❌ Errores del proyecto - Purchase Orders

Error Inicial para la conexión con la base de datos, tanto el DefaultConnection como el SqlDBContext de mi ConnectionStrings deben tienen la misma cadena de conexión como se muestra a continuación:

```
"ConnectionStrings": {

"DefaultConnection": "Server=DESKTOP-U4A8Q3O\\SQLEXPRESS01;Database=PurchaseOrders;User Id=myuser;Password=mypassword;TrustServerCertificate=True;",

"SqlDBContext": "Server=DESKTOP-U4A8Q3O\\SQLEXPRESS01;Database=PurchaseOrders;User Id=myuser;Password=mypassword;TrustServerCertificate=True;"

},
```

Error mas repetido, para la versión del .NET 8 ya esta obsoleta la forma de mostrar ventanas emergentes de esta forma: .Show

await this.DialogAddEditOrderLine.Show();

para ello cambiamos esa lineal por

IsOrderLineDialogVisible = true;

agregamos esto al inicio de @code

```
// Esta es la nueva propiedad para controlar la visibilidad
bool IsOrderLineDialogVisible { get; set; } = false; // Inicialmente oculto
```

Nuestro sfDialog lo tenemos asi:

```
<SfDialog @ref="DialogDeleteOrderLine" IsModal="true" Width="500px" ShowCloseIcon="true" Visible="false">
```

cambiamos la ultima intrucción por
<SfDialog @ref="DialogAddEditOrderLine" IsModal="true" Width="600px" ShowCloseIcon="true" @bind-Visible="@IsOrderLineDialogVisible">

17/07/25

Para mañana revisar la parte de los correos porque no funciona en PurchaseOrdersPage.razor:

el nombre del metodo es: OnInitializedAsync()

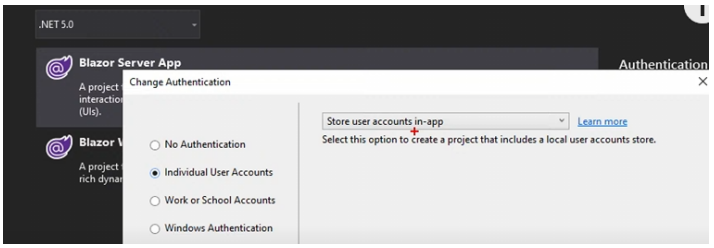
mas especificamente la condicional de if:

```
var authState = await AuthenticationStateProvider.GetAuthenticationStateAsync();
var user = authState.User;

if (user.Identity.IsAuthenticated)
{
UserName =
user.Identity.Name;
}
else
{
UserName = "The user is NOT authenticated.";
}
```

SOLUCIONANDO PROBLEMAS CON EL LOGIN O AUTENTIFICACIÓN:

1. Hice cambios en Program.cs



mañana arreglar el problema de inicio de cierre de sesión

El problema es que en mi Index. razor no se maneja bien el estado para cerrar cesión

//IPOHeaderService.cs

```
// Descomente este esta linea
Task<IEnumerable<POHeader>> POHeaderList();
//Task<IEnumerable<POHeader>> POHeaderList(string @UserName);
//significa que es anulable (ESTA LA AGREGE)
Task<IEnumerable<POHeader>> POHeaderList(string? UserName);
```

