Mettre en œuvre une application mobile FranceConnect avec Xamarin

Kit de démarrage FranceConnect

Publication : Juin 2016

Version : 2016-06

Auteur : Charles de Vandière

Contributeurs/Relecteurs : Philippe Beraud

Pour les dernières informations sur le kit de démarrage FranceConnect, consultez

aka.ms/FranceConnect

Sommaire

[A propos du système FranceConnect 4](#_Toc460222439)

[A propos du kit de démarrage FranceConnect 5](#_Toc460222440)

[Licence 5](#_Toc460222441)

[Commentaires 5](#_Toc460222442)

[Prérequis 6](#_Toc460222443)

[Connaissances et attendu 6](#_Toc460222444)

[Créer une application mobile FranceConnect avec Xamarin 7](#_Toc460222445)

[Obtenir des identifiants FranceConnect 7](#_Toc460222446)

[Créer une application mobile 7](#_Toc460222447)

[Créer le code commun aux applications mobiles 8](#_Toc460222448)

[Créer un client FranceConnect 8](#_Toc460222449)

[Créer une Application Universelle Windows 10 12](#_Toc460222450)

[Ajouter l’application universelle à la solution 12](#_Toc460222451)

[MVVM 13](#_Toc460222452)

[WebView 14](#_Toc460222453)

[Modifier la page principale 15](#_Toc460222454)

[Afficher l’identité pivot 16](#_Toc460222455)

[Obtenir des données depuis un fournisseur de données 17](#_Toc460222456)

[Créer une application Android 19](#_Toc460222457)

[WebView 19](#_Toc460222458)

[Modifier la page principale 21](#_Toc460222459)

[Afficher l’identité pivot 22](#_Toc460222460)

[Obtenir des données depuis un fournisseur de données 24](#_Toc460222461)

[Créer une application iOS 29](#_Toc460222462)

[Storyboard 29](#_Toc460222463)

[Vue principale 31](#_Toc460222464)

[Afficher l’identité pivot 32](#_Toc460222465)

[Obtenir des données depuis un fournisseur de données 33](#_Toc460222466)

[Déployer le canevas d’application mobile préconfiguré du kit de démarrage 36](#_Toc460222467)

[Obtenir les sources de l’application mobile 36](#_Toc460222468)

[Télécharger les sources de l’application mobile 36](#_Toc460222469)

[Cloner les sources de l’application mobile 37](#_Toc460222470)

[Organisation du code source du canevas de l’application mobile 38](#_Toc460222471)

[Comprendre l’organisation du projet 38](#_Toc460222472)

[Appréhender les éléments de configuration 38](#_Toc460222473)

[Publier l’application sur les stores 39](#_Toc460222474)

[Tester l’application mobile 40](#_Toc460222475)

[Configurer le compte FranceConnect 40](#_Toc460222476)

[Se connecter avec le système FranceConnect 40](#_Toc460222477)

[Invoquer un service de données via le système FranceConnect 42](#_Toc460222478)

[Annexe A. Références techniques 43](#_Toc460222479)

# A propos du système FranceConnect

La plateforme FranceConnect est un nouveau système d’identification à l’initiative de la Direction interministérielle du numérique et du système d’information et de communication de l’État (DINSIC) visant à faciliter l’accès des usagers aux services numériques de l’administration en ligne en évitant à tout un chacun de devoir créer un nouveau compte lors de l’accès à un nouveau service et donc à se remémorer différents mots de passe pour l’ensemble des services accédés.

Pour cela, le système FranceConnect permet à chaque usager de disposer d’un mécanisme d’identification reconnu par les téléservices de l’administration au travers du bouton FranceConnect. Lors de l’accès à un nouveau service, et au-delà de la possibilité toujours proposée de s’inscrire vis-à-vis d’une autorité administrative que ne connaitrait pas encore l’usager, le bouton permet de sélectionner une identité compatible dont l’usager disposerait déjà (impôts, sécurité sociale, Caf, etc.) et de l’utiliser dans ce contexte.

Le système FranceConnect agit alors en tant que tiers de confiance mettant en relation usagers et autorités administratives auprès de fournisseurs d'identité compatibles et reconnus. Le système FranceConnect permet donc sous le contrôle de l’usager et de son consentement la fédération de comptes. L’expérience utilisateur (UX) ainsi proposée s’apparente à celle communément proposée par une application dont les comptes auraient été fédérées avec un réseau Facebook. Cela suppose toutefois que le niveau d’authentification (faible, substantiel ou fort) pris en charge par le fournisseur d’identité pour le compte sélectionné soit au moins équivalent à celui imposé au niveau du service numérique accédé.

Le système d’identification FranceConnect couvre les rôles suivants :

* **Fournisseur de service** (FS). Un site web qui s’appuie sur FranceConnect pour identifier les internautes qui le souhaitent (ex. Ameli.fr, Laposte.fr).
* **Fournisseur de données** (FD). Un site/API web qui propose des ressources à un fournisseur de service, et qui utilise FranceConnect pour sécuriser cet échange (ex. EDF, impôts).
* **Fournisseur d’identité** (FI). Un environnent Web qui permet à FranceConnect d'identifier et d'authentifier un internaute (ex. Ameli.fr, Laposte.fr)

Enfin, le système FranceConnect est conforme à la directive européenne eIDAS (Electronic Identification and Signature) proposant ainsi une interopérabilité des systèmes d’identification utilisés par les Etats membres pour accéder à leurs services en ligne. Ainsi, un autre pays de l’Union sera ainsi de facto reconnu sur ledit service numérique de l’Administration en ligne française si le système d’identification d’origine est à la fois compatible eIDAS et fédéré avec FranceConnect.

Compte tenu des éléments précédents, le système FranceConnect – vous l’aurez compris - constitue une composante essentielle de la stratégie d’[Etat plateforme](http://etatplateforme.modernisation.gouv.fr/) poussée par la DINSIC en facilitant l’accès des usagers aux différents services numériques disponibles tout en renforçant la confiance de ces mêmes usagers dans ces services au travers de la prise en charge reproductible et contrôlée des identités existantes compatibles de leur choix. Ceci représente à n’en point douter un élément central dans les échanges de l’Administration électronique.

L’article [L'administration change avec le numérique](http://www.modernisation.gouv.fr/ladministration-change-avec-le-numerique/par-son-systeme-dinformation/france-connect-un-acces-universel-aux-administrations-en-ligne) présente plus en détail cette initiative de la DINSIC.

# A propos du kit de démarrage FranceConnect

Le kit de démarrage FranceConnect sous licence libre est une initiative menée par les équipes Microsoft en charge du [Secteur Public](http://www.microsoft.com/france/entreprises/secteur-public/solutions-pour-secteur-public.aspx)[[1]](#footnote-1). Il propose un ensemble d’exemples/accélérateurs facilitant la mise en œuvre des différents rôles prévus par le système d’identification FranceConnect avec les services, produits et technologies Microsoft.



Plus précisément, ce document détaille la mise en œuvre d’un canevas d’application mobile une application mobile compatible FranceConnect. Il s’appuie pour cela sur la technologie [Xamarin](http://www.dotnetfoundation.org/xamarin.mobile)[[2]](#footnote-2) mise à disposition sous licence libre par la [fondation .NET](http://www.dotnetfoundation.org/)[[3]](#footnote-3).

Note Les très nombreuses contributions de Microsoft dans le domaine de l’Open Source se traduisent en particulier par la création de cette fondation indépendante afin de favoriser le développement ouvert et la collaboration autour du nombre toujours croissant de technologies Open Source pour .NET.

Cette fondation sert de forum pour tous développeurs issus de communautés et d’entreprises commerciales afin de renforcer l’avenir de l’écosystème .NET en favorisant la transparence, la participation communautaire et l’innovation rapide. La fondation .NET préside à la destinée de plus de 32 projets en mode Open Source sous son égide dont le projet Xamarin.mobile dont il est question ici.

Un accompagnement pas-à-pas vous est proposé pour toutes les étapes du processus de mise en œuvre d’un canevas d’application mobile en partant si vous le souhaitez d’un projet générique Xamarin. Une solution complète prête à être configurée et déployée vous est également proposée.

Ainsi, vous abordez directement si vous le souhaitez comment configurer et publier le canevas d’application mobile en Xamarin mise à disposition.

## Licence

Le kit de démarrage FranceConnect est publié sous le contrat de licence de logiciel libre MIT tel que décrit à l’adresse <https://opensource.org/licenses/MIT>.

## Commentaires

Votre opinion nous intéresse. N’hésitez pas à nous transmettre vos commentaires via la boîte aux lettres FranceConnect (<mailto:franceconnect@microsoft.com>) afin que nous puissions améliorer la solution communautaire ainsi mise à disposition.

## Prérequis

Le canevas du kit de démarrage FranceConnect pour réaliser une application mobile en Xamarin nécessite un ordinateur sous Windows 7 ou ultérieur pour les applications Android et Windows 10 universelle et un ordinateur sous Mac OS pour la compilation de l’application iOS.

## Connaissances et attendu

La mise en œuvre d’une application mobile avec le kit de démarrage FranceConnect requiert une expérience préalable :

* De l’environnement de développement [Microsoft Visual Studio Community 2015](https://go.microsoft.com/fwlink/?LinkId=691978&clcid=0x40c)[[4]](#footnote-4),
* De la programmation avec le Microsoft Framework .NET,
* De la technologie Xamarin,
* Des dernières technologies Web comme ASP.NET Core MVC,

Note L’article [Introduction to ASP.NET Core](https://docs.asp.net/en/latest/conceptual-overview/aspnet.html)[[5]](#footnote-5) propose comme son nom l’indique une première introduction à ASP.NET Core (anciennement ASP.NET 5). La page [ASP.NET Core Documentation](https://docs.asp.net/en/latest/)[[6]](#footnote-6) donne par ailleurs accès à l’ensemble de la documentation disponible à date.

* Du développement et de l’utilisation d’interfaces Web de style REST (Representational State Transfer) pour l’invocation de fournisseurs de données (FD),

# Créer une application mobile FranceConnect avec Xamarin

Cette section décrit comment créer une application mobile FranceConnect depuis Visual Studio 2015 en partant de la création d’un projet générique.

Si vous le souhaitez, vous pouvez utiliser directement le projet complet terminé et préconfiguré proposé dans le cadre du kit de démarrage FranceConnect : [franceconnect-mobile-app-xamarin-native](https://github.com/FranceConnectSamples/franceconnect-mobile-app-xamarin-native)[[7]](#footnote-7).

