Lina Web Documentation

Table des matières

Ressources d'aide : 2
Organisation du projet Lina Web 3
Arborescence globale du projet sous Visual Studio 3
Arborescence des dossiers des menus 3
Disposition des menus 4
Ajout de menu / vue 4
Ajout de View component : 8
Configuration du projet : 10
Fichier program.cs 10
Fichier .csproj 11
Utilisation de l'application 13
Connexion 13
Model : 13
Vue : 13
Controller : 13
Sélection de la langue 15
Vue 15
Model 15
Controller 15
Tableau de Bord 16
Mode Visualisation (Par défaut) 16
Mode Edition 17
Suivi des Alarmes 19
Suivi des Courbes 19
Base de données 19
Suivi des Compteurs 20
Base de données 20
Model : 20
Vue : 21
Controller : 23
Suivi des Evènements 25
Base de données 25
Controller : 28
Planning des Ordres de Travail 29

Ressources d'aide :

- ASP.NET MVC

  https://www.tutorialsteacher.com/mvc

  https://docs.microsoft.com/fr-fr/aspnet/core/?view=aspnetcore-6.0

  - Création de vue : https://docs.microsoft.com/fr-fr/aspnet/core/mvc/views/overview?view=aspnetcore-6.0
  - Création de ViewComponent https://docs.microsoft.com/fr-fr/aspnet/core/mvc/views/view-components?view=aspnetcore-6.0
  - Création du vue partielle https://docs.microsoft.com/fr-fr/aspnet/core/mvc/views/partial?view=aspnetcore-6.0
  - Utilisation de SignalR https://docs.microsoft.com/fr-fr/aspnet/core/signalr/configuration?view=aspnetcore-6.0&tabs=dotnet
- Gestion de la langue (localization) https://docs.microsoft.com/fr-fr/aspnet/core/fundamentals/localization?view=aspnetcore-6.0
- DevExpress

  https://demos.devexpress.com/ASPNetMvc/

- TypeScript

  https://www.typescriptlang.org/docs/

  https://geekflare.com/fr/typescript-vs-javascript/

- Bootstrap

  https://getbootstrap.com/docs/5.2/getting-started/introduction/

- Gridstack

  https://gridstackjs.com/

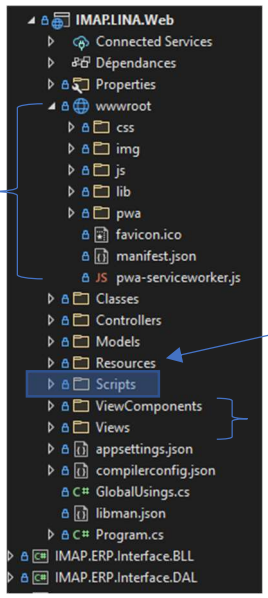
- Scss / css

  https://developer.mozilla.org/fr/docs/Web/CSS

  https://www.linkedin.com/pulse/asp-net-core-minification-et-bundling-des-fichiers-css-ravaille/

Organisation du projet Lina Web

Arborescence globale du projet sous Visual Studio

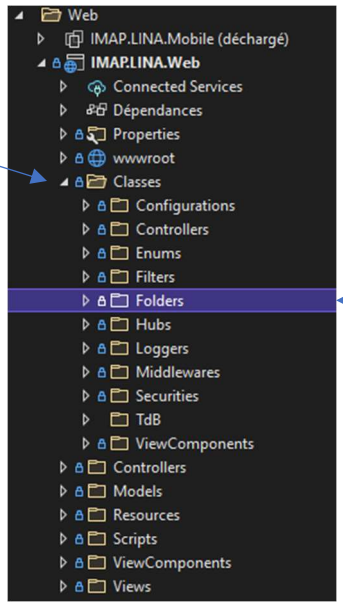


Dossier wwwroot contenant les dossiers css, images, js librairies et autres fichiers statiques du projet

Dossier Ressources (Traduction des textes en diff. Langages)

Dossier Scripts Typescript

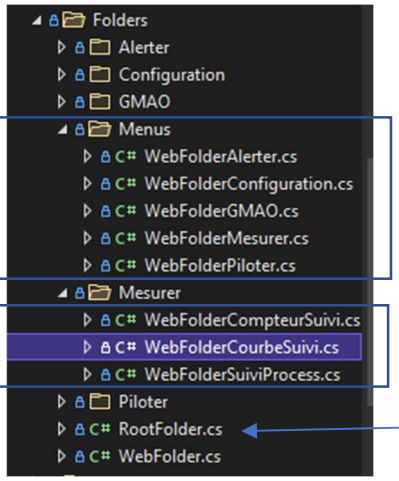
Vues / composants de vues



Dossier Classes content les classes communes à l'application LINA Web

Dossier *Folders* contenant les menus de la sidebar

Arborescence des dossiers des menus



Menus principaux

Sous-menus

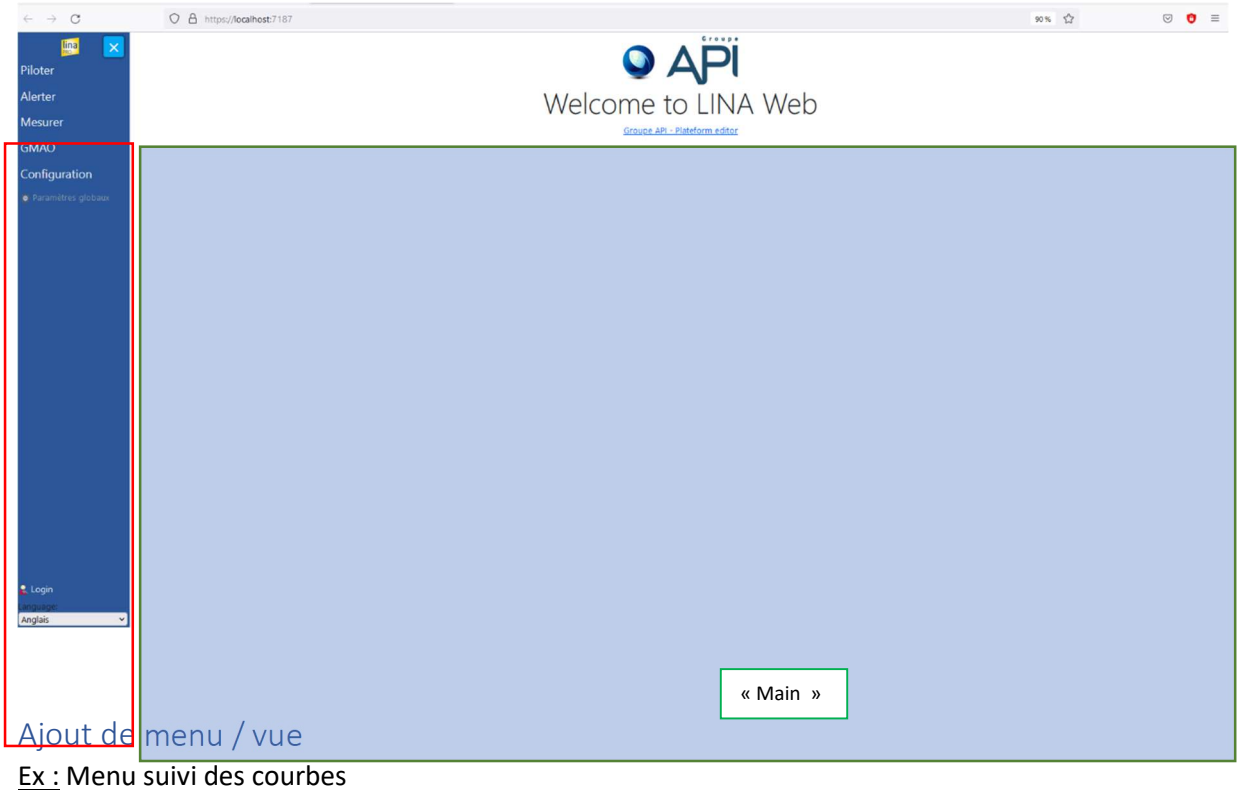
Classe de déclaration des menus principaux

Disposition des menus

L'application exécute par défaut ***_layout.cshtml*** qui contient l'équivalent de la page de démarrage ci-dessous :

- Sidebar avec menus de démarrage et Connexion/déconnexion
- Le « main » avec navbar et vue chargée (ici le home)

Nota : le home n'a pas de navbar

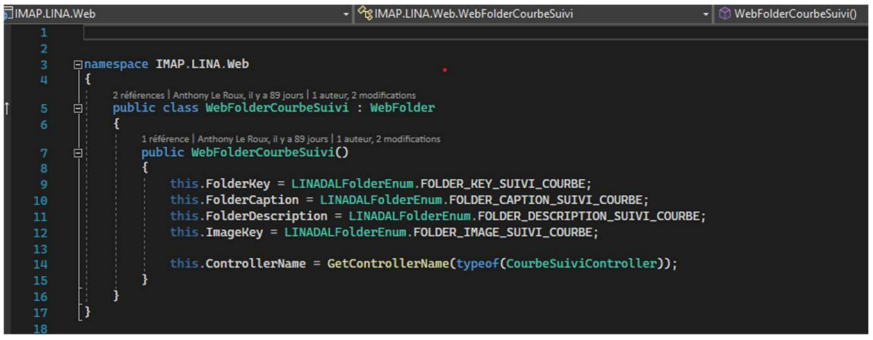


Sidebar

Ajout de menu / vue

Ex : Menu suivi des courbes

Dans le dossier mesurer, on crée **WebfolderCourbeSuivi.cs**



On y renseigne les paramètres voulus (FolderKey, FolderCaption, FolderDescription, ImageKey) ainsi que son Controller :

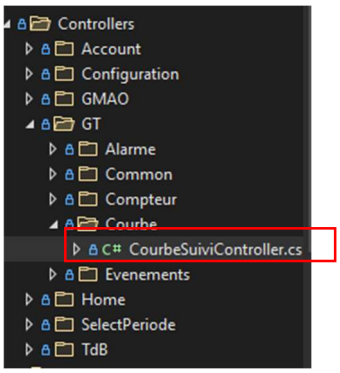
Ici, ***CourbeSuiviController***

Que l'on crée.

On prend soin de déclarer **WebfolderCourbeSuivi** dans la classe **WebFolderMesurer.cs**

```
IMAP.LINA.Web | IMAP.LINA.Web.WebFolderMesurer | WebFolderMesurer()

namespace IMAP.LINA.Web
{
    2 références | thomas.requier, il y a 58 jours | 3 auteurs, 3 modifications
    public class WebFolderMesurer : WebFolder
    {
        1 référence | thomas.requier, il y a 58 jours | 3 auteurs, 3 modifications
        public WebFolderMesurer()
        {
            this.FolderKey = LINADALFolderEnum.FOLDER_KEY_MESURER;
            this.FolderCaption = LINADALFolderEnum.FOLDER_CAPTION_MESURER;
            this.FolderDescription = LINADALFolderEnum.FOLDER_DESCRIPTION_MESURER;
            this.ImageKey = LINADALFolderEnum.FOLDER_IMAGE_MESURER;

            this.ChildsFolder.Add(new WebFolderCourbeSuivi());
            this.ChildsFolder.Add(new WebFolderCompteurSuivi());
            this.ChildsFolder.Add(new WebFolderSuiviProcess());
        }
    }
}
```

Pour notre cas, le Controller se crée ici :



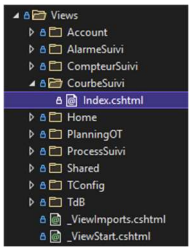
On y renseigne :

```
namespace IMAP.LINA.Web.Controllers
{
    3 références | benoit_charpentier, il y a 19 jours | 4 auteurs, 17 modifications
    public class CourbeSuiviController : LINAController
    {
        const string TEMPDATA_COURBE_SUIVI = "CourbeSuivi";

        0 références | benoit_charpentier, il y a 19 jours | 1 auteur, 1 modification
        public CourbeSuiviController(ArgumentControllerAgregator<CourbeSuiviController> pArgumentControllerAgregator) : base(pArgumentControllerAgregator)
        {

        }

        [LINAAuthorize(LINADALFolderEnum.FOLDER_KEY_SUIVI_COURBE)]
        1 référence | sebastien_lelouette, il y a 70 jours | 3 auteurs, 10 modifications
        public IActionResult Index()
        {
            SetFolderSelected();

            //Récupère le model stocké dans le tempdata
            ListArborescenceFonctSuiviModel model = GetTempData<ListArborescenceFonctSuiviModel>(TEMPDATA_COURBE_SUIVI);
            if (model == null) model = new ListArborescenceFonctSuiviModel(TypeFonction.COURBE);

            //Remplir modèle
            ArborescenceFonctSuiviDataController.InitialiserModele(ref model);

            //Afficher vue
            return View(model);
        }

        [LINAAuthorize(LINADALFolderEnum.FOLDER_KEY_SUIVI_COURBE, PermAction = nameof(CoreSecurity.FolderAction.Visualize))]
        0 références | manuel_louarn, il y a 65 jours | 1 auteur, 1 modification
        public IActionResult QRCode(string? arboID, string? periode)
        {
            ListArborescenceFonctSuiviModel m = null;
            if (arboID != null)
            {
                // Init
                m = new ListArborescenceFonctSuiviModel(TypeFonction.COURBE);

                //Config période
                if (!string.IsNullOrEmpty(periode))
                {
                    SelectionPeriode period = SelectionPeriode.Deserialize(periode.Replace(" " , "+")); //Replace(" " , "+") : "2022-05-13T00:00:00+02:00"
                    m.periode.PeriodeStart = period.StartDate;
                    m.periode.PeriodeStop = period.StopDate;
                }

                //Stocker l'arboID à pré-sélectionner dans le modèle
                m.ArboIDPreselected = Newtonsoft.Json.JsonConvert.DeserializeObject<List<int>>(arboID);
            }

            if (m != null && m.ArboIDPreselected!=null && m.ArboIDPreselected.Count >0)
            {
                //Retourne à la vue Index
                SetTempData(TEMPDATA_COURBE_SUIVI, m);
            }
            return RedirectToAction(nameof(Index));
        }
    }
```