métodos en //POHeaderService.cs

```
// Quite el comentario de esta implementación para la sobrecarga sin argumentos.
// Asume que "spPOHeader_ListAll" o similar obtiene todas las órdenes.
// Si "spPOHeader_List" puede tomar un UserName nulo para obtener todo,
// podrías considerar eliminar esta y adaptar la otra. Pero por ahora,
// para evitar el error, la tendremos.
public async Task<IEnumerable<POHeader>> POHeaderList()
{
IEnumerable<POHeader> poheaders;
using (var conn = new SqlConnection(_configuration.Value))
{
// NOTA: Asegúrate de tener un SP que liste TODO si usas esto.
// Podrías reutilizar spPOHeader_List con un parámetro NULL
// O tener un SP diferente como spPOHeader_ListAll
poheaders = await conn.QueryAsync<POHeader>("spPOHeader_ListAll", commandType:
CommandType.StoredProcedure);
}
return poheaders;
}
// Implementación de la sobrecarga que toma un nombre de usuario.
// Aquí debes manejar si el UserName es nulo/vacío para decidir si filtrar o no.
public async Task<IEnumerable<POHeader>> POHeaderList(string? UserName) // 'string?' aquí también
{
IEnumerable<POHeader> poheaders;
var parameters = new DynamicParameters();

// Aquí está la lógica importante:
// Si UserName es nulo o vacío, pasamos DBNull.Value o simplemente no agregamos el parámetro,
// y el SP debe interpretarlo como "no filtrar".
// O, si tu SP espera un valor siempre, y null significa "todas", entonces está bien.
// Si tu SP espera un valor de cadena, considera pasar string.Empty o "[NULL]" si el SP lo maneja.

// Asumiendo que tu spPOHeader_List puede manejar un @UserName NULL para listar todo:
parameters.Add("@UserName", UserName, DbType.String); // DbType.String maneja el null correctamente con Dap
```

```
per

using (var conn = new SqlConnection(_configuration.Value))
{
    poheaders = await conn.QueryAsync<POHeader>("spPOHeader_List", parameters, commandType: CommandTy
pe.StoredProcedure);
}
return poheaders;
```

}

En Index.razor estos dos métodos quedaría así:

// Dentro de Index.razor en el bloque @code

protected override async Task OnInitializedAsync()
{
var authState = await authenticationStateTask;
var user = authState.User;

```
if (user.Identity?.IsAuthenticated == true)
{
    await GetOrderList();
}
else
{
    poheader = new List<POHeader>(); // Inicializa vacía si no autenticado
}

Toolbaritems.Add(new ItemModel() { Text = "Add", TooltipText = "Add a new order", PrefixIcon = "e-add" });
Toolbaritems.Add(new ItemModel() { Text = "Edit", TooltipText = "Edit selected order", PrefixIcon = "e-edit" });
Toolbaritems.Add(new ItemModel() { Text = "Delete", TooltipText = "Delete selected order", PrefixIcon = "e-delete"
});
Toolbaritems.Add(new ItemModel() { Text = "Preview", TooltipText = "Preview selected order", PrefixIcon = "e-prin
t" });
```

}

protected async Task GetOrderList()
{
var authState = await authenticationStateTask;
var user = authState.User;

```
// Ya sabemos que el usuario está autenticado si llegamos aquí,
// pero es bueno tener la verificación defensiva.
if (user.Identity?.IsAuthenticated == true)
{
    if (user.IsInRole("Admin") || user.IsInRole("Manager"))
    {
        // Llama a la sobrecarga que obtiene todas las órdenes (o el comportamiento para Admin/Manager)
        // Aquí puedes decidir si llamas a POHeaderList() (sin argumentos, si tienes SP_ListAll)
        // O a POHeaderList(null) si tu spPOHeader_List maneja null como "todas"
        poheader = await POHeaderService.POHeaderList(null); // O POHeaderService.POHeaderList(); si tu SP lo sop
orta.
    }
    else
    {
        // Llama a la sobrecarga que filtra por nombre de usuario
        poheader = await POHeaderService.POHeaderList(user.Identity.Name);
```

```
    }
  }
  else
  {
    // Esto no debería ser alcanzado si OnInitializedAsync ya lo filtró,
    // pero es una buena medida defensiva.
    poheader = new List<POHeader>();
  }
```

}

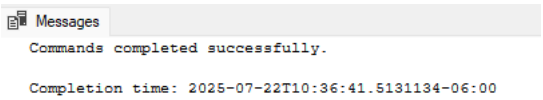
Para la autentificación, conexión con la BD
"DefaultConnection": "Server=DESKTOP-U4A8Q3O\\SQLEXPRESS01;Database=PurchaseOrdersAuth;Trusted_Connection=True;MultipleActiveResultSets=true;Tr
MIGRACIONES: todo debe estar el versión 8.0.0:

<Project Sdk="Microsoft.NET.Sdk.Web">

<PropertyGroup>
<TargetFramework>net8.0</TargetFramework>
<Nullable>enable</Nullable>
<ImplicitUsings>enable</ImplicitUsings>
<UserSecretsId>aspnet-BlazorPurchaseOrders-92cb059c-7f85-44b6-97e3-256b55d93d8a</UserSecretsId>
</PropertyGroup>

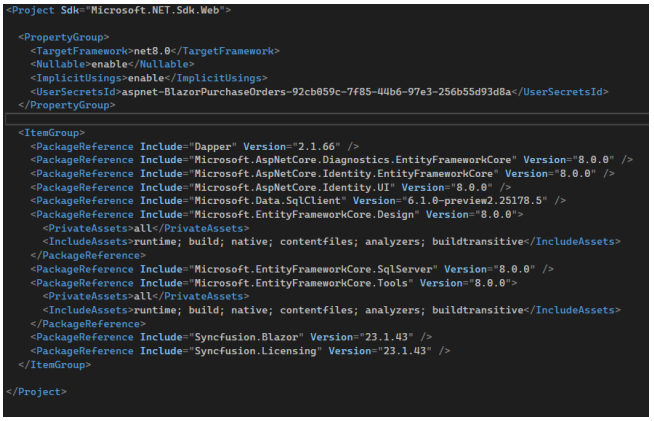
<ItemGroup>
<PackageReference Include="Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore" Version="8.0.0" />
<PackageReference Include="Microsoft.AspNetCore.Identity.EntityFrameworkCore" Version="8.0.0" />
<PackageReference Include="Microsoft.AspNetCore.Identity.UI" Version="8.0.0" />
<PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="8.0.0">
<PrivateAssets>all</PrivateAssets>
<IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
</PackageReference>
<PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="8.0.0" />
<PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="8.0.0">
<PrivateAssets>all</PrivateAssets>
<IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
</PackageReference>
</ItemGroup>

</Project>



ERRORES (2° vez):

primero verificar que se tengan estos package incluidos en el proyecto:



1. Este error se da por la mala conexión con la Base de Datos:

Dapper.SqlMapper.QueryAsync<T>(IDbConnection cnn, Type effectiveType, CommandDefinition command) in `SqlMapper.Async.cs`

BlazorPurchaseOrders.Data.TaxService.TaxList() in `TaxService.cs`

+

1. **taxes = await conn.QueryAsync<Tax>("spTax_List", commandType: CommandType.StoredProcedure);**

paréntesis: esta solución depende del la forma que se tenga la BD: Problema de la conection a BD

"DefaultConnection": "Server=DESKTOP-U4A8Q3O\\SQLEXPRESS01;Database=PurchaseOrders;Trusted_Connection=True;MultipleActiveResultSets=true;Trus

Agregue lo ultimo para la certificación

Solución: en appsettings.json la conexion a la Base de datos debe estar asi:

```
"ConnectionStrings": {
"DefaultConnection": "Server=DESKTOP-U4A8Q3O\\SQLEXPRESS01;Database=PurchaseOrdersAuth;Trusted_Connec
"SqlDBcontext": "Data Source=DESKTOP-U4A8Q3O\\SQLEXPRESS01;Database=PurchaseOrders;Integrated Security=
},
```

💡 PurchaseOrdersAuth: es la base de datos para la autenticación de usuarios

2. Error al cerrar un cuadro de dialogo:
   await this.DialogAddEditTax.Hide(); → el .Hide ya esta obsolete para la versión del .Net 8

solución: cambiamos esa linea de código por :
IsOrderLineDialogVisible = false; // Cierra el diálogo