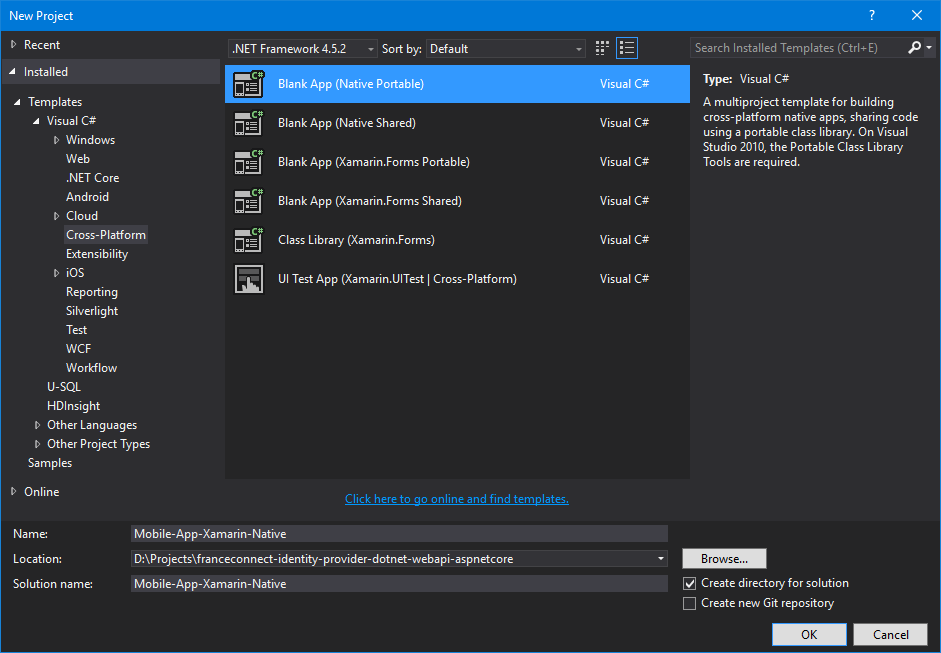
Pour cela, rendez-vous à la section § *Déployer l’application mobile préconfiguré du kit de démarrage*.

## Obtenir des identifiants FranceConnect

Avant de créer votre application mobile, vous devez vous [inscrire](https://doc.integ01.dev-franceconnect.fr/inscription)[[8]](#footnote-8) sur FranceConnect pour obtenir des identifiants (***client\_id***et ***client\_secret***) pour votre application. Notez bien ces valeurs ; ces dernières vous serviront dans la configuration de votre solution.

## Créer une application mobile

Une fois les identifiants obtenus, vous pouvez créer une nouvelle application. Choisissez un projet de type **Blank App (Native Portable)** parmi les différents gabarits (templates) proposés :



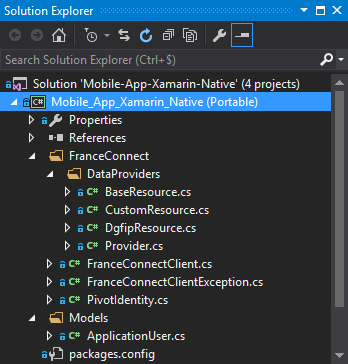
Assistant de création de projet

## Créer le code commun aux applications mobiles

### Créer un client FranceConnect

Les trois applications cible (Android, iOS, et Windows 10 universelle) vont avoir besoin d’identifier leurs utilisateurs via FranceConnect. Afin d’éviter la duplication de code, toute la logique d’identification va se trouver dans le projet partagé.

Voici l’architecture du projet partagé :



#### Identité pivot

Créer une classe **PivotIdentity** :

public class PivotIdentity

{

public string Sub { get; set; }

public string Gender { get; set; }

public DateTimeOffset Birthdate { get; set; }

public string Given\_name { get; set; }

public string Family\_name { get; set; }

public string Email { get; set; }

}

#### Créer un client OpenID Connect (OIDC)

Importez le package NuGet IdentityModel (v 1.11.0).

Créer une classe **FranceConnectClientException** pour jeter des exceptions personnalisées :

public class FranceConnectClientException : Exception

{

public FranceConnectClientException()

{

}

public FranceConnectClientException(string message) : base (message)

{

}

}

Ajoutez une classe **FranceConnectClient** qui va servir de client OIDC pour FranceConnect :

public class FranceConnectClient

{

public const string RedirectUri = "http://Mobile-App-Xamarin-Native/signin\_oidc";

private const string AuthorizationEndpoint = "https://fcp.integ01.dev-franceconnect.fr/api/v1/authorize";

private const string TokenEndpoint = "https://fcp.integ01.dev-franceconnect.fr/api/v1/token";

private const string UserInfoEndpoint = "https://fcp.integ01.dev-franceconnect.fr/api/v1/userinfo";

private const string ClientId = "<votre client\_id FranceConnect>";

private const string ClientSecret = "<votre client\_secret FranceConnect>";

private string \_state;

private string \_accessToken;

public string CreateSignInUri()

{

\_state = Guid.NewGuid().ToString("N");

var scope = new List<string> { "openid", "profile", "email" };

return CreateAuthorizeUri(scope, RedirectUri);

}

private string CreateAuthorizeUri(IEnumerable<string> scope, string redirectUri)

{

var authorizeRequest = new AuthorizeRequest(AuthorizationEndpoint);

return authorizeRequest.CreateAuthorizeUrl(

clientId: ClientId,

responseType: "code",

scope: string.Join(" ", scope),

redirectUri: redirectUri,

state: \_state,

nonce: Guid.NewGuid().ToString("N"));

}

public async Task AuthenticateAsync(string uri, string redirectUri)

{

var code = GetCode(uri);

\_accessToken = await GetAccessTokenAsync(code, redirectUri);

}

private string GetCode(string uri)

{

var authorizeResponse = new AuthorizeResponse(uri);

var state = authorizeResponse.Values["state"];

if (state != \_state)

{

throw new FranceConnectClientException("Invalide state");

}

var code = authorizeResponse.Code;

if (string.IsNullOrEmpty(code))

{

throw new FranceConnectClientException("No authorization code");

}

return code;

}

private async Task<string> GetAccessTokenAsync(string code, string redirectUri)

{

var tokenClient = new TokenClient(TokenEndpoint, ClientId, ClientSecret, AuthenticationStyle.PostValues);

var tokenResponse = await tokenClient.RequestAuthorizationCodeAsync(code, redirectUri);

if (tokenResponse.IsError || string.IsNullOrEmpty(tokenResponse.AccessToken))

{

throw new FranceConnectClientException("Unable to retrive access token");

}

return tokenResponse.AccessToken;

}

public async Task<PivotIdentity> GetUserInfoAsync()

{

if (string.IsNullOrEmpty(\_accessToken))

{

throw new FranceConnectClientException("Access token is null");

}

var userInfoClient = new UserInfoClient(new Uri(UserInfoEndpoint), \_accessToken);

var userInfoResponse = await userInfoClient.GetAsync();

if (userInfoResponse.IsError)

{

throw new FranceConnectClientException("Unable to retrive user informations");

}

return userInfoResponse.JsonObject.ToObject<PivotIdentity>();

}

}

#### Obtenir des données depuis un fournisseur de données (FD)

Tout d’abord, créez une classe pour chaque type de ressource :

namespace Mobile\_App\_Xamarin\_Native.FranceConnect.DataProviders

{

public abstract class BaseResource

{

}

public class CustomResource : BaseResource

{

public string ValueOne { get; set; }

public string ValueTwo { get; set; }

}

public class DgfipResource : BaseResource

{

public decimal Rfr { get; set; }

public char SitFam { get; set; }

public int NbPart { get; set; }

public Pac Pac { get; set; }

}

public class Pac

{

public int NbPac { get; set; }

}

}

Créez une énumération pour référencer les différents fournisseurs et une classe d’extension pour obtenir facilement les informations propres à chaque fournisseur :

namespace Mobile\_App\_Xamarin\_Native.FranceConnect.DataProviders

{

public enum Provider { DGFIP, Custom }

public static class ProviderExtensions

{

public static List<string> GetRequiredScope(this Provider provider)

{

switch (provider)

{

case Provider.DGFIP: return new List<string> { "dgfip\_rfr", "dgfip\_sitfam", "dgfip\_nbpac", "dgfip\_nbpart" };

case Provider.Custom: return new List<string> { "value1", "value2" };

default: throw new NotImplementedException();

}

}

public static Uri GetEndpoint(this Provider provider)

{

switch (provider)

{

case Provider.DGFIP: return new Uri("https://fdp.integ01.dev-franceconnect.fr/courtier/apiuniverselle/2016");

case Provider.Custom: return new Uri("http://franceconnect-data-provider-dotnet-webapi-aspnetcore.azurewebsites.net/api/values");

default: throw new NotImplementedException();

}

}

public static BaseResource ConvertResource(this Provider provider, string json)

{

switch (provider)

{

case Provider.DGFIP: return JsonConvert.DeserializeObject<DgfipResource>(json);

case Provider.Custom: return JsonConvert.DeserializeObject<CustomResource>(json);

default: throw new NotImplementedException();

}

}

public static string GetRedirectUri(this Provider provider)

{

return FranceConnectClient.BaseConsentRedirectUri + provider.ToString();

}

}

}

Afin que l’application puisse obtenir des données depuis un fournisseur de données (FD), vous allez modifier le client FranceConnect de la façon suivante :

Ajoutez la constante suivante qui va servir de base pour les URL de callback :

public const string BaseConsentRedirectUri = "http://Mobile-App-Xamarin-Native/consent/callback\_";

Ajoutez une méthode pour créer l’URL de consentement :

public string CreateConsentUri(Provider provider)

{

\_state = Guid.NewGuid().ToString("N");

var scope = new List<string> { "openid", "email" };

scope.AddRange(provider.GetRequiredScope());

return CreateAuthorizeUri(scope, provider.GetRedirectUri());

}

Ajoutez également une méthode de récupération des données avec le protocole OAuth2 :

public async Task<BaseResource> GetDataAsync(Provider provider)

{

if (string.IsNullOrEmpty(\_accessToken))

{

throw new FranceConnectClientException("Access token is null");

}

using (var client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", \_accessToken);

var response = await client.GetAsync(provider.GetEndpoint());

if (response.IsSuccessStatusCode)

{

var resource = await response.Content.ReadAsStringAsync();

return provider.ConvertResource(resource);

}

else if (response.StatusCode == HttpStatusCode.NotFound)

{

throw new FranceConnectClientException("La ressource demandée n'a pas été trouvée.");

}

else if (response.StatusCode == HttpStatusCode.Unauthorized)

{

throw new FranceConnectClientException("Vous n'êtes pas autorisé à accéder cette ressource.");

}

else

{

throw new FranceConnectClientException("Impossible de récupérer les données auprès du fournisseur choisi.");

}

}

}

Enfin, ajoutez une méthode permettant de retrouver le fournisseur à partir de l’URL de callback :

public static Provider GetProviderFromRedirectUri(string redirectUri)

{

var truncateUri = redirectUri.Substring(BaseConsentRedirectUri.Length);

if (truncateUri.StartsWith(Provider.DGFIP.ToString()))

{

return Provider.DGFIP;

}

else if (truncateUri.StartsWith(Provider.Custom.ToString()))

{

return Provider.Custom;

}

else

{

throw new FranceConnectClientException("Unable to retrive provider from the redirect uri");