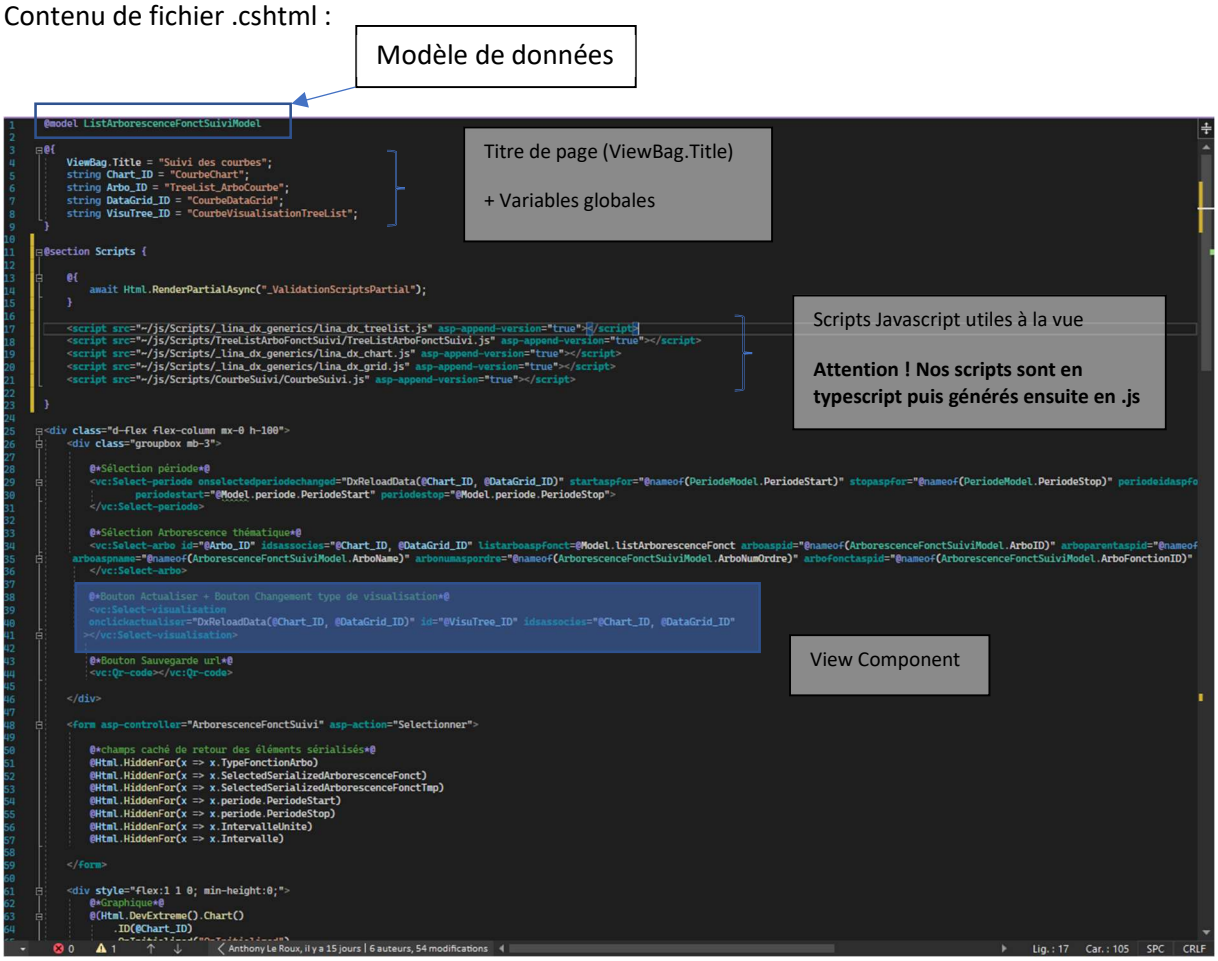
Section liée à l'affichage de la **Vue**

Section liée à l'affichage QRCode

On créée le fichier ***Index.cshtml*** dans le dossier ***CourbeSuivi***, sous-dossier de ***Views*** :

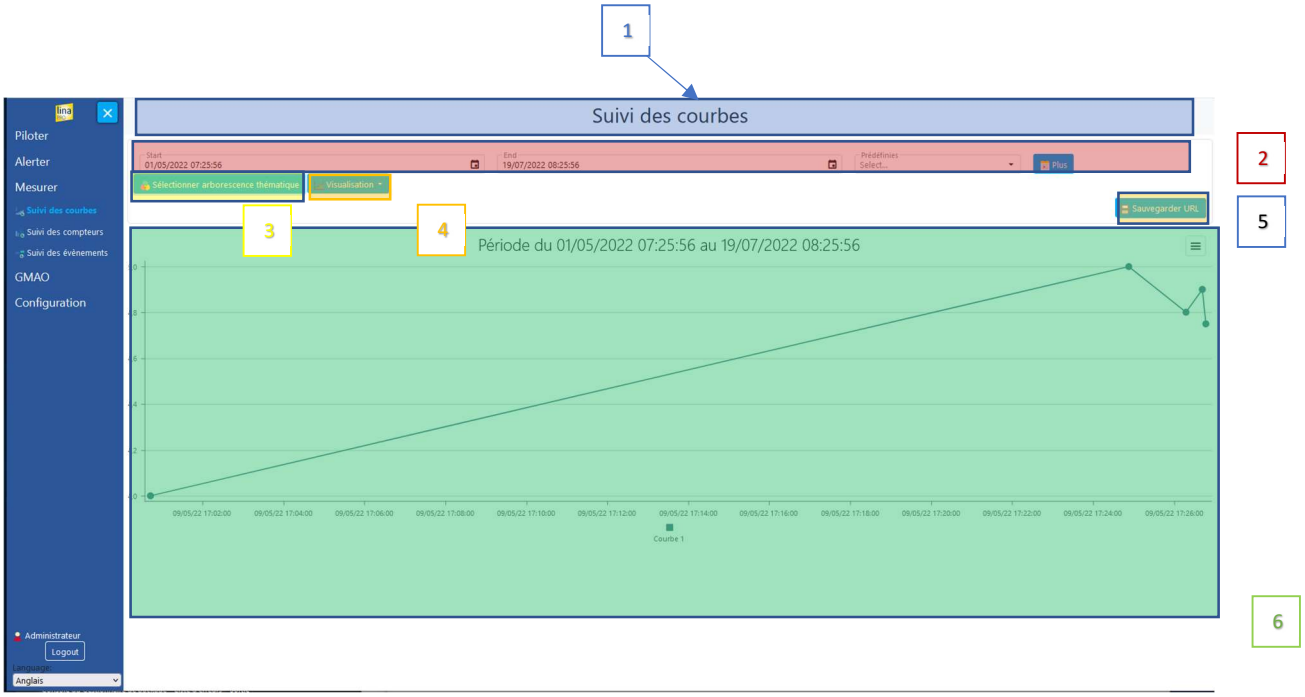


Contenu de fichier .cshtml :



Le model de cette vue est situé dans le dossier *Models/common* car il est commun à plusieurs vues.

On obtient la vue :



1 : Navbar (*ViewBag.title*)

2 : ViewComponent *SelectPeriode*

3 : ViewComponent *SelectArbo*

4 : ViewComponent *SelectVisualisation*

5 : ViewComponent *QrCode*

6 : Composant DevExtreme (DxChart ici)

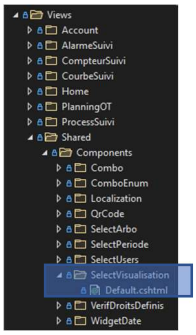
Liaison avec la Bdd :

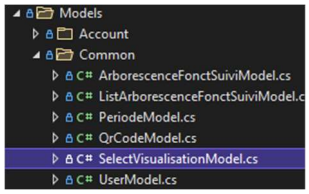
Ajout de View component :

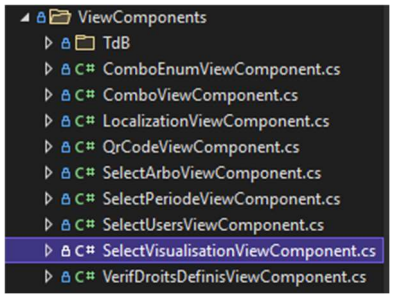
https://docs.microsoft.com/fr-fr/aspnet/core/mvc/views/view-components?view=aspnetcore-6.0

Il est composé en 3 points :

Vue (.cshtml) Modèle Controller





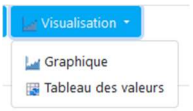


Conventions de nommage :

Supposons que vous vouliez créer un ViewComponent ***VueX***

- Le fichier de la vue devra être crée dans le dossier *Views**Shared**Components****VueX***\ Par défaut, on le nomme ***Default.cshtml***
- Le Fichier modèle devra être dans \Models\Common\ et se nommer **VueX**Model.cs.cs
- Le Controller sera à créer dans ViewComponents\ et se nommer **VueX**ViewComponent.cs

Prenons l'exemple du viewcomponent selectvisualisation : en mode web, il affiche



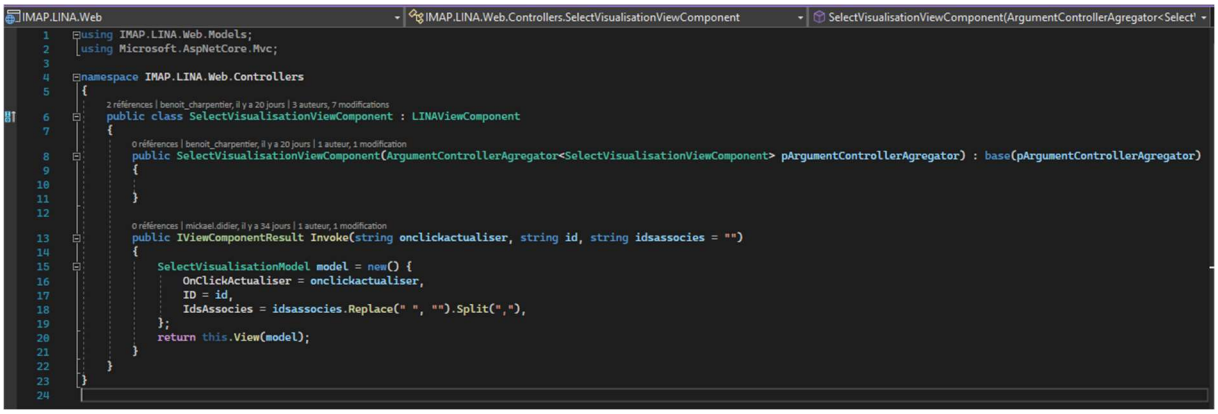
Fichier vue *Default.cshtml*:

```
@model SelectVisualisationModel

@{
    string Visualisation_ID = Model.ID;
}

<input id="@(Visualisation_ID)_ids-associes" name="@(Visualisation_ID)_ids-associes" type="hidden" value="@(JsonConvert.SerializeObject(Model.IdsAssocies))" />


<button class="btn btn-lina dropdown-toggle" type="button" id="dropdownMenuButton" data-bs-toggle="dropdown"  aria-expanded="true">
    <img src="/img/@(IMAP.ERP.Interface.DAL.LINADALImageEnum.Graphique).ico" height="16" width="16" /> Visualisation
</button>

<div class="dropdown-menu" aria-labelledby="dropdownMenuButton">
    @for (var i = 0; i < Model.IdsAssocies?.Length; i++)
    {
        string IdAssocie = Model.IdsAssocies[i].ToLower();
        string imgSrc = "";
        string text = "";
        if (IdAssocie.Contains("chart")) { imgSrc = @IMAP.ERP.Interface.DAL.LINADALImageEnum.Graphique; text = "Graphique"; }
        else if (IdAssocie.Contains("datagrid")) { imgSrc = @IMAP.ERP.Interface.DAL.LINADALImageEnum.Tableau; text = "Tableau des valeurs"; }
        else if (IdAssocie.Contains("treelist")) { imgSrc = @IMAP.ERP.Interface.DAL.LINADALImageEnum.Arborescence; text = "Arborescence"; }
        else if (IdAssocie.Contains("scheduler")) { imgSrc = @IMAP.ERP.Interface.DAL.LINADALImageEnum.Calendrier; text = "Calendrier"; }
        <a class="dropdown-item" href="#" id="action@(i)" onclick="DxShowElement(@i, '@(Visualisation_ID)_ids-associes')">
        <img src="/img/@(imgSrc).ico" height="16" width="16" alt="@(imgSrc)" />    @text</a>

    }
</div>
```