3. Error, no muestra el dialogo de Add Tax Rate y eso para por problema de la licencia.

   Solución: en _Host.cshtml agregar lo siguiente
   <script src="_content/Syncfusion.Blazor/scripts/syncfusion-blazor.min.js" type="text/javascript"></script>

   en Program.cs agregar la licencia:

//Licencia de Syncfusion
Syncfusion.Licensing.SyncfusionLicenseProvider.RegisterLicense("Mzk1MjUzMEAzMjMzMmUzMDJlMzBEdHlwbWEzen

Por ultimo tener las siguientes versiones instaladas

<PackageReference Include="Syncfusion.Blazor" Version="23.1.43" />
<PackageReference Include="Syncfusion.Licensing" Version="23.1.43" />

4. Erro al hacer visible los cuadros de dialogo

Solución: agregar en PurchaseOrderPage.razor lo siguiente:

- <SfDialog @ref="DialogAddEditOrderLine" IsModal="true" Width="600px" ShowCloseIcon="true" @bind-Visible="@DialogVisible">
- Al inico de @code agregar : bool DialogVisible { get; set; } = false;
- En ToolbarClickHandler en Add, Edit agregar : DialogVisible = true;
- El siguiente método debe estar de la siguiente manera:

```
private async Task CloseDialog()
{
DialogVisible = true;
}
```

5. Error al hacer visible la ventana Delete en PurchaseOrderPage.razor

Solución: agregar en PurchaseOrderPage.razor lo siguiente:

```
<SfDialog @ref="DialogDeleteOrderLine" IsModal="true" Width="500px" ShowCloseIcon="true" @bind-Visible="@D
```

- Al inico de @code declarar **una propiedad** `bool` para controlar la visibilidad del diálogo de borrado:
  private bool DeleteDialogVisible { get; set; } = false;
- En ToolbarClickHandler en Delete eliminar `await this.DialogDeleteOrderLine.Show();` y agregar `DeleteDialogVisible = true;`
- Los siguientes métodos deben quedar así:

```
public void ConfirmDeleteNo()
{
DeleteDialogVisible = false;
selectedPOLineID = 0;
}
```

```
public void ConfirmDeleteYes()
{
OrderLineDelete();
DeleteDialogVisible = false;
selectedPOLineID = 0;
}
```

24/07/2025

ARREGLAR EL ERROR DE
ConfirmPage ConfirmOrderDelete;
EN
Smoothing out a few wrinkles

En Index.razor dentro del método ToolbarClickHandler en la opción de Preview cambiar esto:

1.

```
await IJS.InvokeAsync<object>("open", new object[] {
"/previeworder/" + selectedPOHeaderID + "", "_blank" })
```

por esto:

2.

```
await IJS.InvokeVoidAsync("open", new object[] {
"/previeworder/" + selectedPOHeaderID + "", "_blank" });
```

Diferencia:

1: `IJS.InvokeAsync<object>`: Se utiliza cuando la función de JavaScript que se esta llamando devuelve un valor que se desea usar en el código C#

2: `IJS.InvokeVoidAsync`: se utiliza cuando la función JavaScript que se esta llamando no devuelve ningún valor, no espera un valor de retorno que se necesite capturar en el código C#

ERRORES (3° vez):

TaxPage.razor:
Método asincrónico public async Task ToolbarClickHandler:
→ Add → DialogVisible = true

→

```
private async Task CloseDialog()
{
DialogVisible = false; // cierra el dialogo
}
```

→ Revisar el _Host.cshtml

PurchaseOrderPage.razor:

```
<SfDialog @ref="DialogAddEditOrderLine" IsModal="true" Width="600px" ShowCloseIcon="true" @bind-Visible="@I
<DialogTemplates>
```

```
// Nueva propiedad para controlar la visibilidad del diálogo de eliminación
private bool DeleteDialogVisible { get; set; } = false;
bool DialogVisible { get; set; } = false;// Para mostrar los dailogos de añadir y editar
```

```
 DialogVisible = true;
```

```
private async Task CloseDialog()
{
DialogVisible = true;
}
```