}

}

#### ApplicationUser

Ajouter la classe ApplicationUser pour obtenir l’utilisateur authentifié à tout endroit de l’application :

public class ApplicationUser

{

private static ApplicationUser authenticatedUser;

public string Id { get; set; }

public string Gender { get; set; }

public DateTimeOffset Birthdate { get; set; }

public string Firstname { get; set; }

public string Lastname { get; set; }

public string Email { get; set; }

public static ApplicationUser AuthenticateUser(PivotIdentity pivotIdentity)

{

authenticatedUser = new ApplicationUser

{

Id = pivotIdentity.Sub,

Email = pivotIdentity.Email,

Gender = pivotIdentity.Gender,

Birthdate = pivotIdentity.Birthdate,

Firstname = pivotIdentity.Given\_name,

Lastname = pivotIdentity.Family\_name

};

return authenticatedUser;

}

public static bool IsAuthenticated

{

get { return authenticatedUser != null; }

}

public static ApplicationUser GetAuthenticatedUser()

{

if (!IsAuthenticated)

{

throw new NullReferenceException("No authenticated user");

}

return authenticatedUser;

}

public static void SignOut()

{

authenticatedUser = null;

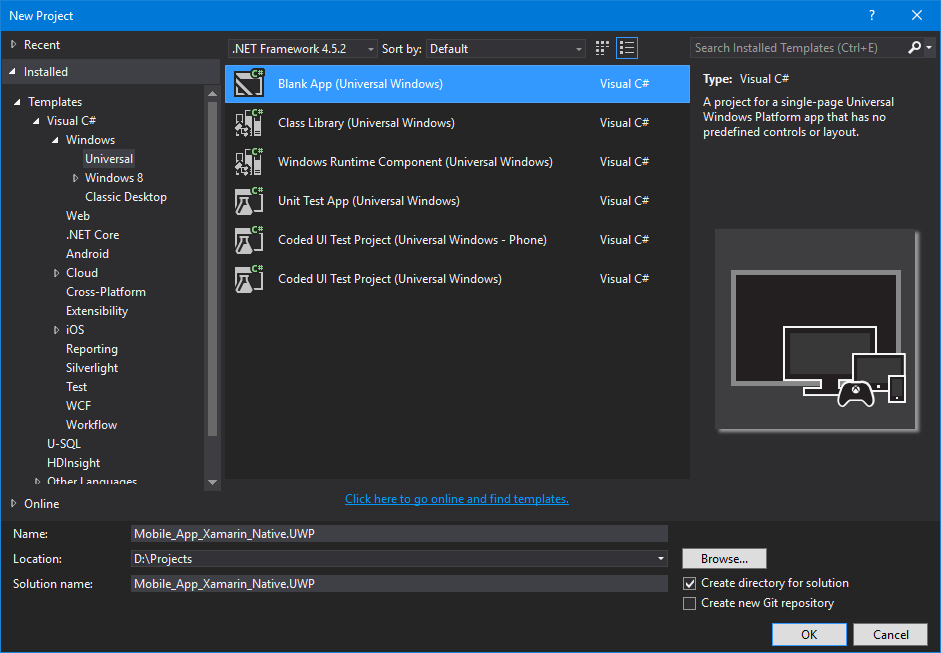
}

}

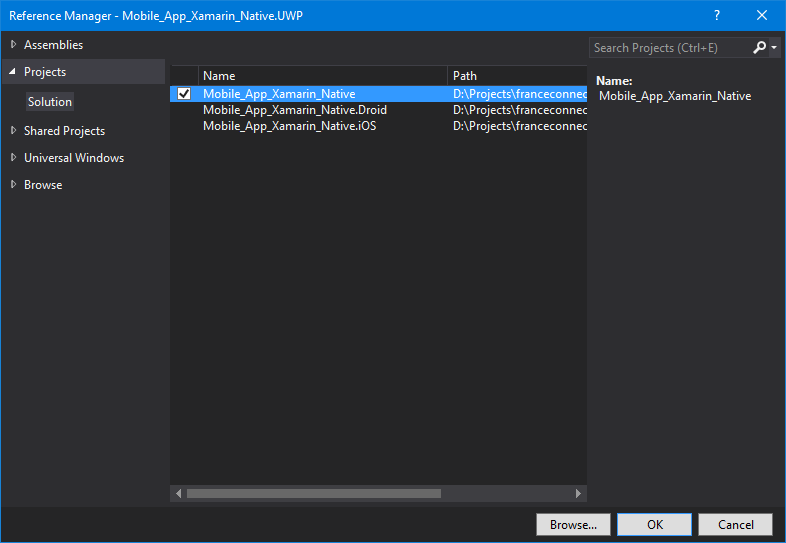
## Créer une Application Universelle Windows 10

### Ajouter l’application universelle à la solution

Ajoutez un nouveau projet de type **Blank App (Universal Windows)** à la solution comme suit.



Ajoutez la référence au projet portable :



### MVVM

Les applications Windows universelles utilisent le design pattern MVVM (Model – View – ViewModel).

Ajoutez donc deux nouveaux dossiers : **Views** et **ViewModels**. Les modèles se trouvent dans le projet portable afin d’être partagé à toutes les applications.

Créez un modèle de vue de base pour simplifier la création de modèle de vue :

public class BaseViewModel : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

protected void RaisePropertyChange(string propertyName)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

### WebView

Ajoutez une page contenant une WebView qui permettra à l’utilisateur de s’identifier et de consentir à l’obtention de ses données. Nommez la **WebViewPage**.

Dans un premier temps, ajouter une WebView à la page :

<WebView x:Name="OidcWebView"

VerticalAlignment="Stretch"

HorizontalAlignment="Stretch"

NavigationStarting="OidcWebView\_NavigationStarting" />

Dans un second temps, ajoutez le code suivant :

public sealed partial class WebViewPage : Page

{

private FranceConnectClient \_client;

public WebViewPage()

{

this.InitializeComponent();

\_client = new FranceConnectClient();

}

protected override void OnNavigatedTo(NavigationEventArgs e)

{

var navigationParams = (NavigationParameters)e.Parameter;

Uri uri;

switch (navigationParams.PageJob)

{

case NavigationParameters.Job.SignIn:

uri = new Uri(\_client.CreateSignInUri());

OidcWebView.ClearCookiesForUrl(uri);

break;

default:

throw new NotImplementedException();

}

OidcWebView.Navigate(uri);

base.OnNavigatedTo(e);

}

private async void OidcWebView\_NavigationStarting(WebView sender, WebViewNavigationStartingEventArgs args)

{

if (args.Uri.AbsoluteUri.StartsWith(FranceConnectClient.RedirectUri, StringComparison.OrdinalIgnoreCase))

{

try

{

await \_client.AuthenticateAsync(args.Uri.AbsoluteUri, FranceConnectClient.RedirectUri);

ApplicationUser.AuthenticateUser(await \_client.GetUserInfoAsync());

}

catch (FranceConnectClientException fce)

{

var errorDialog = new MessageDialog(fce.Message, "Erreur");

await errorDialog.ShowAsync();

}

catch (Exception)

{

var errorDialog = new MessageDialog("Une erreur est survenue", "Erreur");

await errorDialog.ShowAsync();

}

finally

{

Frame.GoBack();

}

}

}

public class NavigationParameters

{

public enum Job { SignIn, Consent }

public Job PageJob { get; set; }

public Provider ResourceProvider { get; set; }

}

}

### Modifier la page principale

Ajoutez un modèle de vue pour la page principale :

public class MainPageViewModel : BaseViewModel

{

public Visibility IsAuthenticated

{

get { return ApplicationUser.IsAuthenticated ? Visibility.Visible : Visibility.Collapsed; }

}

public string LoginLogoutLabel

{

get { return ApplicationUser.IsAuthenticated ? "Se déconnecter" : "Connexion"; }

}

public void Refresh()

{

RaisePropertyChange(nameof(IsAuthenticated));

RaisePropertyChange(nameof(LoginLogoutLabel));

}

}

Ajoutez un bouton sur la page principale :

<Button Content="{x:Bind PageViewModel.LoginLogoutLabel, Mode=OneWay}" Click="LoginLogoutButton\_Click" />

Modifiez le code de la page :

public sealed partial class MainPage : Page

{

public MainPageViewModel PageViewModel { get; private set; }

public MainPage()

{

this.InitializeComponent();

PageViewModel = new MainPageViewModel();

}

private async void LoginLogoutButton\_Click(object sender, RoutedEventArgs e)

{

if (!ApplicationUser.IsAuthenticated)

{

Frame.Navigate(typeof(WebViewPage),

new WebViewPage.NavigationParameters

{

PageJob = WebViewPage.NavigationParameters.Job.SignIn

});

}

else

{

var dialog = new MessageDialog("Voulez-vous vraiment vous déconnecter ?", "Se déconnecter");

dialog.Commands.Add(new UICommand("Oui") { Id = 0 });

dialog.Commands.Add(new UICommand("Non") { Id = 1 });

dialog.DefaultCommandIndex = 0;

dialog.CancelCommandIndex = 1;

var result = await dialog.ShowAsync();

if ((int)result.Id == 0)

{

ApplicationUser.SignOut();

PageViewModel.Refresh();

}

}

}

}

### Afficher l’identité pivot

Créez un modèle de vue **UserPageViewModel** :

public class UserPageViewModel : BaseViewModel

{

private ApplicationUser user;

public ApplicationUser User

{

get { return user; }

set

{

if (value != user)

{

user = value;

RaisePropertyChange(nameof(User));

}

}

}

}

Ajoutez une page **UserPage** au projet :

<StackPanel Grid.Row="1" Padding="12,0">

<TextBlock Text="Id" Margin="0,20,0,0" FontSize="18" />

<TextBlock Text="{x:Bind PageViewModel.User.Id, Mode=OneWay}" Margin="0,8,0,0" />

<TextBlock Text="Email" Margin="0,20,0,0" FontSize="18" />

<TextBlock Text="{x:Bind PageViewModel.User.Email, Mode=OneWay}" Margin="0,8,0,0" />

<TextBlock Text="Gender" Margin="0,20,0,0" FontSize="18" />

<TextBlock Text="{x:Bind PageViewModel.User.Gender, Mode=OneWay}" Margin="0,8,0,0" />

<TextBlock Text="Birthdate" Margin="0,20,0,0" FontSize="18" />

<TextBlock Text="{x:Bind PageViewModel.User.Birthdate, Mode=OneWay}" Margin="0,8,0,0" />

<TextBlock Text="Firstname" Margin="0,20,0,0" FontSize="18" />

<TextBlock Text="{x:Bind PageViewModel.User.Firstname, Mode=OneWay}" Margin="0,8,0,0" />

<TextBlock Text="Lastname" Margin="0,20,0,0" FontSize="18" />

<TextBlock Text="{x:Bind PageViewModel.User.Lastname, Mode=OneWay}" Margin="0,8,0,0" />

