Ejercicio de creación de tablas

Tabla User MGS

No.	Integer (AutoIncrement)
Name	Text[100]
Email	Text[100]
External Id	Code[36]
Mas los campos que necesitéis...	

Tabla de sincronización

Direction	Enum(In,Out)
Action	Enum(Update User)
Data	Blob (JSON a sincronizar en texto)
Status	Enum(On hold, In progress, Success, Error)
Error	Blob (Se guardará solo texto, pero no sabemos el tamaño del texto)
Mas los campos que necesitéis...	

Tabla de configuración

API REST Base URL	Text[100]
Mas los campos que necesitéis...	

Mas las tablas que necesitéis...

Creación de Pages

Crear page tipo lista de usuarios

Page tipo lista de Tabla de sincronización. Los Blob de texto se tienen que ver en un campo de texto.

Crear page tipo card de tabla de configuración con un solo registro.

Creación de Codeunits de cola de proyectos

Crear Codeunit para enviar las actualizaciones de usuarios desde BC a la API REST (Direction en Out y Status On hold) y API REST a la BC (usando el campo updatedAt y también el campo (de la API REST) updateBy = local para no actualizar 2 veces la misma información). La cual se ejecutará en la cola de proyectos. De API REST a BC solo guardar el JSON a usar para actualizar en la tabla de sincronización. Si durante la consulta surge algún error guardar los datos del error en el campo Error y Status en Error.

Crear Codeunit que también estará en la cola de proyectos para actualizar los usuarios usando la tabla se sincronización (solo si Direction es In y Status es On hold).

Codeunti de suscripción de eventos

Utilizar evento de modificación de tabla usuario para generar nuevo registro en tabla de sincronización (Direction en Out y Status On hold).

Creación de API Page

Creación de API Page de usuarios.

Creación de Codeunit CRUD

Crear Codeunit como un CRUD de usuarios para usarlo desde WebService usando JSON.

Creacion de Pila en

Información extra

API REST

Utilizad el JsonServer.exe para ejecutar la API REST en local. Ejecutandolo sin parametros usara el puesto 3000 y el fichero ./db.json, sino pasad el puerto y el nombre del fichero a usar ./build/JsonServer.exe 3000 ./db.json, para generar el fichero db.json usad ./build/JsonServer.exe get-json 100 2024-02-25T21:11:48.110Z 2024-02-25T22:11:48.110Z para generar 100 registros con fechas aleatoria entre 2024-02-25T21:11:48.110Z y 2024-02-25T22:11:48.110Z.

Para filtrar por fechas se usad el formato ISO.

En la raíz http://localhost:3000/ tendréis información de la API REST y un enlace a la documentación a GitHub, es la versión mejor 0 usad la rama del GitHub v0 para ver la correcta

Docker en Cola de proyectos

Es posible que notéis que la cola de proyectos no funciona de manera programada en Docker usad este script para habilitarla:

```
$containerName = "vuestro nombre del container de Docker"
 Invoke-ScriptInNavContainer -containername $containerName -scriptblock {
    Set-NavServerConfiguration -ServerInstance BC -KeyName EnableTaskScheduler -KeyValue
true
    Set-NavServerInstance -ServerInstance BC -restart
 }
```

Para hacer peticiones OData o SOAP utilizad los puertos por defecto:
https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/deployment/deploy-single-computer-environment

Formato ISO de DateTime en BC

Para obtener el formato ISO de un DateTime usad este procedure:

```
procedure GetISODateTimeFromDateTime(DateTime: DateTime): Text
var
    TypeHelper: Codeunit "Type Helper";
    DateTimeFormatISOLbl: Label 'yyyy-MM-ddTHH:mm:ss.fffZ';
    DataFormattingCultureLbl: Label 'en-US';
begin
    exit(TypeHelper.FormatUtcDateTime(DateTime, DateTimeFormatISOLbl, DataFormattingCultureLbl));
end;
```

Documentos de formación y enlaces de interés

API

https://businesscentralgeek.com/json-full-guide-in-business-central#:~:text=In%20Business%20Central%2C%20a%20JSON,reading%20data%20from%20a%20JSON.

https://www.youtube.com/watch?v=BpLJy6HwXyo