Fichier modèle *SelectVisualisationModel.cs*

```
IMAP.LINA.Web                                   IMAP.LINA.Web.Models.SelectVisualisationModel

namespace IMAP.LINA.Web.Models
{
    5 références | mickael.didier, il y a 34 jours | 2 auteurs, 4 modifications
    public class SelectVisualisationModel
    {
        2 références | mickael.didier, il y a 34 jours | 1 auteur, 1 modification
        public string ID { get; set; } = string.Empty;
        1 référence | mickael.didier, il y a 35 jours | 1 auteur, 1 modification
        public string OnClickActualiser { get; set; } = string.Empty;
        4 références | mickael.didier, il y a 34 jours | 1 auteur, 1 modification
        public string[]? IdsAssocies { get; set; } = null;
    }
}
```

Fichier Controlleur *SelectVisualisationViewComponent.cs*



```
IMAP.LINA.Web | IMAP.LINA.Web.Controllers.SelectVisualisationViewComponent | SelectVisualisationViewComponent(ArgumentControllerAgregator<Select'
1   using IMAP.LINA.Web.Models;
2   using Microsoft.AspNetCore.Mvc;
3
4   namespace IMAP.LINA.Web.Controllers
5   {
        2 références | benoit_charpentier, il y a 20 jours | 3 auteurs, 7 modifications
6       public class SelectVisualisationViewComponent : LINAViewComponent
7       {
            0 références | benoit_charpentier, il y a 20 jours | 1 auteur, 1 modification
8           public SelectVisualisationViewComponent(ArgumentControllerAgregator<SelectVisualisationViewComponent> pArgumentControllerAgregator) : base(pArgumentControllerAgregator)
9           {
10
11          }
12
            0 références | mickael.didier, il y a 34 jours | 1 auteur, 1 modification
13          public IViewComponentResult Invoke(string onclickactualiser, string id, string idsassocies = "")
14          {
15              SelectVisualisationModel model = new() {
16                  OnClickActualiser = onclickactualiser,
17                  ID = id,
18                  IdsAssocies = idsassocies.Replace(" ", "").Split(","),
19              };
20              return this.View(model);
21          }
22      }
23  }
24
```

Appel du ViewComponent :

On l'appelle en tant que Tag Helper *kebab-style*:

```
@*Bouton Actualiser + Bouton Changement type de visualisation*@
<vc:Select-visualisation
onclickactualiser="DxReloadData(@Chart_ID, @DataGrid_ID)" id="@VisuTree_ID" idsassocies="@Chart_ID, @DataGrid_ID"
></vc:Select-visualisation>
```

Configuration du projet :

Fichier program.cs

```
WebApplicationBuilder builder = WebApplication.CreateBuilder(args);

// Add services to the container.
// Ajouter les autres services ici (il n'y a pas d'ordre pour les services)
builder.Services
    .AddControllersWithViews(config =>
    {
        //Filtre d'action appliqué sur toutes les actions
        config.Filters.Add<LogPerformanceActionFilterAttribute>();
    })
    .AddViewLocalization() // Localisation
    .AddDataAnnotationsLocalization(); // Localisation

builder.Services.AddScoped(typeof(ArgumentControllerAgregator<>));

// Permet de charger la configuration appsettings.json
builder.Services.Configure<AppMapConfig>(builder.Configuration);
AppMapConfig appMapCfg = builder.Configuration.Get<AppMapConfig>();

// PWA
builder.Services.AddProgressiveWebApp(new PwaOptions()
{
    RegisterServiceWorker = false
});

// Signal R
builder.Services.AddSignalR();
//Utiliser le code ci-dessous pour envoyer un message par SignalR (application .Net Core)
//using Microsoft.AspNetCore.SignalR.Client;
//HubConnection hub = new HubConnectionBuilder().WithUrl("https://localhost:7187/LinaHub").Build();
//await hub.StartAsync();
//await hub.InvokeAsync("SendMessage", DateTime.Now.ToString());

// Internationalisation
// Localisation
builder.Services.AddLocalization(option => { option.ResourcesPath = "Resources"; });
builder.Services.Configure<RequestLocalizationOptions>(option =>
{
    option.SupportedUICultures = new[] { new CultureInfo("fr"), new CultureInfo("en"), new CultureInfo("br-FR"), new
CultureInfo("zh-CN") };
    option.DefaultRequestCulture = new Microsoft.AspNetCore.Localization.RequestCulture("fr");
});

// Chaine de connexion
builder.Services.AddSingleton<ISQLConnectionStringProvider, SQLConnectionStringProviderAppSettingsJSON>();

// Securité
builder.Services.AddHttpContextAccessor();      //Permet d'accéder au context depuis un service
builder.Services.AddSingleton<IAuthorizationHandler, LINAAuthorizationHandler>();
builder.Services.AddSingleton<ICurrentUserInfos, LINAWebCurrentUserInfos>();
builder.Services.AddAuthorization(options =>
{
    options.AddPolicy("Security", policy => policy.Requirements.Add(new LINAAuthorizationRequirement()));
});
builder.Services.AddSession();
builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
    .AddCookie(CookieAuthenticationDefaults.AuthenticationScheme, options =>
    {
        options.LoginPath = new PathString("/Account/Login");
        options.LogoutPath = new PathString("/Account/Logout");
        options.AccessDeniedPath = new PathString("/Account/AccessDenied");
        options.Cookie.Name = "Cookie_Identity";
        options.SlidingExpiration = true;
        options.ExpireTimeSpan = appMapCfg.Account.ExpireTimeSpan;
    });

// Logger
builder.Logging.ClearProviders();
builder.Logging.AddConsole();
builder.Logging.AddProvider(new LINALoggerProvider(appMapCfg));

WebApplication app = builder.Build();

// Récupère le fournisseur de services pour l'envoyer à LINA
LINADependencyInjection.ServiceProvider = app.Services;

// Configure the HTTP request pipeline.
// Ci dessous les middlewares dans l'ordre de traitement (l'ordre est important)
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-
hsts.
    app.UseHsts();
}
else
{
    //app.UseLogPerformanceRequest();     //Mesure temps d'exécution requete
}
app.UseHttpsRedirection();              //Sert à activer la redirection
app.UseStaticFiles();                   //Sert à activer l'utilisation des fichiers statiques (wwwroot)
app.UseRouting();                       //Sert à définir la route à utiliser
app.UseRequestLocalization();           //Sert à activer la localisation
app.UseSession();                       //Sert à activer le cache de session
app.UseCheckDBConnection();             //Sert à vérifier la connexion à la BDD
app.UseAuthentication();                //Sert à vérifier l'authentification
app.UseAuthorization();                 //Sert à vérifier les autorisations
app.UseEndpoints(endpoints => endpoints.MapHub<LinaHub>("/LinaHub")); //Sert à activer SignalR
app.MapControllerRoute(                 //Sert à définir les routes par défaut
    name: "default",
```

```
    pattern: "{controller=Home}/{action=Index}/{id?}");
app.Run();
```

Fichier .csproj

```
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup Label="Configuration" Condition="'$(Configuration)|$(Platform)'=='Debug|AnyCPU'">
    <TypeScriptSourceMap>False</TypeScriptSourceMap>
    <TypeScriptCompileOnSaveEnabled>True</TypeScriptCompileOnSaveEnabled>
    <TypeScriptNoImplicitAny>False</TypeScriptNoImplicitAny>
    <TypeScriptOutDir>wwwroot/js/</TypeScriptOutDir>
    <TypeScriptSourceRoot></TypeScriptSourceRoot>
    <TypeScriptRemoveComments>False</TypeScriptRemoveComments>
    <TypeScriptNoEmitOnError>True</TypeScriptNoEmitOnError>
  </PropertyGroup>
  <PropertyGroup Label="Configuration" Condition="'$(Configuration)|$(Platform)'=='Release|AnyCPU'">
    <TypeScriptSourceMap>False</TypeScriptSourceMap>
    <TypeScriptCompileOnSaveEnabled>True</TypeScriptCompileOnSaveEnabled>
    <TypeScriptNoImplicitAny>False</TypeScriptNoImplicitAny>
    <TypeScriptOutDir>wwwroot/js/</TypeScriptOutDir>
    <TypeScriptSourceRoot></TypeScriptSourceRoot>
    <TypeScriptRemoveComments>False</TypeScriptRemoveComments>
    <TypeScriptNoEmitOnError>True</TypeScriptNoEmitOnError>
  </PropertyGroup>
  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>
  <ItemGroup>
    <Compile Remove="Models\GT\Courbe\**" />
    <Compile Remove="Models\Home\**" />
    <Content Remove="Models\GT\Courbe\**" />
    <Content Remove="Models\Home\**" />
    <EmbeddedResource Remove="Models\GT\Courbe\**" />
    <EmbeddedResource Remove="Models\Home\**" />
    <None Remove="Models\GT\Courbe\**" />
    <None Remove="Models\Home\**" />
    <TypeScriptCompile Remove="wwwroot\**" />
  </ItemGroup>
  <ItemGroup>
    <Content Remove="appsettings.Development.json" />
    <Content Remove="compilerconfig.json" />
  </ItemGroup>
  <ItemGroup>
    <ProjectReference Include="..\IMAP.ERP.Interface.BLL\IMAP.ERP.Interface.BLL.csproj" />
    <ProjectReference Include="..\IMAP.ERP.Interface.DAL\IMAP.ERP.Interface.DAL.csproj" />
    <ProjectReference Include="..\IMAP.LINA.Core\IMAP.LINA.Core.csproj" />
  </ItemGroup>
  <ItemGroup>
    <Folder Include="Classes\Filters\" />
    <Folder Include="Classes\TdB\" />
    <Folder Include="wwwroot\js\Models\Common\" />
  </ItemGroup>
  <ItemGroup>
    <None Include="appsettings.Development.json" />
    <None Include="compilerconfig.json" />
  </ItemGroup>
  <ItemGroup>
    <Reference Include="System.Data.SqlClient">
      <HintPath>..\Ressources\NetCore\System.Data.SqlClient.dll</HintPath>
    </Reference>
    <PackageReference Include="DevExtreme.AspNet.Core" Version="21.2.6" />
    <PackageReference Include="DevExtreme.AspNet.Data" Version="2.8.2" />
    <PackageReference Include="Microsoft.AspNetCore.SignalR" Version="1.1.0" />
    <PackageReference Include="Microsoft.TypeScript.MSBuild" Version="4.7.4">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
    </PackageReference>
    <PackageReference Include="runtime.native.System.Data.SqlClient.sni" Version="4.7.0" />
    <PackageReference Include="WebEssentials.AspNetCore.PWA" Version="1.0.65" />
  </ItemGroup>
  <ItemGroup>
    <Compile Update="Models\Common\PeriodeModel.cs">
      <Generator>DtsGenerator</Generator>
      <LastGenOutput>PeriodeModel.cs.ts</LastGenOutput>
    </Compile>
    <Compile Update="Models\TdB\WidgetBaseModel.cs">
      <Generator>DtsGenerator</Generator>
      <LastGenOutput>WidgetBaseModel.cs.ts</LastGenOutput>
    </Compile>
    <Compile Update="Models\TdB\WidgetCourbeModel.cs">
      <Generator>DtsGenerator</Generator>
      <LastGenOutput>WidgetCourbeModel.cs.ts</LastGenOutput>
    </Compile>
    <Compile Update="Models\TdB\WidgetDateModel.cs">
      <Generator>DtsGenerator</Generator>
      <LastGenOutput>WidgetDateModel.cs.ts</LastGenOutput>
    </Compile>
    <Compile Update="Resources\CommonRessource.Designer.cs">
      <DesignTime>True</DesignTime>
      <AutoGen>True</AutoGen>
      <DependentUpon>CommonRessource.resx</DependentUpon>
    </Compile>
    <Compile Update="Resources\Views\Home\Index.Designer.cs">
      <DesignTime>True</DesignTime>
      <AutoGen>True</AutoGen>
      <DependentUpon>Index.resx</DependentUpon>
    </Compile>
    <Compile Update="Resources\Views\Shared\Components\Localization\Default.Designer.cs">
      <DesignTime>True</DesignTime>
      <AutoGen>True</AutoGen>
      <DependentUpon>Default.resx</DependentUpon>
    </Compile>
  </ItemGroup>
```

```
  <ItemGroup>
    <EmbeddedResource Update="Resources\CommonRessource.resx">
      <Generator>PublicResXFileCodeGenerator</Generator>
      <LastGenOutput>CommonRessource.Designer.cs</LastGenOutput>
    </EmbeddedResource>
    <EmbeddedResource Update="Resources\Views\Home\Index.resx">
      <Generator>PublicResXFileCodeGenerator</Generator>
      <LastGenOutput>Index.Designer.cs</LastGenOutput>
    </EmbeddedResource>
    <EmbeddedResource Update="Resources\Views\Shared\Components\Localization\Default.resx">
      <Generator>PublicResXFileCodeGenerator</Generator>
      <LastGenOutput>Default.Designer.cs</LastGenOutput>
    </EmbeddedResource>
  </ItemGroup>
  <ItemGroup>
    <None Update="Models\Common\PeriodeModel.cs.ts">
      <DesignTime>True</DesignTime>
      <AutoGen>True</AutoGen>
      <DependentUpon>PeriodeModel.cs</DependentUpon>
    </None>
  </ItemGroup>
  <ItemGroup>
    <TypeScriptCompile Update="Models\Common\PeriodeModel.cs.ts">
      <DesignTime>True</DesignTime>
      <AutoGen>True</AutoGen>
      <DependentUpon>PeriodeModel.cs</DependentUpon>
    </TypeScriptCompile>
    <TypeScriptCompile Update="Models\TdB\WidgetBaseModel.cs.ts">
      <DesignTime>True</DesignTime>
      <AutoGen>True</AutoGen>
      <DependentUpon>WidgetBaseModel.cs</DependentUpon>
    </TypeScriptCompile>
    <TypeScriptCompile Update="Models\TdB\WidgetCourbeModel.cs.ts">
      <DesignTime>True</DesignTime>
      <AutoGen>True</AutoGen>
      <DependentUpon>WidgetCourbeModel.cs</DependentUpon>
    </TypeScriptCompile>
    <TypeScriptCompile Update="Models\TdB\WidgetDateModel.cs.ts">
      <DesignTime>True</DesignTime>
      <AutoGen>True</AutoGen>
      <DependentUpon>WidgetDateModel.cs</DependentUpon>
    </TypeScriptCompile>
  </ItemGroup>
  <Target Name="PreBuild" BeforeTargets="PreBuildEvent">
    <Exec Command="xcopy &quot;$(SolutionDir)Ressources\Icones\Icones\*.*&quot; &quot;.\wwwroot\img&quot; /Y /I /E" />
  </Target>
  <ProjectExtensions><VisualStudio><UserProperties appsettings_1json__JsonSchema="" /></VisualStudio></ProjectExtensions>
</Project>
```