</StackPanel>

Puis modifiez le code de la page :

public sealed partial class UserPage : Page

{

public UserPageViewModel PageViewModel { get; set; }

public UserPage()

{

this.InitializeComponent();

PageViewModel = new UserPageViewModel();

}

protected override void OnNavigatedTo(NavigationEventArgs e)

{

base.OnNavigatedTo(e);

PageViewModel.User = ApplicationUser.GetAuthenticatedUser();

}

}

Ajoutez un bouton sur la page principale pour permettre l’accès à cette page :

<Button Visibility="{x:Bind PageViewModel.IsAuthenticated, Mode=OneWay}" Click="PivotIdentityButton\_Click" />

Et modifiez le code de la page principale :

private void PivotIdentityButton\_Click(object sender, RoutedEventArgs e)

{

Frame.Navigate(typeof(UserPage));

}

### Obtenir des données depuis un fournisseur de données

Tout d’abord, créez des modèles de vue pour chaque fournisseur de données :

namespace Mobile\_App\_Xamarin\_Native.UWP.ViewModels

{

public class DgfipResourcePageViewModel : BaseViewModel

{

private DgfipResource dgfipResource;

public DgfipResource DgfipResource

{

get { return dgfipResource; }

set

{

if (value != dgfipResource)

{

dgfipResource = value;

RaisePropertyChange(nameof(DgfipResource));

}

}

}

}

public class CustomResourcePageViewModel : BaseViewModel

{

private CustomResource customResource;

public CustomResource CustomResource

{

get { return customResource; }

set

{

if (value != customResource)

{

customResource = value;

RaisePropertyChange(nameof(CustomResource));

}

}

}

}

}

Créez ensuite les vues correspondantes :

1. **DGFIP** :

<StackPanel Grid.Row="1" Padding="12,0">

<TextBlock Text="Revenu fiscal de référence" Margin="0,20,0,0" FontSize="18" />

<TextBlock Text="{x:Bind PageViewModel.DgfipResource.Rfr, Mode=OneWay}" Margin="0,8,0,0" />

<TextBlock Text="Situation familiale" Margin="0,20,0,0" FontSize="18" />

<TextBlock Text="{x:Bind PageViewModel.DgfipResource.SitFam, Mode=OneWay}" Margin="0,8,0,0" />

<TextBlock Text="Nombre de parts" Margin="0,20,0,0" FontSize="18" />

<TextBlock Text="{x:Bind PageViewModel.DgfipResource.NbPart, Mode=OneWay}" Margin="0,8,0,0" />

<TextBlock Text="Nombre de personnes à charge" Margin="0,20,0,0" FontSize="18" />

<TextBlock Text="{x:Bind PageViewModel.DgfipResource.Pac.NbPac, Mode=OneWay}" Margin="0,8,0,0" />

</StackPanel>

1. **Custom** :

<StackPanel Grid.Row="1" Padding="12,0">

<TextBlock Text="Valeur 1" Margin="0,20,0,0" FontSize="18" />

<TextBlock Text="{x:Bind PageViewModel.CustomResource.ValueOne, Mode=OneWay}" Margin="0,8,0,0" />

<TextBlock Text="Valeur 2" Margin="0,20,0,0" FontSize="18" />

<TextBlock Text="{x:Bind PageViewModel.CustomResource.ValueTwo, Mode=OneWay}" Margin="0,8,0,0" />

</StackPanel>

Puis, modifier le code des pages pour qu’elles affichent les données et qu’elles suppriment la page de consentement de l’historique de navigation, afin que l’utilisateur soit redirigé vers la page d’accueil lorsqu’il appuie sur le bouton retour.

public sealed partial class DgfipResourcePage : Page

{

public DgfipResourcePageViewModel PageViewModel { get; set; }

public DgfipResourcePage()

{

this.InitializeComponent();

PageViewModel = new DgfipResourcePageViewModel();

}

protected override void OnNavigatedTo(NavigationEventArgs e)

{

PageViewModel.DgfipResource = (DgfipResource)e.Parameter;

if (Frame.CanGoBack)

{

PageStackEntry lastPage = Frame.BackStack[Frame.BackStackDepth - 1];

if (lastPage.SourcePageType == typeof(WebViewPage))

{

Frame.BackStack.Remove(lastPage);

}

}

base.OnNavigatedTo(e);

}

}

public sealed partial class CustomResourcePage : Page

{

public CustomResourcePageViewModel PageViewModel { get; set; }

public CustomResourcePage()

{

this.InitializeComponent();

PageViewModel = new CustomResourcePageViewModel();

}

protected override void OnNavigatedTo(NavigationEventArgs e)

{

PageViewModel.CustomResource = (CustomResource)e.Parameter;

if (Frame.CanGoBack)

{

PageStackEntry lastPage = Frame.BackStack[Frame.BackStackDepth - 1];

if (lastPage.SourcePageType == typeof(WebViewPage))

{

Frame.BackStack.Remove(lastPage);

}

}

base.OnNavigatedTo(e);

}

}

Une fois les pages créées, ajoutez les boutons sur la page principale :

<Button Visibility="{x:Bind PageViewModel.IsAuthenticated, Mode=OneWay}" Click="DgfipResourceButton\_Click" />

<Button Visibility="{x:Bind PageViewModel.IsAuthenticated, Mode=OneWay}" Click="CustomResourceButton\_Click" />

Et modifiez le code de la page :

private void DgfipResourceButton\_Click(object sender, RoutedEventArgs e)

{

Frame.Navigate(typeof(WebViewPage),

new WebViewPage.NavigationParameters

{

PageJob = WebViewPage.NavigationParameters.Job.Consent,

ResourceProvider = FranceConnect.DataProviders.Provider.DGFIP

});

}

private void CustomResourceButton\_Click(object sender, RoutedEventArgs e)

{

Frame.Navigate(typeof(WebViewPage),

new WebViewPage.NavigationParameters

{

PageJob = WebViewPage.NavigationParameters.Job.Consent,

ResourceProvider = FranceConnect.DataProviders.Provider.Custom

});

}

Enfin, modifiez le code de la WebViewPage :

protected override void OnNavigatedTo(NavigationEventArgs e)

{

...

switch (navigationParams.PageJob)

{

case NavigationParameters.Job.SignIn:

...

break;

case NavigationParameters.Job.Consent:

uri = new Uri(\_client.CreateConsentUri(navigationParams.ResourceProvider));

break;

default:

throw new NotImplementedException();

}

...

}

private async void OidcWebView\_NavigationStarting(WebView sender, WebViewNavigationStartingEventArgs args)

{

if (args.Uri.AbsoluteUri.StartsWith(FranceConnectClient.RedirectUri, StringComparison.OrdinalIgnoreCase))

{

...

}

else if (args.Uri.AbsoluteUri.StartsWith(FranceConnectClient.BaseConsentRedirectUri, StringComparison.OrdinalIgnoreCase))

{

try

{

var provider = FranceConnectClient.GetProviderFromRedirectUri(args.Uri.AbsoluteUri);

await \_client.AuthenticateAsync(args.Uri.AbsoluteUri, provider.GetRedirectUri());

var resource = await \_client.GetDataAsync(provider);

switch (provider)

{

case Provider.DGFIP:

Frame.Navigate(typeof(DgfipResourcePage), resource);

break;

case Provider.Custom:

Frame.Navigate(typeof(CustomResourcePage), resource);

break;

default:

throw new NotImplementedException();

}

}

catch (FranceConnectClientException fce)

{

var errorDialog = new MessageDialog(fce.Message, "Erreur");

await errorDialog.ShowAsync();

Frame.GoBack();

}

catch (Exception)

{

var errorDialog = new MessageDialog("Une erreur est survenue", "Erreur");

await errorDialog.ShowAsync();

Frame.GoBack();

}

}

}

## Créer une application Android

### WebView

Ajoutez une nouvelle activité pour la WebView. Nommez la **WebViewActivity**. Créez également le layout correspondant, **activity\_web\_view** :

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<WebView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:id="@+id/oidcWebView" />

</LinearLayout>

[Activity(NoHistory = true)]

public class WebViewActivity : Activity

{

public enum Job { SignIn, Consent }

public const string JobExtra = "Job";

public const string ResourceProviderExtra = "ResourceProvider";

private WebView mOidcWebView;

protected override void OnCreate(Bundle savedInstanceState)

{

base.OnCreate(savedInstanceState);

SetContentView(Resource.Layout.activity\_web\_view);

var job = (Job)Intent.GetIntExtra(JobExtra, (int)Job.SignIn);

var provider = (Provider)Intent.GetIntExtra(ResourceProviderExtra, (int)Provider.DGFIP);

switch (job)

{

case Job.SignIn:

this.Title = Resources.GetString(Resource.String.login);

var cookieManager = CookieManager.Instance;

cookieManager.RemoveAllCookie();

break;

case Job.Consent:

this.Title = Resources.GetString(Resource.String.consent);

break;

}

OidcWebViewClient client = new OidcWebViewClient(this, job, provider);

mOidcWebView = FindViewById<WebView>(Resource.Id.oidcWebView);

mOidcWebView.Settings.JavaScriptEnabled = true;

mOidcWebView.SetWebViewClient(client);

mOidcWebView.LoadUrl(client.GetStartUrl());

}

private class OidcWebViewClient : WebViewClient

{

private FranceConnectClient \_client;

private Job \_job;

private Provider \_provider;

private Activity \_activity;

public OidcWebViewClient(Activity activity, Job job, Provider provider)

{

\_activity = activity;

\_client = new FranceConnectClient();

\_job = job;

\_provider = provider;

}

public string GetStartUrl()

{

switch (\_job)

{

case Job.SignIn:

return \_client.CreateSignInUri();

default:

throw new NotImplementedException();

}

}

public override bool ShouldOverrideUrlLoading(WebView view, string url)

{

if (url.StartsWith(FranceConnectClient.RedirectUri, StringComparison.OrdinalIgnoreCase))

{

view.Visibility = ViewStates.Gone;

var task = Task.Factory.StartNew(async () =>

{

await \_client.AuthenticateAsync(url, FranceConnectClient.RedirectUri);

ApplicationUser.AuthenticateUser(await \_client.GetUserInfoAsync());

});

task.Unwrap().ContinueWith((antecedent) =>

{

\_activity.Finish();

},

TaskContinuationOptions.OnlyOnRanToCompletion);

task.Unwrap().ContinueWith((antecedent) =>

{

\_activity.RunOnUiThread(() =>

{

AlertDialog.Builder alert = new AlertDialog.Builder(\_activity);

alert.SetTitle("Erreur");

alert.SetMessage(antecedent.Exception.Message);

alert.SetPositiveButton("Fermer", (senderAlert, args) =>

{

\_activity.Finish();

});

Dialog dialog = alert.Create();

dialog.Show();

});

},

TaskContinuationOptions.OnlyOnFaulted);

}

return base.ShouldOverrideUrlLoading(view, url);