==========

ProductPage.razor:

```
<SfDialog @ref="DialogAddEditProduct" IsModal="true" Width="500px" ShowCloseIcon="true"
Visible="@DialogVisible" VisibleChanged="OnVisibleChanged">
```

```
<SfDialog @ref="DialogDeleteProduct" IsModal="true" Width="500px" ShowCloseIcon="true"
Visible="@DialogDeleteVisible" VisibleChanged="OnDeleteVisibleChanged">
```

```
bool DialogVisible = false;
bool DialogDeleteVisible = false;
```

```
DialogVisible = true;
```

```
DialogDeleteVisible = true;
```

```
DialogVisible = false;
```

```
private async Task CloseDialog()
{
DialogVisible = false;
}
```

```
private void OnVisibleChanged(bool value)
{
DialogVisible = value;
}
```

```
private void OnDeleteVisibleChanged(bool value)
{
DialogDeleteVisible = value;
}
```

```
DialogDeleteVisible = false;
```

```
DialogDeleteVisible = false;
```

==========

PurchaseOrdesPage.razor:

```
<SfDialog @ref="DialogAddEditOrderLine" IsModal="true" Width="600px" ShowCloseIcon="true" @bind-Visible="@I
<DialogTemplates>
```

```
<SfDialog @ref="DialogDeleteOrderLine" IsModal="true" Width="500px" ShowCloseIcon="true" @bind-Visible="@D
<DialogTemplates>
```

```
// Nueva propiedad para controlar la visibilidad del diálogo de eliminación
private bool DeleteDialogVisible { get; set; } = false;
bool DialogVisible { get; set; } = false;// Para mostrar los dailogos de añadir y editar
```

```
DialogVisible = true;
```

```
DeleteDialogVisible = true;
```

```
private async Task CloseDialog()
{
```

```
DialogVisible = true;
}
```

```
public void ConfirmDeleteNo()
{
DeleteDialogVisible = false;
selectedPOLineID = 0;
}

public void ConfirmDeleteYes()
{
OrderLineDelete();
DeleteDialogVisible = false;
selectedPOLineID = 0;
}
```

==========

SupplierPage.razor:

```
<SfDialog @ref="DialogAddEditSupplier" IsModal="true" Width="500px" ShowCloseIcon="true"
Visible="@DialogVisible" VisibleChanged="OnVisibleChanged">
```

```
<SfDialog @ref="DialogDeleteSupplier" IsModal="true" Width="500px" ShowCloseIcon="true"
Visible="@DialogDeleteVisible" VisibleChanged="OnDeleteVisibleChanged">
```

```
bool DialogVisible = false;
bool DialogDeleteVisible = false;
```

```
DialogVisible = true;
```

```
DialogDeleteVisible = true;
```

```
DialogVisible = false;
```

```
private async Task CloseDialog()
{
DialogVisible = false;
}
```

```
private void OnVisibleChanged(bool value)
{
DialogVisible = value;
}

private void OnDeleteVisibleChanged(bool value)
{
DialogDeleteVisible = value;
}

public async void ConfirmDeleteNo()
{
DialogDeleteVisible = false;
```

```
SelectedSupplierId = 0;
}
```

```
DialogDeleteVisible = false;
```

==========

TaxPage.razor

```
<SfDialog @ref="DialogAddEditTax" IsModal="true" Width="500px" ShowCloseIcon="true" Visible="@DialogVisible"
```

```
<SfDialog @ref="DialogDeleteTax" IsModal="true" Width="500px" ShowCloseIcon="true" Visible="@DialogDeleteVis
```

```
bool DialogVisible = false; // para el dialogo Add, Edit de Tax Rates
bool DialogDeleteVisible = false; // para el dialogo Delete de Tax Rates
```

DialogVisible = true;

DialogDeleteVisible = true;

DialogVisible = false;

DialogDeleteVisible = false;