Utilisation de l'application

Connexion

Model :

```
namespace IMAP.LINA.Web.Models
{
    public class LoginModel : UserModel
    {

        [Required(ErrorMessage = "Le mot de passe est requis")]
        [DataType(DataType.Password)]
        [Display(Name = "Mot de passe")]
        public string Password { get; set; } = null!;

        public string ReturnUrl { get; set; } = null!;

        public LoginModel()
        {

        }

    }
}
```

Vue :

```
@model LoginModel
@inject IViewLocalizer localizer

@{
    ViewBag.Title = "Connexion";
}

<hr align="center" width=80% />
<div class="row text-center" >
    <div class="col-md-3 mx-auto" align="center">
        <form asp-action="Login">
            <div asp-validation-summary="ModelOnly" class="text-danger center"></div>
            @Html.HiddenFor(x => x.ReturnUrl)
            <div class="form-group text-center" align="center">
                <label asp-for="UserName" class="control-label text-center"></label>
                <vc:Combo selected-value=@Model.IDUser viewcomponentid="cboUser" viewcomponentname="@nameof(LoginModel.IDUser)"
displaymember=@nameof(T_User.UserName) valuemember=@nameof(T_User.IDUser) blltype=@typeof(BLLUser)
fillmethod=@nameof(BLLUser.FillByActivAndConnect) param=@(new List<bool>(){true, true}.ToArray()) nulltext="Sélectionner un
utilisateur" ></vc:Combo>
                <span asp-validation-for="IDUser" class="text-danger"></span>
            </div>

            <div class="form-group text-center" align="center">
                <label asp-for="Password" class="control-label " align="center"></label>
                <input asp-for="Password" class="form-control text-center">
                <span asp-validation-for="Password" class="text-danger"></span>
            </div>
            <br/>
            <div class="form-group">
                <input type="submit" value=@localizerCommon[CommonRessource.Connexion] class="btn btn-lina" />
            </div>
        </form>
    </div>
</div>
```

Controller :

```
namespace IMAP.LINA.Web.Controllers
{
    public class AccountController : LINAController
    {
        public AccountController(ArgumentControllerAgregator<AccountController> pargs) : base(pargs)
        {
        }

        /// <summary>
        /// Index
        /// </summary>
        /// <returns></returns>
        public IActionResult Index()
        {
            ResetFolderSelected();

            return this.RedirectToAction(nameof(Login));
        }

        /// <summary>
        /// Login
        /// </summary>
        /// <param name="ReturnUrl"></param>
        /// <returns></returns>
        public IActionResult Login(string ReturnUrl = "/")
```

```
        {
            return View(new LoginModel() { ReturnUrl = ReturnUrl });
        }

        /// <summary>
        /// Login
        /// </summary>
        /// <param name="objLoginModel"></param>
        /// <returns></returns>
        [HttpPost]
        public async Task<IActionResult> Login(LoginModel objLoginModel)
        {
            if (ModelState.IsValid)
            {
                using (BLLUser bllUser = new())
                {
                    if (!objLoginModel.IDUser.HasValue)
                    {
                        ModelState.AddModelError(nameof(objLoginModel.UserName),
this.LocalizerCommon[nameof(CommonRessource.L_utilisateur_est_requis)]);
                        return View(objLoginModel);
                    }

                    bllUser.FillByID(objLoginModel.IDUser.Value);
                    if (bllUser.DataSource.Count != 1)
                    {
                        ModelState.AddModelError(nameof(objLoginModel.UserName),
this.LocalizerCommon[nameof(CommonRessource.Utilisateur_inconnu)]);
                        return View(objLoginModel);
                    }

                    T_User rowUser = bllUser.DataSource[0];

                    if (!Security.PasswordCorrect(rowUser.IDUser, objLoginModel.Password))
                    {
                        ModelState.AddModelError(nameof(objLoginModel.Password),
this.LocalizerCommon[nameof(CommonRessource.Mot_de_passe_incorrect)]);
                        return View(objLoginModel);
                    }
                    else
                    {

                        //A claim is a statement about a subject by an issuer and
                        //represent attributes of the subject that are useful in the context of authentication and authorization
operations.
                        List<Claim> claims = new()
                        {
                            new Claim(ClaimTypes.NameIdentifier, Convert.ToString(rowUser.IDUser)),
                            new Claim(ClaimTypes.Name, rowUser.UserName)
                        };

                        //Initialize a new instance of the ClaimsIdentity with the claims and authentication scheme
                        ClaimsIdentity identity = new(claims, CookieAuthenticationDefaults.AuthenticationScheme);

                        //Initialize a new instance of the ClaimsPrincipal with ClaimsIdentity
                        ClaimsPrincipal principal = new(identity);

                        //SignInAsync is a Extension method for Sign in a principal for the specified scheme.
                        //Si IsPersistent est à true, les cookies de connexion sont conservés même si le navigateur est fermé,
seul le ExpireTimeSpan dans Program.cs est pris en compte
                        await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, principal, new
AuthenticationProperties() { IsPersistent = Settings.Account.IsPersistent });

                        return LocalRedirect(objLoginModel.ReturnUrl);
                    }
                }
            }

            return View(objLoginModel);
        }

        /// <summary>
        /// LogOut
        /// </summary>
        /// <returns></returns>
        public async Task<IActionResult> LogOut()
        {
            //SignOutAsync is Extension method for SignOut
            await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);
            //Redirect to home page
            return LocalRedirect("/");
        }

        /// <summary>
        /// AccessAllowed
        /// </summary>
        /// <param name="ReturnUrl"></param>
        /// <returns></returns>
        public IActionResult AccessAllowed(string ReturnUrl = "/")
        {
            //Redirect to home page
            return View(new LoginModel() { ReturnUrl = ReturnUrl });
        }

        /// <summary>
        /// AccessDenied
        /// </summary>
        /// <param name="ReturnUrl"></param>
        /// <returns></returns>
        public IActionResult AccessDenied(string ReturnUrl = "/")
        {
            //Redirect to home page
            return View(new LoginModel() { ReturnUrl = ReturnUrl });
        }

    }
```

```
}
```

Sélection de la langue

Vue

```
@inject IViewLocalizer localizer
@inject IOptions<RequestLocalizationOptions> RequestLocalizationOptions
@model LocalizationComponentModel


@*Label*@
<label>@localizer[nameof(IMAP.LINA.Web.Resources.Views.Shared.Components.Localization.Default.Langue)]</label>

@*Combo avec les cultures supportées*@
<select id="cboLocalization" onChange="onChange()">
    @if(@Model.SupportedUICultures != null)
    {
        foreach (CultureInfo cultureInfo in Model.SupportedUICultures)
        {
            if(Model.CurrentUICulture != null && cultureInfo.Name == Model.CurrentUICulture.Name)
            {
                <option value=@cultureInfo.Name selected>@cultureInfo.DisplayName</option>
            }
            else
            {
                <option value=@cultureInfo.Name>@cultureInfo.DisplayName</option>
            }
        }
    }

</select>

<script  type="text/javascript">

    @*Sur changement de culture*@
    function onChange(){

        @*Culture sélectionnée*@
        let cultureName = document.getElementById('cboLocalization').value;

         @*Url à appeler pour changer de culture*@
        let _url = new URL('@LINAController.TruncateControllerName(nameof(HomeController))/@nameof(HomeController.SetCulture)/' +
cultureName, window.location.origin);

        @*Si success, recharge la page*@
        $.ajax({ type: "POST",  url: _url, success:  window.location.reload() });

      }

</script>
```

Model

```
namespace IMAP.LINA.Web.Models
{
    /// <summary>
    /// Une combo
    /// </summary>
    public class LocalizationComponentModel
    {
        public IList<CultureInfo>? SupportedUICultures { get; init; }
        public CultureInfo? CurrentUICulture { get; init; }

        public LocalizationComponentModel(CultureInfo? pCurrentUICulture, IList<CultureInfo>? pSupportedUICultures )
        {
            CurrentUICulture = pCurrentUICulture;
            SupportedUICultures = pSupportedUICultures;
        }
    }


}
```

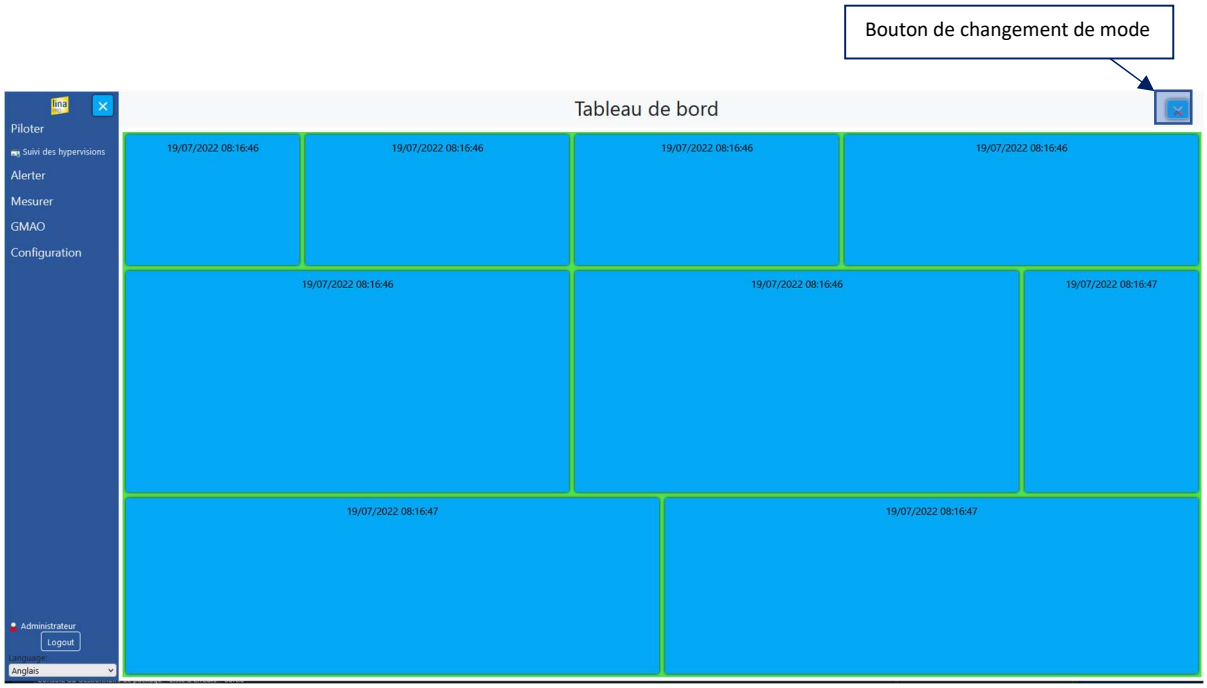
Controller

```
namespace IMAP.LINA.Web.Controllers
{
    public class LocalizationViewComponent : LINAViewComponent
    {
        private RequestLocalizationOptions mRequestLocalizationOptions { get; init; }

        public LocalizationViewComponent(ArgumentControllerAgregator<LocalizationViewComponent> pArgumentControllerAgregator,
IOptions<RequestLocalizationOptions> pRequestLocalizationOptions) : base(pArgumentControllerAgregator)
        {
            mRequestLocalizationOptions = pRequestLocalizationOptions.Value;
        }

        public IViewComponentResult Invoke()
        {

            //Culture courante
            CultureInfo? currentUICulture = null;
            IRequestCultureFeature? requestCulture = this.HttpContext.Features.Get<IRequestCultureFeature>();
            if (requestCulture != null) currentUICulture = requestCulture.RequestCulture.UICulture;

            //Model avec la liste des cultures supportées
```

```
            LocalizationComponentModel model = new LocalizationComponentModel(currentUICulture,
mRequestLocalizationOptions.SupportedUICultures);

            return this.View(model);
        }
    }
}
```

Tableau de Bord

Mode Visualisation (Par défaut)



1

Extrait de ***_layout.cshtml*** :