}

}

}

### Modifier la page principale

Ajoutez un bouton pour accéder à la WebView sur le layout de l’activité principale :

<Button

android:id="@+id/loginLogoutButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="@string/login" />

Et modifiez le code de la l’activité :

[Activity(Label = "FranceConnect Xamarin", MainLauncher = true)]

public class MainActivity : Activity

{

private Button mLoginLogoutButton;

protected override void OnCreate(Bundle bundle)

{

base.OnCreate(bundle);

SetContentView(Resource.Layout.activity\_main);

mLoginLogoutButton = FindViewById<Button>(Resource.Id.loginLogoutButton);

ShowHideElements();

mLoginLogoutButton.Click += LoginLogoutButton\_Click;

}

protected override void OnStart()

{

base.OnStart();

ShowHideElements();

}

private void ShowHideElements()

{

if (ApplicationUser.IsAuthenticated)

{

mLoginLogoutButton.Text = Resources.GetString(Resource.String.logout);

}

else

{

mLoginLogoutButton.Text = Resources.GetString(Resource.String.login);

}

}

private void LoginLogoutButton\_Click(object sender, EventArgs e)

{

if (ApplicationUser.IsAuthenticated)

{

AlertDialog.Builder alert = new AlertDialog.Builder(this);

alert.SetTitle("Se déconnecter");

alert.SetMessage("Voulez-vous vraiment vous déconnecter ?");

alert.SetPositiveButton("Oui", (senderAlert, args) =>

{

ApplicationUser.SignOut();

});

alert.SetNegativeButton("Non", (senderAlert, args) => { });

Dialog dialog = alert.Create();

dialog.Show();

}

else

{

var activity = new Intent(this, typeof(WebViewActivity));

activity.PutExtra(WebViewActivity.JobExtra, (int)WebViewActivity.Job.SignIn);

StartActivity(activity);

}

}

}

### Afficher l’identité pivot

Ajoutez une nouvelle activité **PivotIdentityActivity** :

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<TextView

android:text="@string/pivot\_identity\_id"

android:textAppearance="?android:attr/textAppearanceMedium"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingTop="16dp"

android:paddingBottom="8dp"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:id="@+id/idTextView"

android:text="Text"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:text="@string/pivot\_identity\_email"

android:textAppearance="?android:attr/textAppearanceMedium"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingTop="16dp"

android:paddingBottom="8dp"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:id="@+id/emailTextView"

android:text="Text"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:text="@string/pivot\_identity\_gender"

android:textAppearance="?android:attr/textAppearanceMedium"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingTop="16dp"

android:paddingBottom="8dp"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:id="@+id/genderTextView"

android:text="Text"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:text="@string/pivot\_identity\_birthdate"

android:textAppearance="?android:attr/textAppearanceMedium"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingTop="16dp"

android:paddingBottom="8dp"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:id="@+id/birthdateTextView"

android:text="Text"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:text="@string/pivot\_identity\_firstname"

android:textAppearance="?android:attr/textAppearanceMedium"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingTop="16dp"

android:paddingBottom="8dp"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:id="@+id/firstnameTextView"

android:text="Text"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:text="@string/pivot\_identity\_lastname"

android:textAppearance="?android:attr/textAppearanceMedium"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingTop="16dp"

android:paddingBottom="8dp"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:id="@+id/lastnameTextView"

android:text="Text"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

</LinearLayout>

[Activity(Label = "Identité pivot")]

public class PivotIdentityActivity : Activity

{

public const string IdExtra = "Sub";

public const string EmailExtra = "Email";

public const string GenderExtra = "Gender";

public const string BirthdateExtra = "Birthdate";

public const string FirstnameExtra = "Firstname";

public const string LastnameExtra = "Lastname";

private TextView mIdTextView;

private TextView mEmailTextView;

private TextView mGenderTextView;

private TextView mBirthdateTextView;

private TextView mFirstnameTextView;

private TextView mLastnameTextView;

protected override void OnCreate(Bundle savedInstanceState)

{

base.OnCreate(savedInstanceState);

SetContentView(Resource.Layout.activity\_pivot\_identity);

mIdTextView = FindViewById<TextView>(Resource.Id.idTextView);

mEmailTextView = FindViewById<TextView>(Resource.Id.emailTextView);

mGenderTextView = FindViewById<TextView>(Resource.Id.genderTextView);

mBirthdateTextView = FindViewById<TextView>(Resource.Id.birthdateTextView);

mFirstnameTextView = FindViewById<TextView>(Resource.Id.firstnameTextView);

mLastnameTextView = FindViewById<TextView>(Resource.Id.lastnameTextView);

mIdTextView.Text = Intent.GetStringExtra(IdExtra);

mEmailTextView.Text = Intent.GetStringExtra(EmailExtra);

mGenderTextView.Text = Intent.GetStringExtra(GenderExtra);

mBirthdateTextView.Text = Intent.GetStringExtra(BirthdateExtra);

mFirstnameTextView.Text = Intent.GetStringExtra(FirstnameExtra);

mLastnameTextView.Text = Intent.GetStringExtra(LastnameExtra);

}

}

Et modifiez le code de l’activité principale :

<Button

android:id="@+id/pivotIdentityButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingLeft="16dp"

android:paddingRight="16dp"

android:text="@string/pivot\_identity" />

public class MainActivity : Activity

{

...

private Button mPivotIdentityButton;

protected override void OnCreate(Bundle bundle)

{

...

mPivotIdentityButton = FindViewById<Button>(Resource.Id.pivotIdentityButton);

...

mPivotIdentityButton.Click += PivotIdentityButton\_Click;

}

private void ShowHideElements()

{

if (ApplicationUser.IsAuthenticated)

{

mPivotIdentityButton.Visibility = ViewStates.Visible;

mLoginLogoutButton.Text = Resources.GetString(Resource.String.logout);

}

else

{

mPivotIdentityButton.Visibility = ViewStates.Gone;

mLoginLogoutButton.Text = Resources.GetString(Resource.String.login);

}

}

...

private void PivotIdentityButton\_Click(object sender, EventArgs e)

{

var user = ApplicationUser.GetAuthenticatedUser();

if (user != null)

{

var activity = new Intent(this, typeof(PivotIdentityActivity));

activity.PutExtra(PivotIdentityActivity.IdExtra, user.Id);

activity.PutExtra(PivotIdentityActivity.EmailExtra, user.Email);

activity.PutExtra(PivotIdentityActivity.GenderExtra, user.Gender);

activity.PutExtra(PivotIdentityActivity.BirthdateExtra, user.Birthdate.ToString("D"));

activity.PutExtra(PivotIdentityActivity.FirstnameExtra, user.Firstname);

activity.PutExtra(PivotIdentityActivity.LastnameExtra, user.Lastname);

StartActivity(activity);

}

}

}

### Obtenir des données depuis un fournisseur de données

Créez deux nouvelles activités pour les données de la DGFIP et celle du fournisseur de données Custom :

1. **DGFIP** :

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<TextView

android:text="@string/dgfip\_rfr"

android:textAppearance="?android:attr/textAppearanceMedium"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingTop="16dp"

android:paddingBottom="8dp"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:id="@+id/rfrTextView"

android:text="Text"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:text="@string/dgfip\_sitfam"

android:textAppearance="?android:attr/textAppearanceMedium"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingTop="16dp"

android:paddingBottom="8dp"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:id="@+id/sitfamTextView"

android:text="Text"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:text="@string/dgfip\_nbpart"

android:textAppearance="?android:attr/textAppearanceMedium"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingTop="16dp"

android:paddingBottom="8dp"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:id="@+id/nbpartTextView"

android:text="Text"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:text="@string/dgfip\_nbpac"

android:textAppearance="?android:attr/textAppearanceMedium"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingTop="16dp"

android:paddingBottom="8dp"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:id="@+id/nbpacTextView"

android:text="Text"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

</LinearLayout>

[Activity(Label = "DGFIP")]

public class DgfipResourceActivity : Activity

{

public const string RfrExtra = "rfr";

public const string SitfamExtra = "sitfam";

public const string NbpartExtra = "nbpart";

public const string NbpacExtra = "nbpac";

private TextView mRfrTextView;

private TextView mSitfamTextView;

private TextView mNbpartTextView;

private TextView mNbpacTextView;

protected override void OnCreate(Bundle savedInstanceState)

{

base.OnCreate(savedInstanceState);

SetContentView(Resource.Layout.activity\_dgfip\_resource);

mRfrTextView = FindViewById<TextView>(Resource.Id.rfrTextView);

mSitfamTextView = FindViewById<TextView>(Resource.Id.sitfamTextView);

mNbpartTextView = FindViewById<TextView>(Resource.Id.nbpartTextView);

mNbpacTextView = FindViewById<TextView>(Resource.Id.nbpacTextView);

mRfrTextView.Text = Intent.GetStringExtra(RfrExtra);

mSitfamTextView.Text = Intent.GetStringExtra(SitfamExtra);

mNbpartTextView.Text = Intent.GetStringExtra(NbpartExtra);

mNbpacTextView.Text = Intent.GetStringExtra(NbpacExtra);

}

}

1. **Custom** :

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<TextView

android:text="@string/custom\_value\_one"

android:textAppearance="?android:attr/textAppearanceMedium"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingTop="16dp"

android:paddingBottom="8dp"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:id="@+id/valueOneTextView"

android:text="Text"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:text="@string/custom\_value\_two"

android:textAppearance="?android:attr/textAppearanceMedium"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingTop="16dp"

android:paddingBottom="8dp"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

<TextView

android:id="@+id/valueTwoTextView"

android:text="Text"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingRight="16dp"

android:paddingLeft="16dp" />

</LinearLayout>

[Activity(Label = "Custom")]

public class CustomResourceActivity : Activity

{

public const string ValueOneExtra = "value\_one";

public const string ValueTwoExtra = "value\_two";

private TextView mValueOneTextView;

private TextView mValueTwoTextView;

protected override void OnCreate(Bundle savedInstanceState)

{

base.OnCreate(savedInstanceState);

SetContentView(Resource.Layout.activity\_custom\_resource);

mValueOneTextView = FindViewById<TextView>(Resource.Id.valueOneTextView);

mValueTwoTextView = FindViewById<TextView>(Resource.Id.valueTwoTextView);

mValueOneTextView.Text = Intent.GetStringExtra(ValueOneExtra);

mValueTwoTextView.Text = Intent.GetStringExtra(ValueTwoExtra);

}

}

Puis modifiez le code de l’activité principale pour accéder à ces activités :

<Button

android:id="@+id/dgfipResourceButton"

android:layout\_below="@id/pivotIdentityButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingLeft="16dp"

android:paddingRight="16dp"

android:text="@string/dgfip" />

<Button

android:id="@+id/customResourceButton"

android:layout\_below="@id/dgfipResourceButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingLeft="16dp"

android:paddingRight="16dp"

android:text="@string/custom" />

public class MainActivity : Activity

{

...

