput.MessageSuccess("iQueriesConfig
                se creó con éxito!");
        }
}
private bool createProcedureQuery()
{
        Folder folder = Folder.Get(UIServices.KB.CurrentModel,
        "iQueries");
        string name = "Proc" + this.txtNameQuery.Text.Trim();
        bool success = false;
        Procedure procQuery = Procedure.Get(CKb.GetCurrentModel(),
        name);
        if (procQuery == null)
        {
                try
                {
                        procQuery = new Procedure(CKb.GetCurrentModel());
                        procQuery.Parent = folder;
                        procQuery.Name = name;
                        procQuery.Rules.Source = "parm(out:&query);";
                        procQuery.Save();
```

```
                        Variable var = new Variable(procQuery.Variables);
                        var.Name = "query";
                        var.Type = eDBType.VARCHAR; //definimos el tipo
                        var.Length = 9999;          //definimos el tamaño
                        procQuery.Variables.Add(var);
                        procQuery.Save();
                        string sql = this.txtSql.Text.Trim();
                        sql = sql.Replace("\"", "'");
                        string[] source = new string[3];
                        source[0] = "&query = \"";
                        source[1] = sql;
                        source[2] = "\"";
                        string cadena = string.Concat(source);
                        procQuery.ProcedurePart.Source = cadena;
                        procQuery.Save();
                        success = true;
                        CMessageOutput.MessageSuccess("El " +
                        procQuery.Name + " ah sido creado con éxito...");
                }
                catch (Exception ex)
                {
                        CMessageOutput.MessageSuccess(ex.Message);
                }
        }
        else
        {
                CMessageOutput.MessageFailed("Ya existe el nombre:
                Proc" + this.txtNameQuery.Text.Trim());
        }
        return success;
}
private bool createSdtQuery()
{
        Folder folder = Folder.Get(CKb.GetCurrentModel(),
        "iQueries");
        string name = "Sdt" + this.txtNameQuery.Text.Trim();
        SDT sdtQuery = SDT.Get(CKb.GetCurrentModel(), name);
        bool success = false;
        if (sdtQuery == null)
        {
                try
                {
                        sdtQuery = new SDT(CKb.GetCurrentModel());
                        sdtQuery.Parent = folder;
                        string sdtName = name;
                        sdtQuery.Name = sdtName;
                        SDTLevel sdtLevel = sdtQuery.SDTStructure.Root;
                        sdtLevel.Name = "Item";
                        sdtLevel.Items.Clear();
                        sdtLevel.IsCollection = true;
                        sdtLevel.CollectionItemName = "Item";

                        foreach (DictionaryEntry entry in hashAttributes)
                        {
                                TableAttribute att = (TableAttribute)entry.Value;
                                string key = att.Table.Name + att.Name;
```

```
                                        if (hashChilds.ContainsKey(key))
                                        {
                                                SDTItem item = new SDTItem(
                                                sdtLevel.SDTStructure);
                                                item.Name = att.Attribute.Description;
                                                item.Type = att.Attribute.Type;
                                                item.Length = att.Attribute.Length;
                                                item.IsCollection = false;
                                                sdtLevel.AddItem(item);
                                        }
                                }
                                sdtQuery.Save();
                                CMessageOutput.MessageSuccess("El " +
                                sdtQuery.Name + " ah sido creado con éxito...");
                                success = true;
                        }
                        catch (Exception ex)
                        {
                                CMessageOutput.MessageSuccess(ex.Message);
                        }
                }
                else
                {
                        CMessageOutput.MessageFailed("Ya existe
                        el nombre: Sdt" + this.txtNameQuery.Text.Trim());
                }
                return success;
        }
        private void GetTables()
        {
                this.gxDataStore = CKb.GetDefaultDataStore();
                if (this.gxDataStore != null)
                {
                        this.treeView.Nodes.Clear();
                        this.ListTables();
                }
                else
                {
                        CMessageOutput.MessageError("No se pudo
                        obtener el DataStore por defecto.");
                }
        }
        private void ListSavedQueries()
        {
                hashElementsXml = new Hashtable();
                string path = CKb.GetKBLocation();
                path = System.IO.Path.Combine(path, "iQueries");
                path = Path.Combine(path, "admin");
                String[] files = Directory.GetFiles(path);
                treeViewQueries.Nodes.Clear();
                foreach (String file in files)
                {
                        TreeNode nodeXml = new TreeNode();
                        nodeXml.Name = Path.GetFileName(file);
                        nodeXml.Text = Path.GetFileName(file);
                        nodeXml.ImageIndex = 3;
```

