Ejercicio A.12: Gestión de datos. Almacenamiento aislado, Data Scope

Los datos almacenados en las tablas y campos de nuestra app son accesibles a través de otras apps.

En general, este es un comportamiento deseable, sin embargo hay algunas ocasiones en las que se debería proteger los datos almacenados. Por ejemplo:

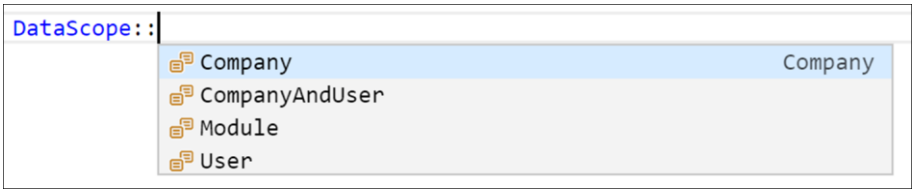
- **Datos de licenciamiento**. En caso que nuestra app tenga implementado un sistema de licenciamiento, los datos se deben guardar en algún lugar, pero no debe ser accesible ni modificable por otras apps.
- **Datos de acceso**. En caso que nuestra app tenga una integración con un servicio externo que requiera datos de conexión y contraseñas, estos datos se deben almacenar en un sitio que no sea accesible ni modificable por otras apps.

Almacenamiento aislado

La tabla de sistema 2000000107 Isolated Storage se usa para almacenar datos sensibles en forma de blob. Estos datos solo son accesibles por la app que ha almacenado dichos datos.

Data Scope

Los datos aislados solo pueden ser accesibles desde la propia app que los crea. Además de esta restricción, también podemos definir distintos niveles de acceso para limitar el acceso a una única empresa o a un único usuario.



- Module – Los datos son accesibles desde la app, para todas las empresas y usuarios.
- Company – Los datos son accesibles desde la app, para todos los usuarios, pero únicamente desde la empresa que guardó los datos.
- CompanyAndUser – Los datos son accesibles desde la App, pero solo para el usuario y la empresa desde el que se guardaron los datos.
- User – Los datos son accesibles desde la App, en todas las empresas, pero solo con el usuario que guardó los datos.

Documentación:
https://docs.microsoft.com/es-es/dynamics365/business-central/dev-itpro/developer/devenv-isolated-storage
https://www.kauffmann.nl/2018/10/30/isolated-storage-in-your-business-central-app/

Datos de conexión

- Entorno de desarrollo:
  - Servidor: localhost\NAVDEMO
  - BBDD: BCCU0_AppDevelopment
  - Autenticación: Windows

Indicaciones para el ejercicio

En vscode, vamos a crear una extensión en la que vamos a almacenar datos sensibles de forma aislada. Para ello, crearemos una tabla en la que:

- Insertaremos datos de forma temporal
- Transformaremos los datos de la tabla en formato JSON y los guardaremos en Isolated Storage
- Cada vez que necesitemos acceder a los datos de licencia, los descargaremos en la tabla, de forma temporal.

Después crearemos una segunda extensión para intentar acceder a los datos almacenados.

Indicaciones paso a paso

Sigue los siguientes pasos en el entorno de desarrollo C/SIDE:

1. En vscode, crea un nuevo proyecto y descarga los símbolos.
   1.1. Pulsa F1 para abrir la paleta de comandos.
   1.2. Ejecuta el comando AL: Go!
   Nombre del proyecto: Ejercicio A9 - Isolated Storage
   1.3. Modifica el fichero launch.json

```
"serverInstance": "BCCU0_AppDevelopment",
"authentication": "Windows",
```

   1.4. Ejecuta el comando AL: Download symbols.

2. Crea una nueva tabla llamada MyLicenseTable

Tab.50100.MyLicenseTable.al

```
table 50100 "MyLicenseTable"
{
    fields
    {
        field(1; "Primary key"; Code[10]) { }
    }

    keys
    {
        key(PK; "Primary key")
        {
            Clustered = true;
```

3. Para poder ofrecer una versión de prueba por un tiempo limitados, crea los siguientes campos en la tabla:

```
fields
{
    field(1; "Primary key"; Code[10]) { }
    field(2; "Demo activation"; DateTime) { }
    field(3; "Demo valid until"; DateTime) { }
}
```

4. En el momento de Instalar la App, nos vamos a guardar la fecha y hora de la primera instalación.
   4.1. Crear una codeunit de subtipo Install.

```
codeunit 50100 "InstallMgt"
{
    Subtype = Install;

    trigger OnInstallAppPerCompany()
    begin

    end;
}
```

   4.2. En el trigger OnInstallAppPerCompany añadir el siguiente código

```
trigger OnInstallAppPerCompany()
var
    LicenseMgt: Codeunit LicenseMgt; //> Crear variable
begin
    if InstalledForTheFirstTime() then //> Añadir código
      LicenseMgt.ActivateDemoTrial();
end;
```

   4.3. Crea el procedimiento InstalledForTheFirstTime

```
local procedure InstalledForTheFirstTime(): Boolean
var
    ExtensionInfo: ModuleInfo;
begin
    // La versión 0.0.0.0 indica que se trata de la primera instalación de la app
    // en esta base de datos, ya que no existen datos previos almacenados
    if ExtensionInfo.DataVersion() = Version.Create(0, 0, 0, 0) then
        exit(true);
    exit(false); // En caso que haya datos previos, se trata de una reinstalación
end;
```

   4.4. Crea la codeunit LicenseMgt y el procedimiento ActivateDemoTrial.
   El procedimiento debe declarar una variable temporal de la tabla MyLicenseTable.
   En el campo "Demo activation" le pondremos la fecha/hora actual
   En el campo "Demo valid until" le pondremos la fecha/hora actual más 30 días.