private Button mDgfipResourceButton;

private Button mCustomResourceButton;

protected override void OnCreate(Bundle bundle)

{

...

mDgfipResourceButton = FindViewById<Button>(Resource.Id.dgfipResourceButton);

mCustomResourceButton = FindViewById<Button>(Resource.Id.customResourceButton);

...

mDgfipResourceButton.Click += DgfipResourceButton\_Click;

mCustomResourceButton.Click += CustomResourceButton\_Click;

}

...

private void ShowHideElements()

{

if (ApplicationUser.IsAuthenticated)

{

mPivotIdentityButton.Visibility = ViewStates.Visible;

mDgfipResourceButton.Visibility = ViewStates.Visible;

mCustomResourceButton.Visibility = ViewStates.Visible;

mLoginLogoutButton.Text = Resources.GetString(Resource.String.logout);

}

else

{

mPivotIdentityButton.Visibility = ViewStates.Gone;

mDgfipResourceButton.Visibility = ViewStates.Gone;

mCustomResourceButton.Visibility = ViewStates.Gone;

mLoginLogoutButton.Text = Resources.GetString(Resource.String.login);

}

}

...

private void DgfipResourceButton\_Click(object sender, EventArgs e)

{

var activity = new Intent(this, typeof(WebViewActivity));

activity.PutExtra(WebViewActivity.JobExtra, (int)WebViewActivity.Job.Consent);

activity.PutExtra(WebViewActivity.ResourceProviderExtra, (int)Provider.DGFIP);

StartActivity(activity);

}

private void CustomResourceButton\_Click(object sender, EventArgs e)

{

var activity = new Intent(this, typeof(WebViewActivity));

activity.PutExtra(WebViewActivity.JobExtra, (int)WebViewActivity.Job.Consent);

activity.PutExtra(WebViewActivity.ResourceProviderExtra, (int)Provider.Custom);

StartActivity(activity);

}

}

Enfin, modifiez le code de la l’activité **WebView** :

public class WebViewActivity : Activity

{

...

private class OidcWebViewClient : WebViewClient

{

...

public string GetStartUrl()

{

switch (\_job)

{

case Job.SignIn:

...

case Job.Consent:

return \_client.CreateConsentUri(\_provider);

default:

throw new NotImplementedException();

}

}

public override bool ShouldOverrideUrlLoading(WebView view, string url)

{

if (url.StartsWith(FranceConnectClient.RedirectUri, StringComparison.OrdinalIgnoreCase))

{

...

}

else if (url.StartsWith(FranceConnectClient.BaseConsentRedirectUri, StringComparison.OrdinalIgnoreCase))

{

view.Visibility = ViewStates.Gone;

var task = Task.Factory.StartNew(async () =>

{

var provider = FranceConnectClient.GetProviderFromRedirectUri(url);

await \_client.AuthenticateAsync(url, provider.GetRedirectUri());

var resource = await \_client.GetDataAsync(provider);

Intent resourceActivity;

switch (provider)

{

case Provider.DGFIP:

resourceActivity = new Intent(\_activity, typeof(DgfipResourceActivity));

resourceActivity.PutExtra(DgfipResourceActivity.RfrExtra, ((DgfipResource)resource).Rfr.ToString("C"));

resourceActivity.PutExtra(DgfipResourceActivity.SitfamExtra, ((DgfipResource)resource).SitFam.ToString());

resourceActivity.PutExtra(DgfipResourceActivity.NbpartExtra, ((DgfipResource)resource).NbPart.ToString());

resourceActivity.PutExtra(DgfipResourceActivity.NbpacExtra, ((DgfipResource)resource).Pac.NbPac.ToString());

\_activity.StartActivity(resourceActivity);

break;

case Provider.Custom:

resourceActivity = new Intent(\_activity, typeof(CustomResourceActivity));

resourceActivity.PutExtra(CustomResourceActivity.ValueOneExtra, ((CustomResource)resource).ValueOne);

resourceActivity.PutExtra(CustomResourceActivity.ValueTwoExtra, ((CustomResource)resource).ValueTwo);

\_activity.StartActivity(resourceActivity);

break;

default:

throw new NotImplementedException();

}

});

task.ContinueWith(t =>

{

\_activity.RunOnUiThread(() =>

{

AlertDialog.Builder alert = new AlertDialog.Builder(\_activity);

alert.SetTitle("Erreur");

alert.SetMessage(t.Exception.Message);

alert.SetPositiveButton("Fermer", (senderAlert, args) =>

{

\_activity.Finish();

});

Dialog dialog = alert.Create();

dialog.Show();

});

},

TaskContinuationOptions.OnlyOnFaulted);

}

return base.ShouldOverrideUrlLoading(view, url);

}

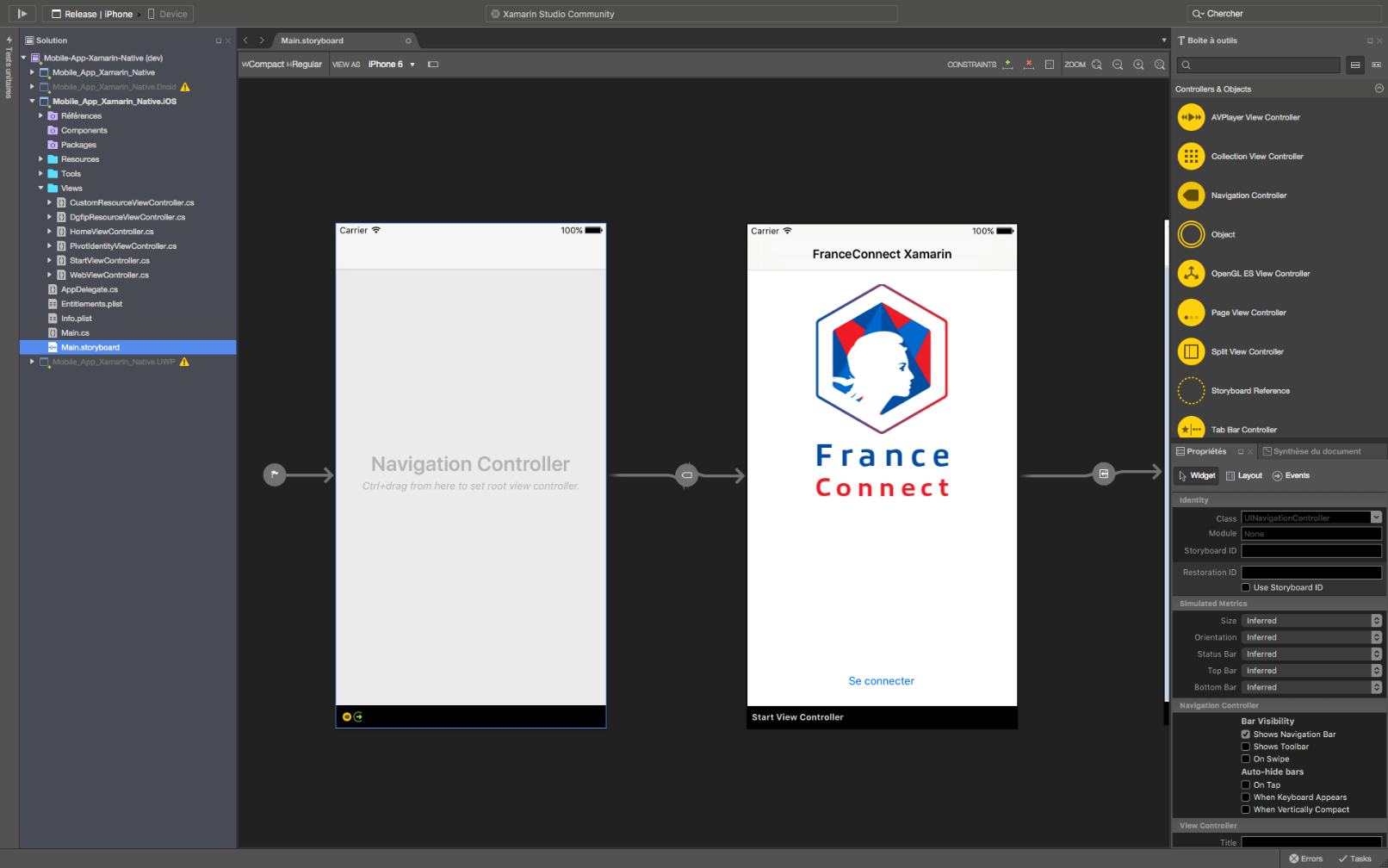
}

}

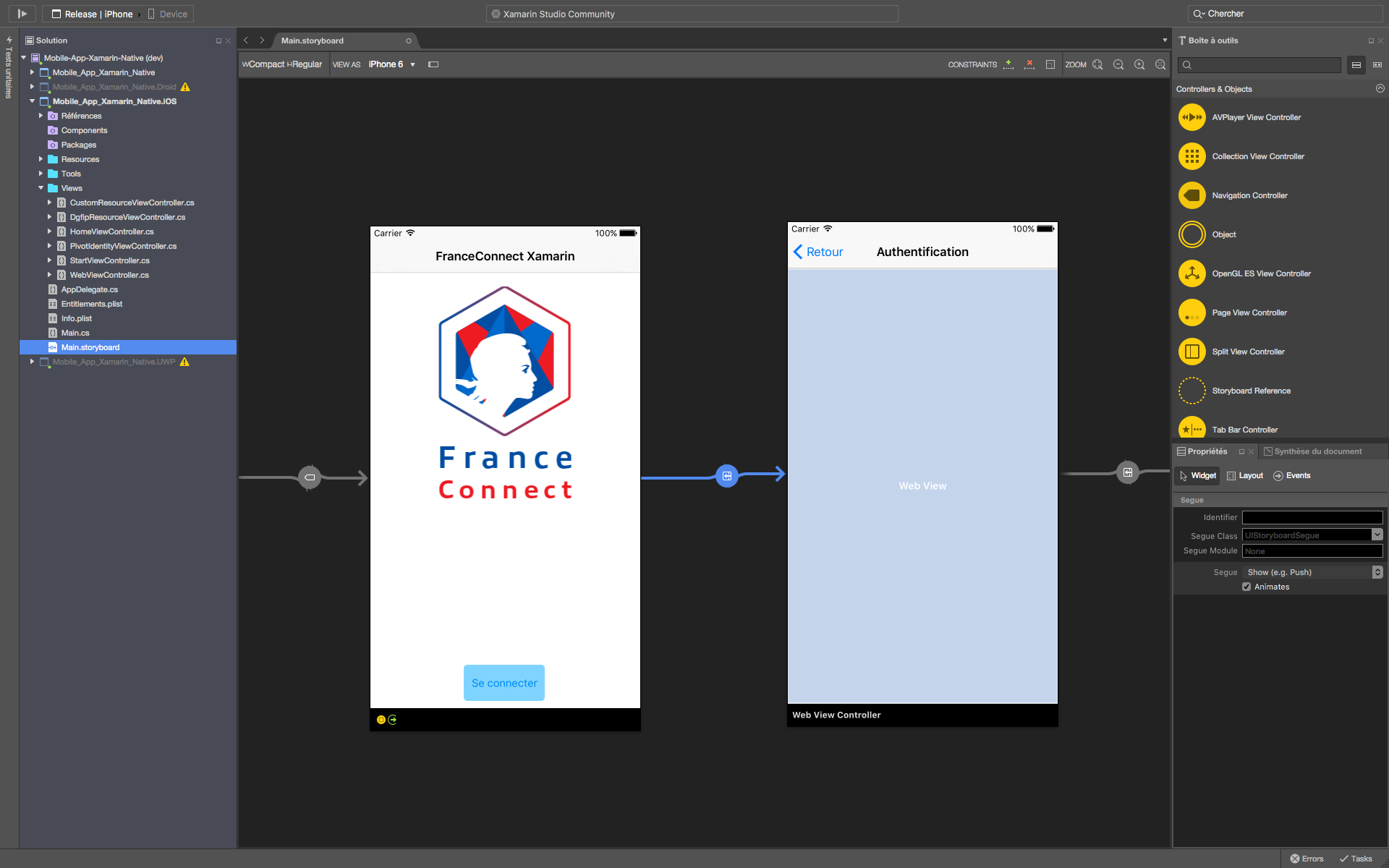
## Créer une application iOS

### Storyboard

Modifiez le storyboard principal, **Main.Storyboard** pour ajouter un Navigation Controller et une vue de démarrage :