```
                treeViewQueries.Nodes.Add(nodeXml);
            }
        }
        private void ListTables()
        {
            IEnumerator<Table> tables = Table.GetAll(
            UIServices.KB.CurrentModel).GetEnumerator();
            hashParents = new Hashtable();
            hashChilds = new Hashtable();
            hashElements = new Hashtable();
            hashAttributes = new Hashtable();
            while (tables.MoveNext())
            {
                Table table = tables.Current;
                BaseCollection<TableAttribute> atts =
                table.TableStructure.Attributes;
                TreeNode[] childrens = new TreeNode[atts.Count];
                int index = 0;
                foreach (TableAttribute att in atts)
                {
                    string keyAtt = att.Table.Name + att.Name;
                    hashAttributes.Add(keyAtt, att);
                    TreeNode child = new TreeNode(att.Name);
                    child.Name = att.Name;
                    child.Text = att.Name;
                    child.ImageIndex = 1;
                    childrens[index] = child;
                    index += 1;
                }
                TreeNode parent = new TreeNode(table.Name, childrens);
                parent.Name = table.Name;
                parent.Text = table.Name;
                parent.ImageIndex = 0;
                treeView.Nodes.Add(parent);
            }
        }
        private void BuildQueryXml(Hashtable parents, Hashtable childs)
        {
            this.SetSqlQuery("");
            List<string> strs = new List<string>();
            IEnumerator<Table> tables = Table.GetAll(
            UIServices.KB.CurrentModel).GetEnumerator();
            while (tables.MoveNext())
            {
                Table table = (Table)tables.Current;
                if (parents.ContainsKey(table.Name))
                {
                    IEnumerator<TableAttribute> atts =
                    table.TableStructure.Attributes.GetEnumerator();
                    string nameParent = table.Name;
                    while (atts.MoveNext())
                    {
                        Artech.Genexus.Common.Objects.Attribute att
                        = (Artech.Genexus.Common.Objects.Attribute)
                        atts.Current;
```

```
                                        if (childs.ContainsKey(nameParent + att.Name))
                                        {
                                                if (strs.IndexOf(att.Name) != -1)
                                                {
                                                        continue;
                                                }
                                                strs.Add(att.Name);
                                                QueryElement queryElement = new
                                                QueryElement(CQuery.QueryObject.
                                                QueryStructurePart, att, true);
                                                CQuery.AddQueryElementXml(queryElement);
                                        }
                                }
                        }
                }

                string getSQLSentence = CQuery.GetQueryXml(hashElementsXml);
                if (string.IsNullOrEmpty(getSQLSentence))
                {
                        CMessageOutput.MessageFailed("Las relaciones de
                        las tablas no están bien hechas");
                }
                this.SetSqlQuery(getSQLSentence);
        }
}
}
```

i. Package.cs

```
using Artech.Architecture.Common.Packages;
using Artech.Architecture.Common.Services;
using Artech.Architecture.UI.Framework.Objects;
using Artech.Architecture.UI.Framework.Controls;
using Artech.Architecture.UI.Framework.Services;
using Artech.Architecture.Common.Descriptors;
using Artech.Architecture.UI.Framework.Packages;
using System;
using System.Runtime.InteropServices;
using Microsoft.Practices.CompositeUI.EventBroker;
using Artech.Architecture.Common.Events;
using Artech.Architecture.Common.Objects;


namespace IvanGQ.iSQL
{
[Guid("1708ee5f-0f9c-4d8e-8590-c5eba3589015")]
public class Package : AbstractPackageUI
{
        public override string Name
        {
                get { return "iQueries"; }
        }

        public override void Initialize(
        IGxServiceProvider services)
        {
```

```
            base.Initialize(services);
        }

        private iQueries showSQL;

        public override IToolWindow CreateToolWindow(
        Guid toolWindowId)
        {
            if (toolWindowId.Equals(iQueries.guid))
            {
                if (showSQL == null)
                {
                    showSQL = new iQueries();
                }
                return showSQL;
            }
            return base.CreateToolWindow(toolWindowId);
        }

        [EventSubscription(ArchitectureEvents.AfterCloseKB)]
        public void OnAfterCloseKB(object sender,
        EventArgs args) {
            showSQL.Dispose();
        }
}
}
```

j. PropertiesHelper

```
using System;
using System.Collections.Generic;
using System.Text;
using System.ComponentModel;

namespace IvanGQ.iSQL
{
public class PropertiesHelper
{
        private object instance;

        public PropertiesHelper(object obj) {
            this.instance = obj;
        }

        private PropertyDescriptorCollection properties;
        public PropertyDescriptorCollection Properties {
            get {
                if(this.properties == null){
                    this.properties = TypeDescriptor.
                    GetProperties(this.instance);
                }
                return this.properties;
            }
        }

        public object GetProperty(string propName) {
```

```
                PropertyDescriptor item = this.Properties[propName];
                if (item != null)
                {
                        return item.GetValue(this.instance);
                }
                else {
                        return null;
                }
        }
        public T GetProperty<T>(string propName) {
                object property = this.GetProperty(propName);
                if (property == null) {
                        Type type = typeof(T);
                        if(type.IsValueType){
                                return (T)Activator.CreateInstance(type);
                        }
                }
                return (T)property;
        }
}
}
```

Anexo C: Encuesta

Nombres(s) y Apellidos: _____

Ocupación: _____

¿Qué le pareció la instalación de la extensión?

Muy Fácil	Fácil	Regular	Difícil	Muy Difícil

¿Qué le pareció la programación para realizar consultas dinámicas con la extensión?

Muy Fácil	Fácil	Regular	Difícil	Muy Difícil

¿Cómo califica en grado de utilidad a la extensión?

Muy útil	útil	Regular	Improductivo

¿Cómo le pareció la compresión de la guía de extensiones?

Muy Fácil	Fácil	Regular	Difícil	Muy Difícil

¿Qué le pareció los ejemplos de creación de objetos GeneXus en la guía de extensiones?

Muy útil	útil	Regular	Improductivo

OBSERVACIONES Y/O SUGERENCIAS:

_____

_____

_____

_____

_____