Code du Bouton Changement de mode :

```
@*Main*@
            <div id="main" class="d-flex flex-column col px-0 max-vh-100 justify-content-
between" style="overflow-y:auto;">
                @*Menu supérieur*@
                @if (ViewBag.Title=="Tableau de bord")
                {
                    <nav class="d-block navbar navbar-expand navbar-light bg-light d-block"
style="flex:0 1 0;">
                        <div div class="container-fluid">
                        <h1 class="container-fluid row justify-content-
center">@ViewBag.Title</h1>
                        @*Définition du bouton de choix de type de visualisation (Edition ou
Consultation)*@
                        @if (Model.Mode == WidgetDisplayMode.Display)
                        {

                            <a class="btn btn-lina m-2 visualisation-mode-change data-bs-
toggle="tooltip" data-bs-placement="left" title="Passer en Mode Edition"  asp-
controller="@WebFolder.GetControllerName(typeof(TdBController))" asp-
action="@nameof(TdBController.Design)">
                            <img
src="/img/@(IMAP.ERP.Interface.DAL.LINADALImageEnum.Outils).ico" height="16" width="16" />
                            </a>
                        }
                        else
                        {
                        <a class="btn btn-lina visualisation-mode-change data-bs-
toggle="tooltip" data-bs-placement="left" title="Passer en Mode Visualisation" asp-
controller="@WebFolder.GetControllerName(typeof(TdBController))" asp-
action="@nameof(TdBController.Index)">
                        <img
src="/img/@(IMAP.ERP.Interface.DAL.LINADALImageEnum.HypervisionStart).ico" height="16"
width="16" />
```

```
                        </a>
                         }
                        </div>
                    </nav>
                }
                else{
                @if (!string.IsNullOrEmpty(ViewBag.Title))
                {
                    <nav class="d-block navbar navbar-expand navbar-light bg-light d-block"
style="flex:0 1 0;">
                        <h1 class="text-center">@ViewBag.Title</h1>
                    </nav>
                }
                }
                @*Body*@
                <div class="d-block p-2" style="flex:1 1 0;min-height:0;">
                    @*<hr />*@
                    @RenderBody()
                </div>
            </div>
```

Mode Edition

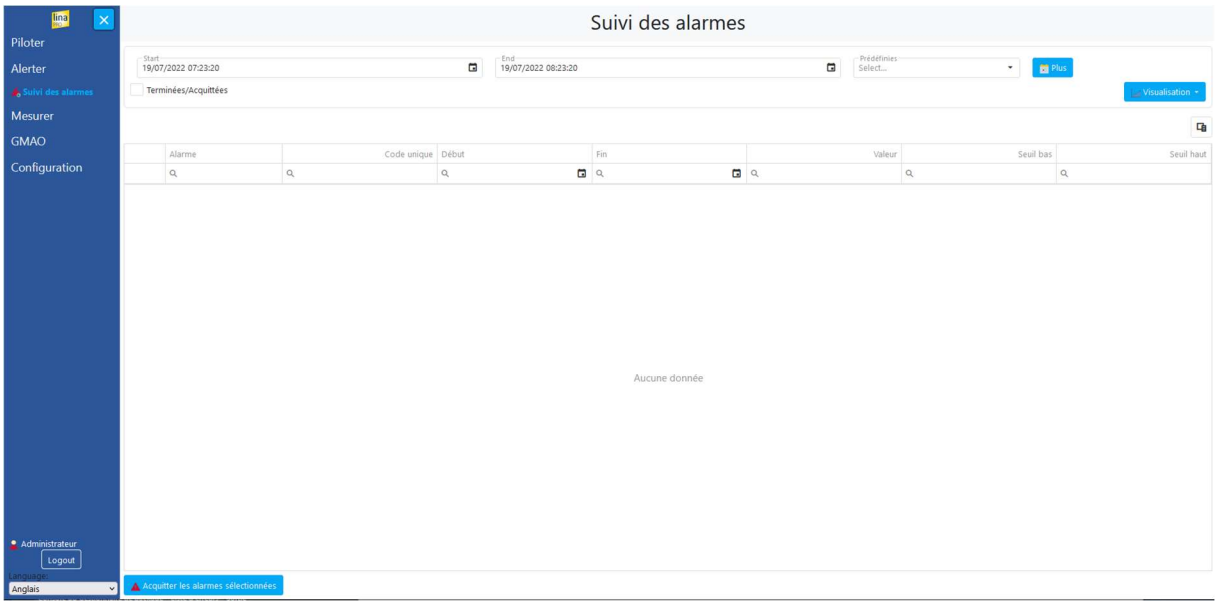
Prenons le code pour afficher la sidebar toolbox :

Sidebar Toolbox

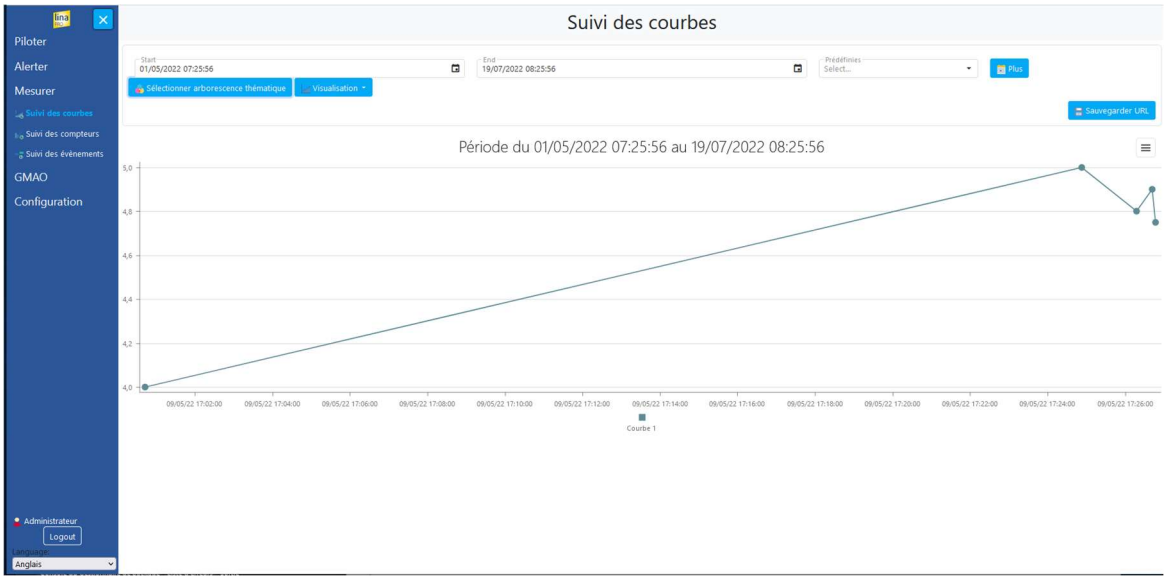
Extrait de **Index.cshtml** dans *Views\TdB*

```
<div id="tdb-main" class="d-flex flex-column mx-0 h-100 align-items-stretch" style="text-align:center">

    <div>
        @*Champs de texte cachés (Gestion du mode + couleur de fond)*@
        <input asp-for="Mode" type="hidden" id="@nameof(TdBModel.Mode)" autocomplete="off" />
        <input asp-for="CouleurFond" id="tdbCouleurFond" value="@Model.CouleurFond" autocomplete="off" type="hidden" />
    </div>

    <div class="row mx-0 justify-content-between flex-fill min-h-0">

        @* Définition de la zone allouée au Tableau de Bord*@
        <div class="col-auto flex-fill px-0" style="background-color:@(Model.CouleurFond);">
            <div class="grid-stack ui-droppable @(Model.Mode == WidgetDisplayMode.Design ? "grid-stack-grid-background grid-
stack-border ml-2": "")">

                @foreach (WidgetModel widget in Model.Widgets)
                {
                    <div id="gridstack-widget-date-@(widget.WidgetGuid)" class="newWidget grid-stack-item" @widget.GSAttrib>
                        <div class="grid-stack-item-content ui-draggable-handle btn btn-lina btn-lg py-2 px-4">

                                @switch (widget.WidgetType.ToLower())
                                {
                                    case "widget-date":
                                        <span> <vc:Widget-date mode="@Convert.ToInt32(Model.Mode)" config="@widget.WidgetParams"
/></span>
                                        break;
                                    default:
                                        break;
                                }

                        </div>
                    </div>
                }
            </div>
        </div>



        @*Barre de menus droite (mode design)*@
        @if (Model.Mode == WidgetDisplayMode.Design)
        {
            <div id="gridstack-toolbox" class="col-auto pt-2 px-0 ml-0 h-100  sidebarDashboard">

                <div id="menu" class="row mx-0 sidebarDashboard-menu w-100">

                    <div class="col-md-12 d-none d-md-block px-0" style="padding: 10px; margin-bottom:5px; margin-top:30px;">
                         <div class="col-12 px-0 sidebarDashboard-menu sidebarDashboard-toolbox w-100">
                        <label class="toolbox"><h5><b>Tableau de Bord</b></h5></label>
                    </div>
                        @* Bouton d'ouverture modal paramètres *@
                        <button data-bs-toggle="modal" data-bs-target="#tdbModal" class="btn btn-lina-toolbox mb-2">
                            <img src="~/img/Parametres.png" width="25" height="25"/> Paramètres
                        </button>
                         @*Bouton Enregister + Corbeille*@
                        <button onClick="saveGrid()" class="btn btn-lina-toolbox mb-2 py-2 px-2 data-bs-toggle="tooltip" data-bs-
placement="left" title="Sauvegarde la configuration"><img src="/img/@(IMAP.ERP.Interface.DAL.LINADALImageEnum.Enregistrer).ico"
height="16" width="16" /> Enregistrer</button>

                    </div>
                </div>

                <div class="row mx-0">
                    <div class="col-12 px-0 sidebarDashboard-menu sidebarDashboard-toolbox w-100">
                        <label class="toolbox"><h5><b>Boite à Outils</b></h5></label>
                    </div>
                </div>

                @*Définition des boutons composant la boite à outils*@
                <div id="menu" class="sideDashboard row mx-0">
                    <div class="col-md-12 d-none d-md-block px-0">
                        <button id="trash" class="btn btn-lina-danger-toolbox ui-droppable grid-stack-item-trash mb-2 py-2 px-4
data-bs-toggle="tooltip" data-bs-placement="left" title="Déplacer un widget ici pour le supprimer"><img
src="/img/@(IMAP.ERP.Interface.DAL.LINADALImageEnum.Corbeille).ico" height="16" width="16" />  Corbeille</button>
                        <div id="gridstack-widget-date" class="newWidget grid-stack-item">
                            <div class="grid-stack-item-content ui-draggable-handle btn btn-lina-toolbox mb-2 py-2 px-4">
                                <div>
                                    <img src="/img/@(IMAP.ERP.Interface.DAL.LINADALImageEnum.Heure).ico" height="32" width="32"
class="sidebarDashboard-img" /><span> <vc:Widget-date mode="1" config="" /></span>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>

            </div>
            @*Bouton de toggle:Cacher ou Afficher la toobox*@
            <button class="btn btn-lina btn-lg m-2 sidebarDashboard-toggle" onclick="popupSidebarDashBoard()"
alt="sidebarButtonToggle">
                <span></span><span></span><span></span><span></span>
            </button>
        }

    </div>
</div>
```