Ajoutez une nouvelle vue pour l’authentification et ajoutez le contrôleur de vue correspondant **WebViewController**. Créez un bouton **Se connecter** pour rediriger l’utilisateur sur la vue d’authentification :



Ajoutez également une WebView à la vue d’authentification.

Modifiez le code du **WebViewController** de la façon suivante :

public partial class WebViewController : UIViewController

{

private FranceConnectClient \_client;

const string AuthorizeToHomeSegue = "AuthorizeToHome";

public WebViewController(IntPtr handle) : base(handle)

{

\_client = new FranceConnectClient();

}

public override void ViewDidLoad()

{

base.ViewDidLoad();

// Perform any additional setup after loading the view, typically from a nib.

var uri = new NSUrl(\_client.CreateSignInUri());

CookieHelper.ClearCookies();

WebView.LoadRequest(new NSUrlRequest(uri));

WebView.ShouldStartLoad += (webView, request, navigationType) =>

{

if (request.Url.AbsoluteString.StartsWith(FranceConnectClient.RedirectUri, StringComparison.OrdinalIgnoreCase))

{

var task = Task.Factory.StartNew(async () =>

{

await \_client.AuthenticateAsync(request.Url.AbsoluteString, FranceConnectClient.RedirectUri);

ApplicationUser.AuthenticateUser(await \_client.GetUserInfoAsync());

});

task.Unwrap().ContinueWith((antecedent) =>

{

InvokeOnMainThread(() =>

{

PerformSegue(AuthorizeToHomeSegue, this);

});

},

TaskContinuationOptions.OnlyOnRanToCompletion);

task.Unwrap().ContinueWith((antecedent) =>

{

InvokeOnMainThread(() =>

{

var alert = UIAlertController.Create("Erreur", "Une erreur est survenue", UIAlertControllerStyle.Alert);

alert.AddAction(UIAlertAction.Create("Ok", UIAlertActionStyle.Cancel, null));

PresentViewController(alert, animated: true, completionHandler: null);

});

},

TaskContinuationOptions.OnlyOnFaulted);

return false;

}

return true;

};

}

public override void PrepareForSegue(UIStoryboardSegue segue, NSObject sender)

{

base.PrepareForSegue(segue, sender);

}

public override void DidReceiveMemoryWarning()

{

base.DidReceiveMemoryWarning();

// Release any cached data, images, etc that aren't in use.

}

}

public static class CookieHelper

{

public static void ClearCookies()

{

var storage = NSHttpCookieStorage.SharedStorage;

foreach (var cookie in storage.Cookies)

{

storage.DeleteCookie(cookie);

}

}

}

Par défaut, iOS 9 refuse toute requête vers une URL non sécurisé (non HTTPS). Si le fournisseur de données ou le fournisseur d’identité que vous utilisez n’est pas configuré pour l’HTTPS, ajoutez les lignes suivantes au fichier Info.plist :

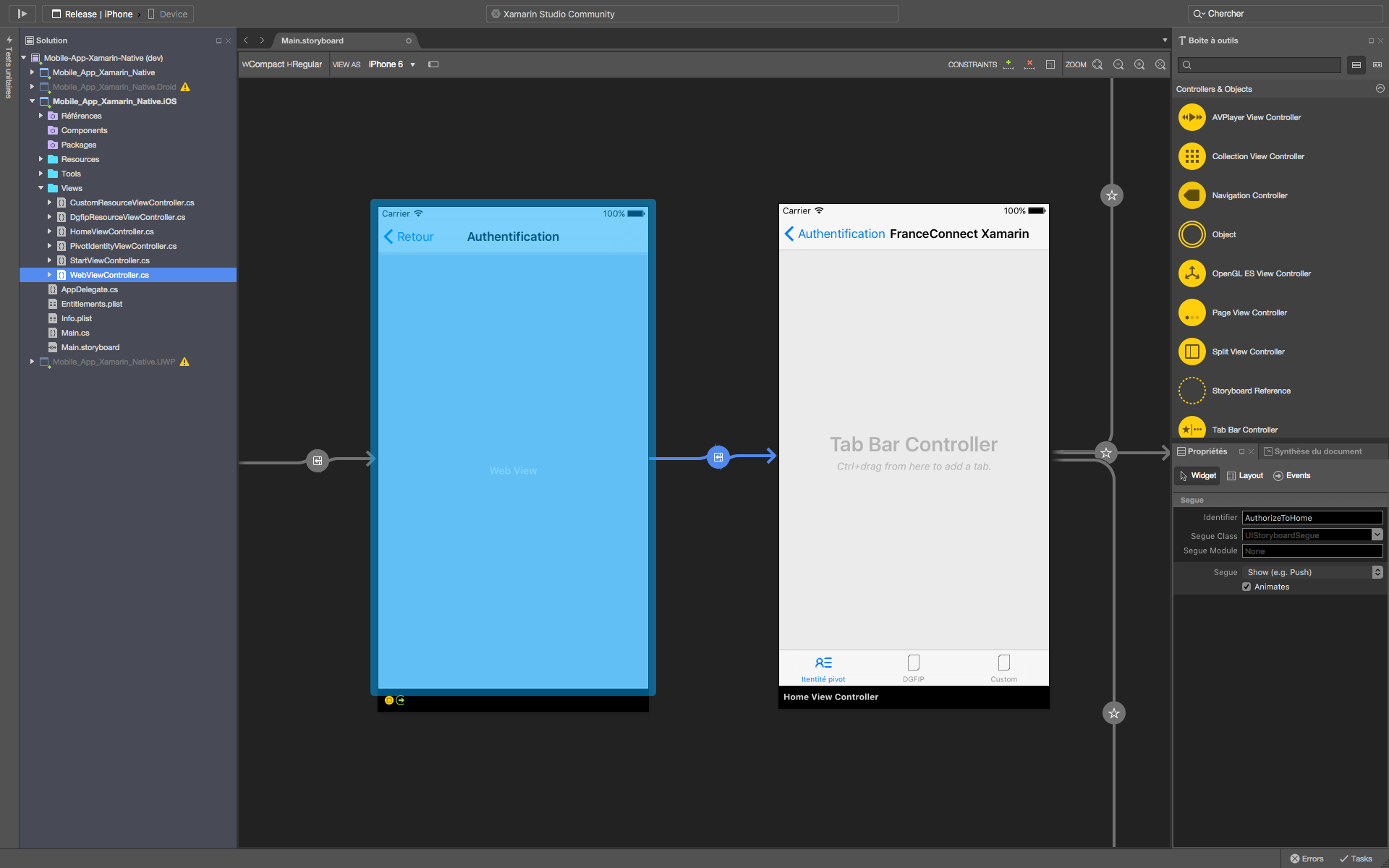
<dict>

<key>NSAllowsArbitraryLoads</key>

<true/>

</dict>

Ajoutez un TabBarController qui sera la vue principale de l’application et créez un Segue qui relie la vue d’authentification à cette vue que vous nommerez **AuthorizeToHome**.



### Vue principale

Liez la vue principale au contrôleur de vue **HomeViewController** et modifiez le code de la façon suivante :

public partial class HomeViewController : UITabBarController

{

public HomeViewController(IntPtr handle) : base(handle)

{

}

public override void ViewDidLoad()

{

base.ViewDidLoad();

// Perform any additional setup after loading the view, typically from a nib.

NavigationItem.HidesBackButton = true;

NavigationItem.RightBarButtonItem = new UIBarButtonItem(UIBarButtonSystemItem.Stop, (sender, args) =>

{

var alert = UIAlertController.Create("Se déconnecter", "Voulez vous vraiment vous déconnecter ?", UIAlertControllerStyle.Alert);

alert.AddAction(UIAlertAction.Create("Non", UIAlertActionStyle.Default, null));

alert.AddAction(UIAlertAction.Create("Oui", UIAlertActionStyle.Destructive, p =>

{

ApplicationUser.SignOut();

NavigationController.PopToRootViewController(true);

}));

PresentViewController(alert, animated: true, completionHandler: null);

});

}

public override void DidReceiveMemoryWarning()

{

base.DidReceiveMemoryWarning();

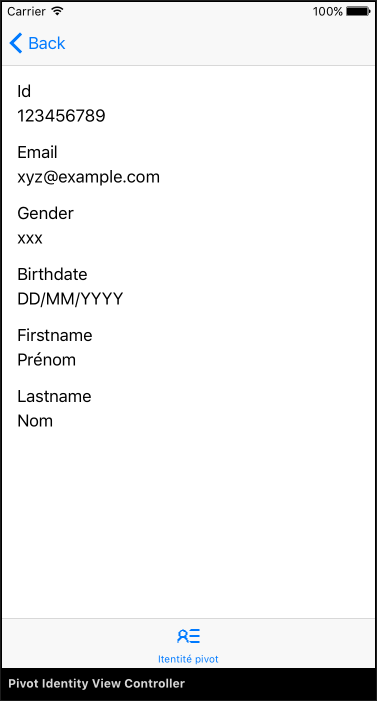
// Release any cached data, images, etc that aren't in use.