Cod.50101.LicenseMgt.al

```
codeunit 50101 "LicenseMgt"
{
    procedure ActivateDemoTrial()
    begin
        MyLicense.Init();
        MyLicense."Demo activation" := CurrentDateTime();
        MyLicense."Demo valid until" := MyLicense."Demo activation" + Days(30);
        MyLicense.insert;
    end;

    local procedure Days(Days: Decimal): Decimal
    begin
        EXIT(Days * 24 * 60 * 60 * 1000); //Convert to Milliseconds
    end;

    var
        MyLicense: Record MyLicenseTable temporary;
}
```

4.5. Para poder hacer las pruebas, cambia 30 días por 0.002 días. Esto son aproximadamente 3 minutos.

```
MyLicense."Demo valid until" := MyLicense."Demo activation" + Days(0.002);
```

El siguiente paso es guardar la información en la tabla Isolated Storage.

Sin embargo no podemos guardar un Record, sino únicamente cadenas de texto, que se almacenarán en forma de blob.

Es por ello que vamos a transformar la información del Record a texto, usando el formato JSON.

5. En la tabla MyLicenseTable, crea el procedimiento global WriteToJson

```
local procedure WriteToJson(): Text
var
    Json: Codeunit "Json Text Reader/Writer";
begin
    Json.WriteStartObject('');
    Json.WriteStringProperty(FieldName("Demo activation"), "Demo activation");
    Json.WriteStringProperty(FieldName("Demo valid until"), "Demo valid until");
    Json.WriteEndObject();
    EXIT(Json.GetJSonAsText());
end;
```

6. Edita la tabla MyLicenseTable.
   Añade el procedimiento StoreLicense para guardar la licencia en la tabla IsolatedStorage.

```
procedure StoreLicense() //>> Crear este procedimiento
var
    MyLicenseString: Text;
begin
    MyLicenseString := WriteToJson();
    if not IsolatedStorage.Contains(StorageKey, DataScope::Module) then
        IsolatedStorage.set(StorageKey, MyLicenseString, DataScope::Module);
end;

local procedure StorageKey(): Text //>> Crear este procedimiento
begin
    exit('b2294b68-a3e3-44be-ae61-9a021e828897-Lic');
end;
```

7. Edita la codeunit LicenseMgt. Añade una llamada al procedimiento StoreLicense.

```
procedure ActivateDemoTrial()
begin
    MyLicense.Init();
    MyLicense."Demo activation" := CurrentDateTime();
    MyLicense."Demo valid until" := MyLicense."Demo activation" + Days(0.002);
    MyLicense.insert;
    MyLicense.StoreLicense(); //>> Añadir esta línea
end;
```

8. Más adelante vamos a necesitar recuperar los datos guardados como almacenamiento aislado. En la tabla MyLicenseTable, crea los siguientes procedimientos:

```
local procedure LoadFromJson(JsonTxt: Text)
var
    Json: Codeunit "Json Text Reader/Writer";
    JsonBuffer: Record "JSON Buffer" temporary;
    AuxValue: text;
begin
    Json.ReadJSonToJSonBuffer(JsonTxt, JsonBuffer);

    DeleteAll();
    Init();
    JsonBuffer.GetPropertyValue(AuxValue, FieldName("Demo activation"));
    Evaluate("Demo activation", AuxValue);

    JsonBuffer.GetPropertyValue(AuxValue, FieldName("Demo valid until"));
    Evaluate("Demo valid until", AuxValue);
    insert;
end;
```

```
procedure RetrieveLicense()
var
    MyLicenseString: Text;
begin
    if IsolatedStorage.Contains(StorageKey, DataScope::Module) then begin
        IsolatedStorage.Get(StorageKey, DataScope::Module, MyLicenseString);
        LoadFromJson(MyLicenseString);
    end;
end;
```

9. En la codeunit LicenseMgt, crea el procedimiento IsAppActive.
   Este procedimiento se podrá usar después en todos las áreas relevantes de la aplicación para ejecutar una lógica de negocio u otra en función de si estamos en el periodo de prueba o no.

```
procedure IsAppActive(): Boolean
begin
    GetMyLicense;
    if MyLicense."Demo valid until" > CurrentDateTime() then
        exit(true);
    exit(false);
end;

local procedure GetMyLicense()
begin
    if MyLicense.IsEmpty() then
        MyLicense.RetrieveLicense;
end;

var
    MyLicense: Record MyLicenseTable temporary;
```

Una vez tenemos el módulo de licenciamiento funcionando, podemos usarlo en nuestro código.

```
pageextension 50100 "HelloWorld" extends "Customer List"
{
    trigger OnOpenPage();
    begin
        //Message('App published: Hello world');
        if LicenseMgt.IsAppActive then
            Message('App is Active')
        else
            Message('App demo is over')
    end;

    var
        LicenseMgt: codeunit LicenseMgt;
}
```