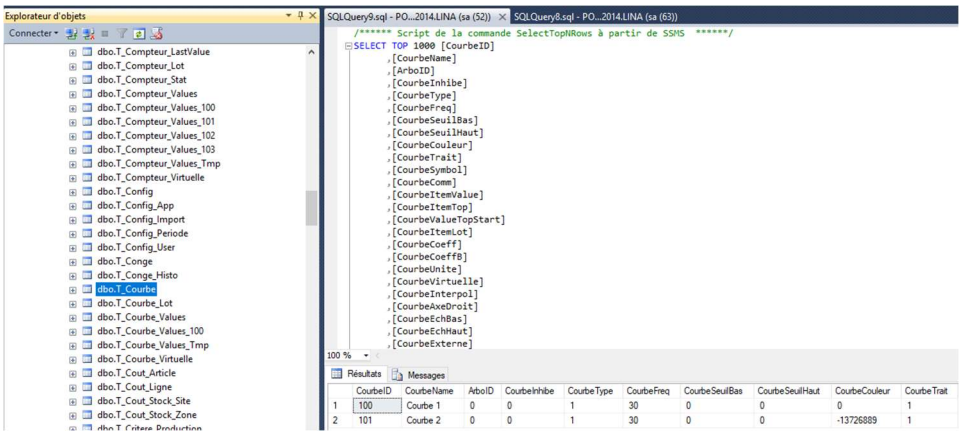
Suivi des Alarmes



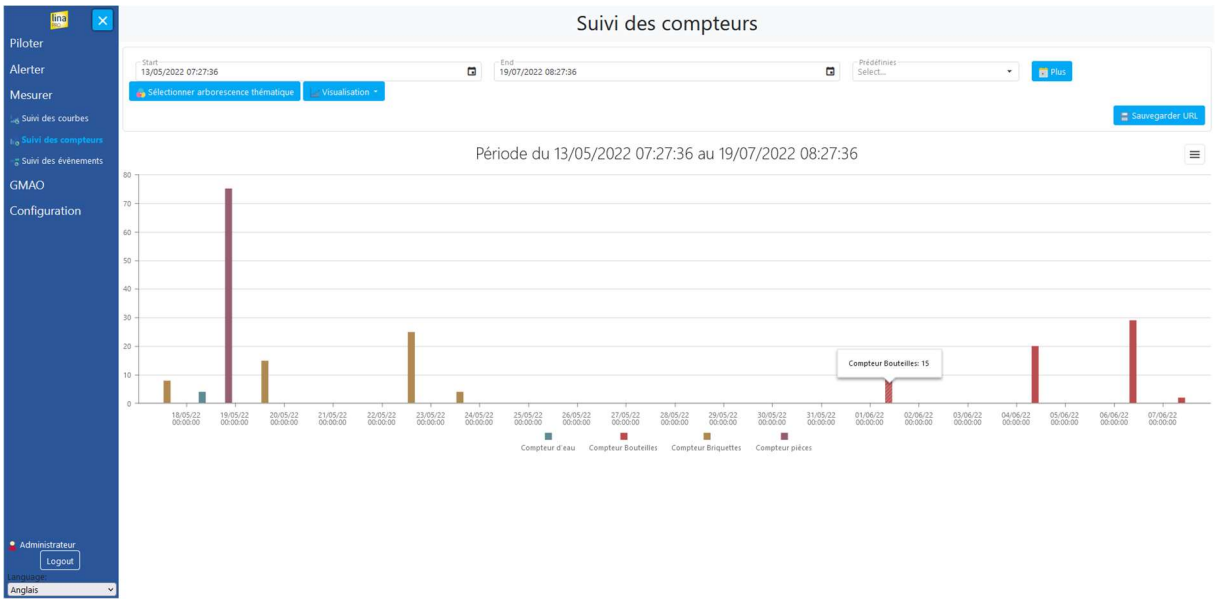
Suivi des Courbes



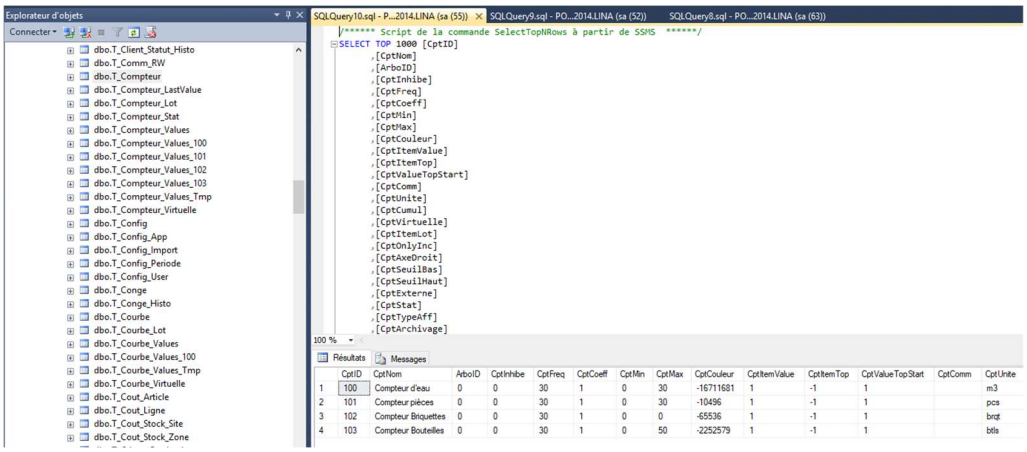
Base de données



Suivi des Compteurs



Base de données



Model :

```
public class ListArborescenceFonctSuiviModel
    {

        [Display(Name = "Période")]
        public PeriodeModel periode { get; set; } = new();

        /// <summary>
        /// Type de l'arborescence fonctionnelle à afficher
        /// </summary>
        public TypeFonction TypeFonctionArbo { get; set; } = TypeFonction.AUCUN;

        /// <summary>
        /// Infos d'intervalle pour les compteurs
        /// </summary>
        [Display(Name = "Unité intervalle")]
        public Unite_Intervalle IntervalleUnite { get; set; } = Unite_Intervalle.Heure;
        [Display(Name = "Intervalle")]
        public int Intervalle { get; set; } = 1;
        /// <summary>
        /// Permet de masquer les infos d'intervalle si on ne charge pas de compteur dans l'arborescence
        /// </summary>
        public string ContientCompteur { get { if (TypeFonctionArbo == TypeFonction.COMPTEUR || TypeFonctionArbo ==
TypeFonction.AUCUN) return ""; else return "hidden"; } }

        /// <summary>
        /// Liste des éléments de l'arborescence fonctionnelle à afficher
        /// </summary>
        public List<ArborescenceFonctSuiviModel>? listArborescenceFonct { get; set; } = null;

        /// <summary>
        /// Liste sérialisée des élements sélectionnés dans l'arborescence fonctionnelle
        /// </summary>
        public string? SelectedSerializedArborescenceFonct { get; set; } = string.Empty;

        /// <summary>
        /// Liste sérialisée temporaire des élements sélectionnés dans l'arborescence fonctionnelle
        /// </summary>
        public string? SelectedSerializedArborescenceFonctTmp { get; set; } = string.Empty;

        /// <summary>
        /// Permet de présélectionner une liste d'ArboID et ses enfants directs et indirects à l'ouverture
```

```
        /// </summary>
        public List<int>? ArboIDPreselected = null;

        /// <summary>
        /// Constructeur vide
        /// </summary>
        public ListArborescenceFonctSuiviModel()
        {
        }

        /// <summary>
        /// Constructeur avec paramètres
        /// </summary>
        /// <param name="pTypeFonctionArbo"></param>
        public ListArborescenceFonctSuiviModel(TypeFonction pTypeFonctionArbo)
        {
            TypeFonctionArbo = pTypeFonctionArbo;
        }

    }
```

Vue : ***Index.cshtml*** dans *Views\CompteurSuivi*

```
@model ListArborescenceFonctSuiviModel

@{
    ViewBag.Title = "Suivi des compteurs";
    string TreeList_ID = "TreeList_ArboCompteur";
    string Chart_ID = "CompteurChart";
    string DataGrid_ID = "CompteurDataGrid";
    string VisuTree_ID = "CompteurVisualisationTreeList";
}


<div class="d-flex flex-column mx-0 h-100">
    <div class="groupbox mb-3">

    @*Sélection période*@
        <vc:Select-periode onselectedperiodechanged="DxReloadData(@Chart_ID,@DataGrid_ID)"
startaspfor="@nameof(PeriodeModel.PeriodeStart)" stopaspfor="@nameof(PeriodeModel.PeriodeStop)"
periodeidaspfor="@nameof(PeriodeModel.PeriodeID)"
    periodestart="@Model.periode.PeriodeStart" periodestop="@Model.periode.PeriodeStop"></vc:Select-periode>

    @*Sélection Arborescence thématique*@
        <vc:Select-arbo id="@TreeList_ID" idsassocies="@Chart_ID,@DataGrid_ID" listarboaspfonct=@Model.listArborescenceFonct
arboaspid="@nameof(ArborescenceFonctSuiviModel.ArboID)" arboparentaspid="@nameof(ArborescenceFonctSuiviModel.ArboParentID)"
     arboaspname="@nameof(ArborescenceFonctSuiviModel.ArboName)"
arbonumaspordre="@nameof(ArborescenceFonctSuiviModel.ArboNumOrdre)"
arbofonctaspid="@nameof(ArborescenceFonctSuiviModel.ArboFonctionID)"
arbofonctasptype="@nameof(ArborescenceFonctSuiviModel.ArboFonctionType)"></vc:Select-arbo>

     @*Bouton Actualiser + Bouton Changement type de visualisation*@
       <vc:Select-visualisation
        onclickactualiser="DxReloadData(@Chart_ID,@DataGrid_ID)" id="@VisuTree_ID"
idsassocies="@Chart_ID,@DataGrid_ID"></vc:Select-visualisation>
     @*Bouton Sauvegarde url*@
       <vc:Qr-code></vc:Qr-code>

    </div>

    <form asp-controller="ArborescenceFonctSuivi" asp-action="Selectionner">

        @*champs caché de retour des éléments sérialisés*@
        @Html.HiddenFor(x => x.TypeFonctionArbo)
        @Html.HiddenFor(x => x.SelectedSerializedArborescenceFonct)
        @Html.HiddenFor(x => x.SelectedSerializedArborescenceFonctTmp)
        @Html.HiddenFor(x => x.periode.PeriodeStart)
        @Html.HiddenFor(x => x.periode.PeriodeStop)
        @Html.HiddenFor(x => x.IntervalleUnite)
        @Html.HiddenFor(x => x.Intervalle)

    </form>

    <div style="flex:1 1 0; min-height:0;">
        @*Affiche le Graphique en Barre des données*@
        @(Html.DevExtreme().Chart()
        .ID(@Chart_ID)
        .OnInitialized("OnInitialized")
        .Palette(VizPalette.Office)

            .CommonSeriesSettings(s => s
            .ArgumentField(nameof(FonctionGraphicDataPointModel.DateValeur))
            .ValueField(nameof(FonctionGraphicDataPointModel.Valeur))
            .Type(SeriesType.Bar)
        )

            .ArgumentAxis(a => a
            .ArgumentType(ChartDataType.DateTime)

            .Label(l => l
                .Format("dd/MM/yy HH:mm:ss")
            )
            )

        .SeriesTemplate(t => t
            .NameField(nameof(FonctionGraphicDataPointModel.SerieName))
        )

        .Export(e => e.Enabled(true))

        .Legend(l => l
            .VerticalAlignment(VerticalEdge.Bottom)
            .HorizontalAlignment(HorizontalAlignment.Center)
        )
```

```
        .Tooltip(t => t
            .Enabled(true)
            .Shared(true)
            .Container(nameof(FonctionGraphicDataPointModel.DateValeur)+nameof(FonctionGraphicDataPointModel.Valeur))
        )

        .ElementAttr("class", "chart-height-70-Pourc")

        .DataSource(d => d.Mvc()
                .Controller(LINAController.TruncateControllerName(nameof(ArborescenceFonctSuiviDataController)))
                .LoadAction(nameof(ArborescenceFonctSuiviDataController.Get))
                .OnBeforeSend("compteur_ChartBeforeSend")
                )
        )

        @*Table des valeurs*@
        @(Html.DevExtreme().DataGrid<FonctionGraphicDataPointModel>()
            .ID(@DataGrid_ID)
            .OnInitialized("OnInitialized")
            .Width("100%").Height("100%")
            .DataSource(d => d.Mvc()
                    .Controller(LINAController.TruncateControllerName(nameof(ArborescenceFonctSuiviDataController)))
                    .LoadAction(nameof(ArborescenceFonctSuiviDataController.Get))
                    .OnBeforeSend("compteur_GridBeforeSend"))
            .RemoteOperations(false)
            .AllowColumnReordering(true)
                .FilterRow(filterRow => filterRow
                .Visible(true)
                .ApplyFilter(GridApplyFilterMode.Auto)
            )
            .RowAlternationEnabled(true)
            .ShowBorders(true)
            .Paging(p => p.PageSize(50))
            .Pager(p => p
                .ShowPageSizeSelector(true)
                .AllowedPageSizes(new[] { 10, 25, 50, 100 })
            )
            .SearchPanel(s => s
                .Visible(true)
                .HighlightCaseSensitive(true)
            )
            .AllowColumnResizing(false)
            .ColumnMinWidth(250)
            .ColumnAutoWidth(true)
            .Columns(columns => {
                columns.AddFor(m => m.DateValeur)
                    .Width(100)
                    .Alignment(HorizontalAlignment.Center)
                    .Format("dd/MM/yy HH:mm:ss");
                columns.AddFor(m => m.Valeur)
                    .Width(120)
                    .Alignment(HorizontalAlignment.Center)
                    .Format("#.00");
                columns.AddFor(m => m.SerieName)
                .GroupIndex(0)
                .Width(200)
                .Alignment(HorizontalAlignment.Left);
                    })
            .Grouping(grouping => grouping.AutoExpandAll(true))
            .GroupPanel(groupPanel => groupPanel.Visible(true))
        )
    </div>
</div>


@section Scripts {

    @{
    await Html.RenderPartialAsync("_ValidationScriptsPartial");
    }

    <script src="~/js/Scripts/_lina_dx_generics/lina_dx_treelist.js" asp-append-version="true"></script>
    <script src="~/js/Scripts/TreeListArboFonctSuivi/TreeListArboFonctSuivi.js" asp-append-version="true"></script>
    <script src="~/js/Scripts/_lina_dx_generics/lina_dx_chart.js" asp-append-version="true"></script>
    <script src="~/js/Scripts/_lina_dx_generics/lina_dx_grid.js" asp-append-version="true"></script>
    <script src="~/js/Scripts/CompteurSuivi/CompteurSuivi.js" asp-append-version="true"></script>

}
```

Controller : **CompteurSuiviController.cs**

```
namespace IMAP.LINA.Web.Controllers
{
    public class CompteurSuiviController : LINAController
    {
        const string TEMPDATA_COMPTEUR_SUIVI = "CompteurSuivi";

        public CompteurSuiviController(ArgumentControllerAgregator<CompteurSuiviController> pArgumentControllerAgregator) :
base(pArgumentControllerAgregator)
        {

        }

        [LINAAuthorize(LINADALFolderEnum.FOLDER_KEY_SUIVI_COMPTEUR)]
        public IActionResult Index()
        {
            SetFolderSelected();

            //Récupére le model stocké dans le tempdata
            ListArborescenceFonctSuiviModel model = GetTempData<ListArborescenceFonctSuiviModel>(TEMPDATA_COMPTEUR_SUIVI);
            if (model == null) model = new ListArborescenceFonctSuiviModel(TypeFonction.COMPTEUR);

            //Remplir modèle
            ArborescenceFonctSuiviDataController.InitialiserModele(ref model);

            //Afficher vue
            return View(model);
        }

        [LINAAuthorize(LINADALFolderEnum.FOLDER_KEY_SUIVI_COMPTEUR, PermAction = nameof(CoreSecurity.FolderAction.Visualize))]
        public IActionResult QRCode(string? arboID, string? periode)
        {
            ListArborescenceFonctSuiviModel m = null;
            if (arboID != null)
            {
                // Init
                m = new ListArborescenceFonctSuiviModel(TypeFonction.COMPTEUR);

                //Config période
                if (!string.IsNullOrEmpty(periode))
                {
                    SelectionPeriode period = SelectionPeriode.Deserialize(periode.Replace(" ", "+")); //Replace(" " , "+") :
"2022-05-13T00:00:00+02:00" est automatiquement remplacé par "2022-05-13T00:00:00 02:00" (manque le '+')
                    m.periode.PeriodeStart = period.StartDate;
                    m.periode.PeriodeStop = period.StopDate;
                }

                //Stocker l'arboID à pré-sélectionner dans le modèle
                m.ArboIDPreselected = Newtonsoft.Json.JsonConvert.DeserializeObject<List<int>>(arboID);
            }

            if (m != null && m.ArboIDPreselected != null && m.ArboIDPreselected.Count > 0)
            {
                //Retourne à la vue Index
                SetTempData(TEMPDATA_COMPTEUR_SUIVI, m);
            }
            return RedirectToAction(nameof(Index));
        }

    }
}
```

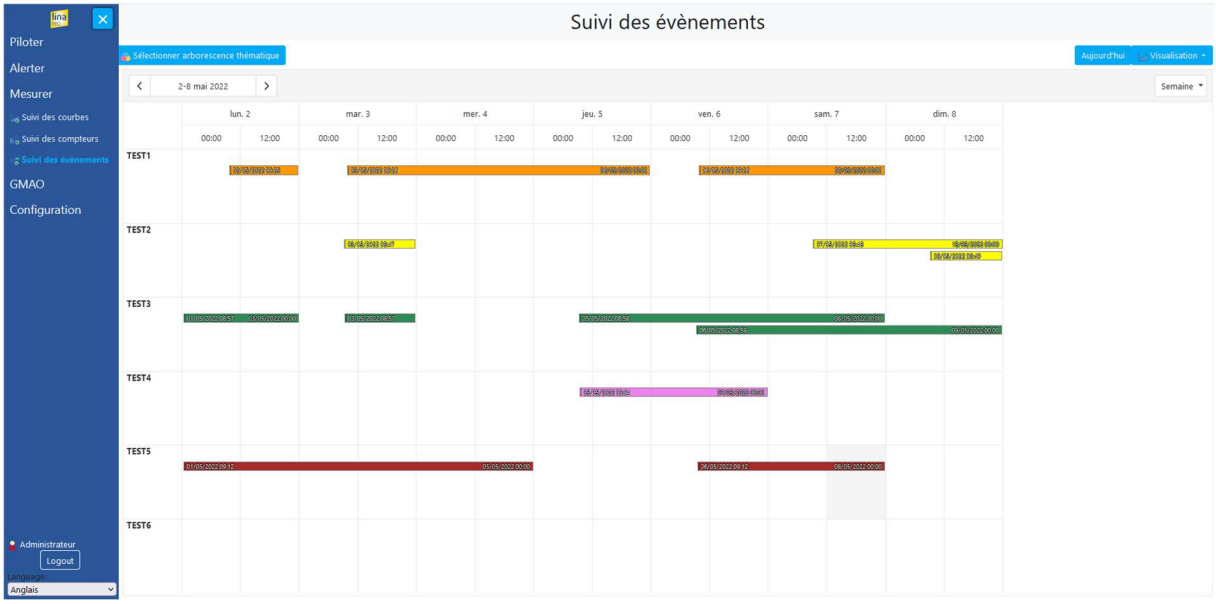
Appel de **ArborescenceFonctSuiviDataController** Méthode *InitialiserModele*

```
/// <summary>
        /// Initialiser l'arborescence du modèle pModel
        /// </summary>
        /// <param name="pModel"></param>
        /// <returns></returns>
        public static bool InitialiserModele(ref ListArborescenceFonctSuiviModel pModel)
        {
            using (LBLLNavArboFonct bllArboFonct = new LBLLNavArboFonct())
            {
                if (pModel.TypeFonctionArbo == TypeFonction.AUCUN) bllArboFonct.FillForMultiFonctions();
                else bllArboFonct.FillByFonctionTypeWithExternData(pModel.TypeFonctionArbo);

                //Gestion droits
                ArboSecurity.ApplyArboSecurityOnArboFonct(bllArboFonct.DataSource);

                //Supprime dossiers vides
                ArboSecurity.DeleteEmptyDirectoriesFonct(bllArboFonct.DataSource, 0);

                pModel.listArborescenceFonct = (from x in bllArboFonct.DataSource select new
ArborescenceFonctSuiviModel(x)).ToList();
                bllArboFonct.DataSource.Clear();
            }

            //S'il existe une pré-sélection à faire
            if (pModel.ArboIDPreselected != null && pModel.ArboIDPreselected.Count > 0)
            {
                //Récupérer liste des enfants directs et indirects de model.ArboIDPreselected dans model.listArborescenceFonct et
la sérialiser dans SelectedSerializedArborescenceFonct
                List<ArborescenceFonctSuiviModel> listPreselection = new List<ArborescenceFonctSuiviModel>();
                foreach (int arboIDPreSelect in pModel.ArboIDPreselected)
                {
                    GetListEnfantDirectEtIndirect(pModel.listArborescenceFonct, arboIDPreSelect, ref listPreselection);
                }
                if (listPreselection.Count > 0) pModel.SelectedSerializedArborescenceFonct =
Newtonsoft.Json.JsonConvert.SerializeObject(listPreselection);
            }

            return true;}
```

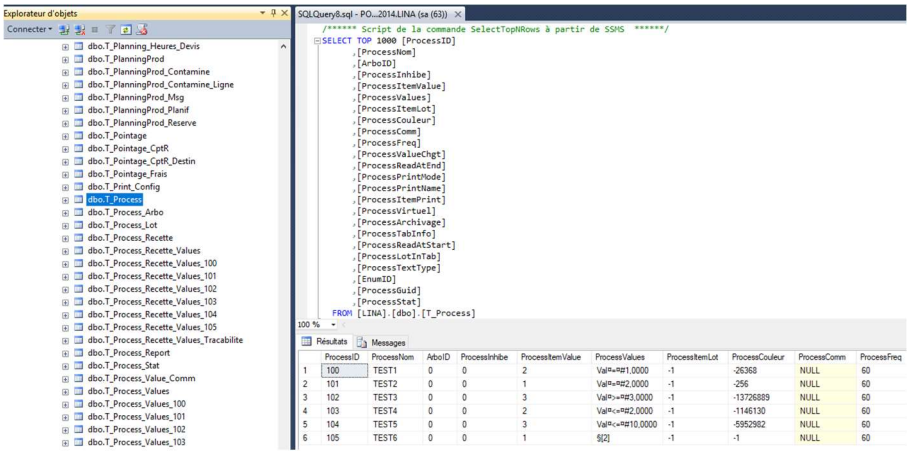
Exemple de fonctions typescript : ***CompteurSuivi.ts***

```
//*************************************************************************************************************************************
//**** CompteurSuivi
//**
//** Contient toutes les fonctions spécifiques au suivi des compteurs
//*************************************************************************************************************************************

/// <reference path="../_lina_dx_generics/lina_dx_chart.ts"/>
/// <reference path="../TreeListArboFonctSuivi/TreeListArboFonctSuivi.ts"/>

/** Avant d'envoyer la requête Get au webservice */
function compteur_ChartBeforeSend(operation, ajaxSettings) { DxGetVar("CompteurChart")?.OnBeforeSend(operation, ajaxSettings); }
/** Avant d'envoyer la requête Get au webservice */
function compteur_GridBeforeSend(operation, ajaxSettings) { DxGetVar("CompteurDataGrid")?.OnBeforeSend(operation, ajaxSettings);
}

// Inutilisé ?
function set_seriestype(type: boolean) {
    if (type = true) {
        this.Instance.option({set_seriestype:"bar"})
    }

    else {
        this.Instance.option({set_seriestype:"line"})
    }
}

//*************************************************************************************************************************************
/** TreeList_ArboCompteur
 *
 * Classe spécifique à la liste des compteurs à sélectionner, héritant de la classe TreeListArborescenceFonctSuiviModel
 */
class TreeList_ArboCompteur extends TreeListArborescenceFonctSuiviModel {  }

//*************************************************************************************************************************************
/** CompteurChart
 *
 * Classe spécifique au graphique des compteurs, héritant de la classe générique DxChart
 */
class CompteurChart extends DxChart {

    /** Avant d'envoyer la requête Get au webservice */
    public OnBeforeSend(operation, ajaxSettings) {
        const DxStart = DxGetInstance("PeriodeStart", "datebox");
        const DxStop = DxGetInstance("PeriodeStop", "datebox");
        this.Instance.option("title", "Période du " + DxStart.option("text") + " au " + DxStop.option("text"));

        //DateBox période
        ajaxSettings.data.PeriodeStartMs = GetDateTime(GetDateBoxValue("PeriodeStart"))
        ajaxSettings.data.PeriodeStopMs = GetDateTime(GetDateBoxValue("PeriodeStop"))
        ajaxSettings.data.PeriodeID = GetSelectBoxValue("PeriodeID");
        ajaxSettings.data.Intervalle = 1;
        ajaxSettings.data.IntervalleUnite = 2;

        //Liste sérialisée des arbo fonctionnel
        ajaxSettings.data.SelectedSerializedArborescenceFonct =
(<HTMLInputElement>document.getElementById("SelectedSerializedArborescenceFonct")).value;
    }

}

//*************************************************************************************************************************************
/** CompteurDataGrid
 *
 * Classe spécifique à la grille des compteurs, héritant de la classe générique DxDataGrid
 */
class CompteurDataGrid extends DxDataGrid {

    //Avant appel du WebServices de la grille
    public OnBeforeSend(operation, ajaxSettings) {

        //DateBox période
        ajaxSettings.data.PeriodeStartMs = GetDateTime(GetDateBoxValue("PeriodeStart"))
        ajaxSettings.data.PeriodeStopMs = GetDateTime(GetDateBoxValue("PeriodeStop"))
        ajaxSettings.data.PeriodeID = GetSelectBoxValue("PeriodeID");
        ajaxSettings.data.Intervalle = 1;
        ajaxSettings.data.IntervalleUnite = 2;

        //Liste sérialisée des arbo fonctionnel
        ajaxSettings.data.SelectedSerializedArborescenceFonct =
(<HTMLInputElement>document.getElementById("SelectedSerializedArborescenceFonct")).value;
    }

}
```

Suivi des Evènements



Base de données



Model : ***ProcessSuiviModel*** dans *Models\GT\Process*

```
namespace IMAP.LINA.Web.Models
{

    public class ProcessSuiviModel
    {
        public ListArborescenceFonctSuiviModel? ListArboModel { get; set; } = null!;
    }

    public class ProcessModel
    {
        public int ProcessID { get; init; }
        public string ProcessNom { get; init; } = string.Empty;
        public string ProcessCouleur { get { return
$"#{ProcessCouleur_Color.R:X2}{ProcessCouleur_Color.G:X2}{ProcessCouleur_Color.B:X2}"; }}
        public string ProcessComm { get; init; } = string.Empty;
        public System.Drawing.Color ProcessCouleur_Color { get; init; }

        public ProcessModel(T_Process pRowProcess)
        {
            ProcessID = pRowProcess.ProcessID;
            ProcessNom = pRowProcess.ProcessNom;
            ProcessCouleur_Color = pRowProcess.ProcessCouleur_Color;
            ProcessComm = pRowProcess.ProcessComm;
        }
    }

    public class ProcessValuesModel
    {
        public string ValueIDInView { get { return $"{this.ProcessID}-{this.ValueID}"; } }
        public long ValueID { get; init; }
        public int ProcessID { get; init; }
        public string ProcessNom { get; init; } = string.Empty;
        public string Value { get; init; } = string.Empty;
        public DateTime? DateStart { get; init; } = null!;
        public DateTime? DateStop { get; init; } = null!;
        public string ProcessComm { get; init; } = string.Empty;
        public TimeSpan? DateDuree { get { return this.DateStop - this.DateStart; } }
        public double DateDureeSec { get; init; } = 0;

        public ProcessValuesModel(T_Process_Values pRowProcessValues)
        {
```

```
            ValueID = pRowProcessValues.ValueID;
            ProcessID = pRowProcessValues.ProcessID;
            Value = pRowProcessValues.Value.ToString();
            DateStart = pRowProcessValues.ValueStart;
            DateStop = pRowProcessValues.ValueStop;
            ProcessNom = pRowProcessValues.ProcessNom;
            ProcessComm = pRowProcessValues.ProcessComm;
            DateDureeSec = pRowProcessValues.DureeSecNotNULL;
        }
    }

    public class ProcessRecetteValuesModel
    {
        public string ProcessRecNom { get; init; } = string.Empty;
        public string RecValueDisplay { get; init; } = string.Empty;

        public ProcessRecetteValuesModel(T_Process_Recette_Values pRowProcessRecetteValues)
        {
            ProcessRecNom = pRowProcessRecetteValues.ProcessRecNom;
            RecValueDisplay = pRowProcessRecetteValues.RecValueDisplay;
        }
    }

    public class ProcessValueCommModel
    {
        public string CommTexte { get; init; } = string.Empty;
        public DateTime CommDate { get; init; }
        public int CommID { get; init; }

        public ProcessValueCommModel(T_Process_Value_Comm pRowProcessValueComm)
        {
            CommTexte = pRowProcessValueComm.CommTexte;
            CommDate = pRowProcessValueComm.CommDate;
            CommID = pRowProcessValueComm.CommID;

        }
    }
}
```

Vue : **Index.cshtml** dans *\Views\ProcessSuivi*

```
@model ProcessSuiviModel

@{
    ViewBag.Title = "Suivi des évènements";
    string Scheduler_ID = "ProcessScheduler";
    string DataGrid_ID = "ProcessDataGrid";
    string ArboTreeList_ID = "ProcessSchedulerTreeList";
    string VisuTreeList_ID = "ProcessVisualisationTreeList";
}


<partial name="_SchedulerPopup.cshtml" model="@Scheduler_ID" />

@* Configuration des Views (CellDuration = minutes de chaque colonne, impossible de dépasser 1440 (1jour) ?) *@
@{
    void InitViews (CollectionFactory<SchedulerViewBuilder> configurator)
    {
        configurator.Add()
            .Name("Heure")
            .Type(SchedulerViewType.TimelineDay)
            .CellDuration(15)
            .IntervalCount(1);

        configurator.Add()
            .Name("Jour")
            .Type(SchedulerViewType.TimelineDay)
            .GroupOrientation(Orientation.Vertical)
            .CellDuration(60)
            .IntervalCount(1);

        configurator.Add()
            .Name("Semaine")
            .Type(SchedulerViewType.TimelineWeek)
            .CellDuration(720)
            .IntervalCount(1);

        configurator.Add()
            .Name("Mois")
            .Type(SchedulerViewType.TimelineMonth)
            .CellDuration(1440);
    }
}




    <div class="d-flex flex-column mx-0 h-100 align-items-stretch">
        <div class="mb-2 d-flex flex-wrap justify-content-end align-items-end" style="flex:0 1 auto;">
            <div style="flex:1 1 0; margin-left:-15px;">

                @*Sélection Arborescence thématique*@
                <vc:Select-arbo
                    id="@ArboTreeList_ID"
                    idsassocies="@Scheduler_ID,@DataGrid_ID"
                    listarboaspfonct=@Model.ListArboModel?.listArborescenceFonct
                    arboaspid="@nameof(ArborescenceFonctSuiviModel.ArboID)"
                    arboparentaspid="@nameof(ArborescenceFonctSuiviModel.ArboParentID)"
                    arboaspname="@nameof(ArborescenceFonctSuiviModel.ArboName)"
                    arbonumaspordre="@nameof(ArborescenceFonctSuiviModel.ArboNumOrdre)"
                    arbofonctaspid="@nameof(ArborescenceFonctSuiviModel.ArboFonctionID)"
                    arbofonctasptype="@nameof(ArborescenceFonctSuiviModel.ArboFonctionType)"
                    arboaspshowid="true"
                ></vc:Select-arbo>

            </div>
```

```
            <div class="ml-2" style="flex:0 1 auto;">
                <button class="btn btn-lina" onclick="Scheduler_ScrollToToday(@Scheduler_ID)">Aujourd'hui</button>
            </div>
            @*Bouton Actualiser + Bouton Changement type de visualisation*@
            <vc:Select-visualisation
                onclickactualiser="DxReloadData(@Scheduler_ID, @DataGrid_ID)" id="@VisuTreeList_ID" idsassocies="@Scheduler_ID,
@DataGrid_ID">
            </vc:Select-visualisation>
        </div>
        <div style="flex:1 1 0; min-height:0;">
            @* Process Scheduler *@
            @(Html.DevExtreme().Scheduler()
            .ID(@Scheduler_ID)
            .OnInitialized("OnInitialized")
            .DataSource(ds => ds.Mvc()
                .Controller(LINAController.TruncateControllerName(nameof(ProcessSuiviController)))
                .UpdateAction(nameof(ProcessSuiviController.GetProcessValues))
                .LoadAction(nameof(ProcessSuiviController.GetProcessValues))
                .OnBeforeSend("processScheduler_beforeSend")
            )
            .MaxAppointmentsPerCell(5)
            .UseDropDownViewSwitcher(true)
            .StartDateExpr(nameof(ProcessValuesModel.DateStart))
            .EndDateExpr(nameof(ProcessValuesModel.DateStop))
            .TextExpr(nameof(ProcessValuesModel.ProcessNom))
            .Views(configurator => InitViews(configurator))
            .CurrentView(SchedulerViewType.TimelineWeek)
            .Groups(new[] { nameof(ProcessValuesModel.ProcessID) })

            .Resources(r => {
                r.Add()
                    .DataSource(ds => ds.Mvc()
                        .Controller(LINAController.TruncateControllerName(nameof(ProcessSuiviController)))
                        .UpdateAction(nameof(ProcessSuiviController.GetProcessValues))
                        .LoadAction(nameof(ProcessSuiviController.GetProcessValues))
                        .OnBeforeSend("processScheduler_beforeSendProcessValues")
                        .LoadMode(DataSourceLoadMode.Raw)
                    )
                    .FieldExpr(nameof(ProcessValuesModel.ValueID))
                    .AllowMultiple(false)
                    .ValueExpr(nameof(ProcessValuesModel.ValueIDInView))
                    .DisplayExpr(nameof(ProcessValuesModel.Value))
                    .Label("Valeur");
                r.Add()
                    .DataSource(ds => ds.Mvc()
                        .Controller(LINAController.TruncateControllerName(nameof(ProcessSuiviController)))
                        .LoadAction(nameof(ProcessSuiviController.GetProcess))
                        .OnBeforeSend("processScheduler_beforeSendProcess")
                    )
                    .FieldExpr(nameof(ProcessModel.ProcessID))
                    .AllowMultiple(false)
                    .Label("Événement")
                    .ValueExpr(nameof(ProcessModel.ProcessID))
                    .DisplayExpr(nameof(ProcessModel.ProcessNom))
                    .ColorExpr(nameof(ProcessModel.ProcessCouleur))
                    .UseColorAsDefault(true);
                }
            )
            .Editing(false)
            .Height("100%")
            .Width("100%")
            .ShowAllDayPanel(false)
            .Scrolling(config => { config.Mode(SchedulerScrollingMode.Virtual); })
            .CrossScrollingEnabled(true)
            @* Template pour afficher la date de début et de fin de chaque appointment *@
            .AppointmentTemplate(
                @<text>
                    <div class="position-absolute font-sm font-weight-bold d-block" style="text-overflow:''; max-
width:90%;"></div>
                    <div class="position-absolute font-sm font-weight-bold d-block" style="text-overflow:''; max-
width:90%;"></div>
                </text>)
            )

            @* Process Data Grid *@
            @(Html.DevExtreme().DataGrid<ProcessValuesModel>()
            .ID(@DataGrid_ID).OnInitialized("OnInitialized")
            .DataSource(ds => ds.Mvc().Controller(LINAController.TruncateControllerName(nameof(ProcessSuiviController)))
                .LoadAction(nameof(ProcessSuiviController.GetProcessValues))
                .OnBeforeSend("processDataGrid_beforeSend")
            )
            .Width("100%").Height("100%")
            .RowAlternationEnabled(true)
            .ShowBorders(true)
            .ColumnAutoWidth(true)
            .Paging(paging => paging.Enabled(false))
            .Columns(columns => {

                columns.AddFor(m => m.ProcessNom).Caption("Événement").HidingPriority(6);
                columns.AddFor(m => m.DateStart).Caption("Début").HidingPriority(3);
                columns.AddFor(m => m.DateStop).Caption("Fin").HidingPriority(4);
                columns.AddFor(m => m.DateDuree).Caption("Durée").HidingPriority(2);
                columns.AddFor(m => m.DateDureeSec).Caption("Durée (sec)").HidingPriority(1);
                columns.AddFor(m => m.Value).Caption("Valeur").HidingPriority(5);

            })
            )

        </div>
    </div>


<form asp-controller="ArborescenceFonctSuivi" asp-action="Selectionner">
    @*champs caché de retour des éléments sérialisés*@
    @Html.Hidden("TypeFonctionArbo", Model.ListArboModel.TypeFonctionArbo)
    @Html.Hidden("SelectedSerializedArborescenceFonct", Model.ListArboModel.SelectedSerializedArborescenceFonct)
```

```
    @Html.Hidden("SelectedSerializedArborescenceFonctTmp", Model.ListArboModel.SelectedSerializedArborescenceFonctTmp)
</form>

@section Scripts {
    @{
        await Html.RenderPartialAsync("_ValidationScriptsPartial");
    }

    <script src="~/js/Scripts/_lina_dx_generics/lina_dx_treelist.js" asp-append-version="true"></script>
    <script src="~/js/Scripts/_lina_dx_generics/lina_dx_scheduler.js" asp-append-version="true"></script>
    <script src="~/js/Scripts/_lina_dx_generics/lina_dx_grid.js" asp-append-version="true"></script>
    <script src="~/js/Scripts/ProcessSuivi/ProcessSuivi.js" asp-append-version="true"></script>
}
```

Controller :

```
namespace IMAP.LINA.Web.Controllers
{

    public class ProcessSuiviController : LINAController
    {

        const string TEMPDATA_PROCESS_SUIVI = "ProcessSuivi";
        public ProcessSuiviController(ArgumentControllerAgregator<ProcessSuiviController> pArgumentControllerAgregator) :
base(pArgumentControllerAgregator)
        {

        }

        [LINAAuthorize(LINADALFolderEnum.FOLDER_KEY_PROCESS)]
        public IActionResult Index()
        {
            SetFolderSelected();

            //Récupére le model stocké dans le tempdata
            ProcessSuiviModel model = GetTempData<ProcessSuiviModel>(TEMPDATA_PROCESS_SUIVI) ?? new ProcessSuiviModel();

            //Récupére le model stocké dans le tempdata
            ListArborescenceFonctSuiviModel ListArboModel = GetTempData<ListArborescenceFonctSuiviModel>(TEMPDATA_PROCESS_SUIVI)
?? new(TypeFonction.EVENEMENT);

            //Remplir modèle
            ArborescenceFonctSuiviDataController.InitialiserModele(ref ListArboModel!);

            model.ListArboModel = ListArboModel;

            return View(model);
        }

        /// <summary>
        /// Web service qui renvoi la liste de model process
        /// </summary>
        /// <param name="pLoadOptions"></param>
        /// <param name="pProcessIDList"></param>
        /// <returns></returns>
        public IActionResult GetProcess(DataSourceLoadOptions pLoadOptions, string? pProcessIDList)
        {
            JsonResult result = GetJsonResult(string.Empty, pLoadOptions);

            if (!string.IsNullOrWhiteSpace(pProcessIDList))
            {
                using (BLLProcess bllProcess = new())
                {
                    bllProcess.FillByListIDInt(Array.ConvertAll(pProcessIDList.Split(','), int.Parse).ToList());
                    result = GetJsonResult((from x in bllProcess.DataSource select new ProcessModel(x)).ToList(), pLoadOptions);
                    bllProcess.DataSource.Clear();
                }
            }

            return result;
        }

        /// <summary>
        /// Web service qui renvoi la liste de model process value
        /// </summary>
        /// <param name="pLoadOptions"></param>
        /// <param name="pDateStart"></param>
        /// <param name="pDateStop"></param>
        /// <param name="pProcessIDList"></param>
        /// <returns></returns>
        public IActionResult GetProcessValues(DataSourceLoadOptions pLoadOptions, string pDateStart, string pDateStop, string?
pProcessIDList)
        {
            JsonResult result = GetJsonResult(string.Empty, pLoadOptions);

            if (!string.IsNullOrWhiteSpace(pProcessIDList))
            {
                DateTime.TryParse(pDateStart, out DateTime dtStart);
                DateTime.TryParse(pDateStop, out DateTime dtStop);

                using (BLLProcessValues_Process bllProcessValues = new())
                {
                    bllProcessValues.FillByListProcessIDAndPeriode(Array.ConvertAll(pProcessIDList.Split(','),
int.Parse).ToList(), dtStart, dtStop);
                    result = GetJsonResult((from x in bllProcessValues.DataSource select new ProcessValuesModel(x)).ToList(),
pLoadOptions);

                    bllProcessValues.DataSource.Clear();
                }
            }
            return result;
        }

        /// <summary>
        /// Renvoi les données pour le popup d'édition
        /// </summary>
```

```
        /// <param name="pLoadOptions"></param>
        /// <param name="pProcessID"></param>
        /// <param name="pValueID"></param>
        /// <param name="pTabIndex"></param>
        /// <returns></returns>
        public IActionResult GetAppointmentDonneesAssocieesCommentaires(DataSourceLoadOptions pLoadOptions, int pProcessID, long
pValueID, int pTabIndex)
        {
            JsonResult result = GetJsonResult(string.Empty, pLoadOptions);

            using (BLLProcessRecetteValues_ProcessRecette_ProcessValues bllProcessRecVal = new())
            using (BLLProcessValueComm bllProcValueComm = new())
            {
                //Données pour l'onglet valeurs
                if (pTabIndex == 0)
                {
                    bllProcessRecVal.ProcessID = pProcessID;
                    bllProcessRecVal.FillByValueID(pValueID);
                    result = GetJsonResult((from x in bllProcessRecVal.DataSource select new
ProcessRecetteValuesModel(x)).ToList(), pLoadOptions);
                }
                //Données pour l'onglet commentaires
                else if (pTabIndex == 1)
                {
                    bllProcValueComm.FillByProcessIDAndValueID(pProcessID, pValueID);
                    result = GetJsonResult((from x in bllProcValueComm.DataSource select new ProcessValueCommModel(x)).ToList(),
pLoadOptions);
                }

                //Libère
                bllProcessRecVal.DataSource.Clear();
                bllProcValueComm.DataSource.Clear();
            }
            return result;
        }

    }
}
```

Planning des Ordres de Travail