}

}

### Afficher l’identité pivot

Ajoutez une TabView pour afficher l’identité pivot. Ajoutez également le contrôleur de vue correspondant **PivotIdentityViewController** :



public partial class PivotIdentityViewController : UIViewController

{

public PivotIdentityViewController(IntPtr handle) : base(handle)

{

}

public override void ViewDidLoad()

{

base.ViewDidLoad();

// Perform any additional setup after loading the view, typically from a nib.

var user = ApplicationUser.GetAuthenticatedUser();

IdLabel.Text = user.Id;

GenderLabel.Text = user.Gender;

EmailLabel.Text = user.Email;

BirthdateLabel.Text = string.Format("{0:D}", user.Birthdate);

FirstnameLabel.Text = user.Firstname;

LastnameLabel.Text = user.Lastname;

}

public override void DidReceiveMemoryWarning()

{

base.DidReceiveMemoryWarning();

// Release any cached data, images, etc that aren't in use.

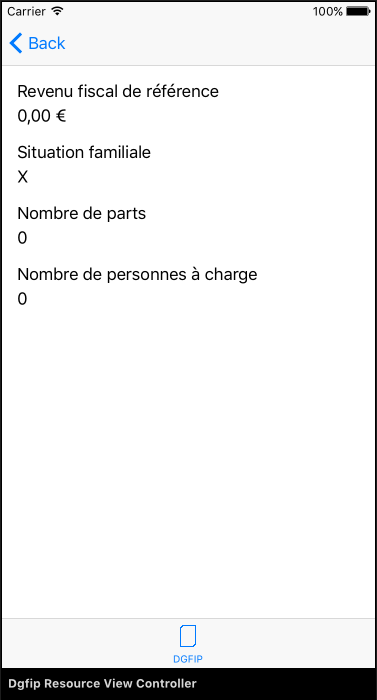
}

}

### Obtenir des données depuis un fournisseur de données

Créez deux TabView pour les données de la DGFIP et celles du fournisseur de données Custom :

1. **DGFIP** :



public partial class DgfipResourceViewController : UIViewController

{

private FranceConnectClient \_client;

public DgfipResourceViewController(IntPtr handle) : base(handle)

{

\_client = new FranceConnectClient();

}

public override void ViewDidLoad()

{

base.ViewDidLoad();

// Perform any additional setup after loading the view, typically from a nib.

var consentWebView = new UIWebView(View.Bounds);

View.AddSubview(consentWebView);

var url = \_client.CreateConsentUri(Provider.DGFIP);

consentWebView.LoadRequest(new NSUrlRequest(new NSUrl(url)));

consentWebView.ShouldStartLoad += (webView, request, navigationType) =>

{

if (request.Url.AbsoluteString.StartsWith(FranceConnectClient.BaseConsentRedirectUri, StringComparison.OrdinalIgnoreCase))

{

var task = Task.Factory.StartNew(async () =>

{

await \_client.AuthenticateAsync(request.Url.AbsoluteString, Provider.DGFIP.GetRedirectUri());

var resource = await \_client.GetDataAsync(Provider.DGFIP) as DgfipResource;

InvokeOnMainThread(() =>

{

RfrLabel.Text = string.Format("{0:C}", resource.Rfr);

SitFamLabel.Text = resource.SitFam.ToString();

NbPartLabel.Text = resource.NbPart.ToString();

NbPacLabel.Text = resource.Pac.NbPac.ToString();

consentWebView.RemoveFromSuperview();

});

});

task.Unwrap().ContinueWith((antecedent) =>

{

InvokeOnMainThread(() =>

{

var alert = UIAlertController.Create("Erreur", "Une erreur est survenue", UIAlertControllerStyle.Alert);

alert.AddAction(UIAlertAction.Create("Ok", UIAlertActionStyle.Cancel, null));

PresentViewController(alert, animated: true, completionHandler: null);

});

},

TaskContinuationOptions.OnlyOnFaulted);

return false;

}

return true;

};

}

public override void DidReceiveMemoryWarning()

{

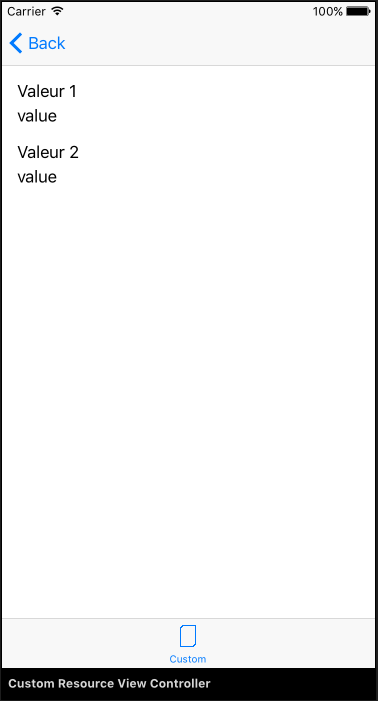
base.DidReceiveMemoryWarning();

// Release any cached data, images, etc that aren't in use.

}

}

1. **Custom** :



public partial class CustomResourceViewController : UIViewController

{

private FranceConnectClient \_client;

public CustomResourceViewController(IntPtr handle) : base(handle)

{

\_client = new FranceConnectClient();

}

public override void ViewDidLoad()

{

base.ViewDidLoad();

// Perform any additional setup after loading the view, typically from a nib.

var consentWebView = new UIWebView(View.Bounds);

View.AddSubview(consentWebView);

var url = \_client.CreateConsentUri(Provider.Custom);

consentWebView.LoadRequest(new NSUrlRequest(new NSUrl(url)));

consentWebView.ShouldStartLoad += (webView, request, navigationType) =>

{

if (request.Url.AbsoluteString.StartsWith(FranceConnectClient.BaseConsentRedirectUri, StringComparison.OrdinalIgnoreCase))

{

var task = Task.Factory.StartNew(async () =>

{

await \_client.AuthenticateAsync(request.Url.AbsoluteString, Provider.Custom.GetRedirectUri());

var resource = await \_client.GetDataAsync(Provider.Custom) as CustomResource;

InvokeOnMainThread(() =>

{

ValueOneLabel.Text = resource.ValueOne;

ValueTwolabel.Text = resource.ValueTwo;

consentWebView.RemoveFromSuperview();

});

});

task.Unwrap().ContinueWith((antecedent) =>

{

InvokeOnMainThread(() =>

{

var alert = UIAlertController.Create("Erreur", "Une erreur est survenue", UIAlertControllerStyle.Alert);

alert.AddAction(UIAlertAction.Create("Ok", UIAlertActionStyle.Cancel, null));

PresentViewController(alert, animated: true, completionHandler: null);

});

},

TaskContinuationOptions.OnlyOnFaulted);

return false;

}

return true;

};

}

public override void DidReceiveMemoryWarning()

{

base.DidReceiveMemoryWarning();

// Release any cached data, images, etc that aren't in use.

}

}

# Déployer le canevas d’application mobile préconfiguré du kit de démarrage

## imageObtenir les sources de l’application mobile

Le kit de démarrage FranceConnect est disponible avec le code source des différents exemples/accélérateurs ainsi proposés sur le repo/la forge communautaire GitHub à l’adresse <https://github.com/FranceConnectSamples>.

Vous trouverez à cette adresse notamment le projet relatif à l’exemple/accélérateur d’application mobile : [franceconnect-mobile-app-xamarin-native](https://github.com/FranceConnectSamples/franceconnect-mobile-app-xamarin-native)[[9]](#footnote-9).

Les solutions communautaires évoluant avec l’investissement de la communauté, il est souhaitable de vous assurer que vous disposez de la dernière version, ne serait-ce que pour bénéficier, le cas échéant, de l’ensemble des correctifs.

La disponibilité du projet comme repo GitHub signifie que vous pouvez non seulement obtenir le package relatif au projet avec le code source correspondant à compiler mais également cloner le repo Git, accéder au code source, le compiler le code mais aussi soumettre des demandes de type Pull (pull request) à l’instar de n’importe quel autre package open source vous pourriez trouver sur GitHub et, ce afin de proposer et/ou de collaborer sur des évolutions au niveau du repo.

Les deux prochaines sections explorent les deux options possibles.

### Télécharger les sources de l’application mobile

Pour télécharger les sources depuis la forge GitHub, le moyen le plus simple et rapide est de télécharger le package complet sous forme de fichier archive .zip.

**Pour cela, procédez comme suit :**

1. **Ouvrez une session browser et naviguer vers le projet** [franceconnect-mobile-app-xamarin-native](https://github.com/FranceConnectSamples/franceconnect-mobile-app-xamarin-native)**.**
2. **Cliquez sur Clone or Download.**

****

1. **Cliquez sur Download ZIP.**
2. **Enregistrez le fichier *franceconnect-mobile-app-xamarin-native-master.zip* en local sur votre machine.**
3. **Extrayez le contenu du fichier *franceconnect-mobile-app-xamarin-native-master.zip* sur un disque local, par exemple sous le dossier *C:\Code*.**

**Le code source du canevas d’application mobile FranceConnect est alors situé sous le dossier *C:\Code\franceconnect-mobile-app-xamarin-native-master\Source.***

Si cette approche constitue le moyen le plus simple, la seconde option abordée ci-après se révèle plus adaptée pour mettre en place une gestion de versions.

### Cloner les sources de l’application mobile

 Comme décrit précédemment, le kit de démarrage FranceConnect est disponible sur le repo **GitHub**, qui repose sur **Git** comme système de contrôle de code source.

Pour accéder au package du projet et au code source correspondant, il est préférable dans cette seconde approche de disposer d’une connaissance préalable de [Git](http://git-scm.com/)[[10]](#footnote-10), [GitHub](https://github.com/FranceConnectSamples)[[11]](#footnote-11) ou [Visual Studio](https://msdn.microsoft.com/vstudio/aa718325.aspx)[[12]](#footnote-12). Ceci étant, les étapes suivantes fournissent des informations et des liens pour vous aider à démarrer.

Note Pour plus d’informations sur la façon de configurer Git et GitHub, consultez l’article [Set up Git](https://help.github.com/articles/set-up-git/)[[13]](#footnote-13) sur le site de GitHub.

Pour accéder au package source du projet et contribuer davantage à celui, vous devez i) créer une branche (fork) sur le repo Git qui le contient et ii) cloner cette branche sur votre machine de développement.

Pour créer un fork sur le repo Git, procédez comme suit :

1. **Ouvrez une session browser et naviguer vers le projet** [franceconnect-mobile-app-xamarin-native](https://github.com/FranceConnectSamples/franceconnect-mobile-app-xamarin-native).
2. Cliquez sur **Fork** dans le coin supérieur droit la fenêtre de votre browser.

Vous pouvez dès lors cloner la branche à l’aide de l’application GitHub ou via la ligne de commande dans le Shell Git.

Pour cloner le repo avec le Shell Git, procédez comme suit :

1. Ouvrez un Shell Git Shell en double-cliquant sur l’icône éponyme sur votre bureau Windows.
2. Depuis l’invite de commande, tapez la commande suivante :

C:\Users\philber\Documents\GitHub>Copygit clone https://github.com/FranceConnectSamples/franceconnect-mobile-app-xamarin-native

Le clonage démarre.

C:\Users\philber\Documents\GitHub>Copygit clone https://github.com/FranceConnectSamples/franceconnect-mobile-app-xamarin-native

Cloning into 'franceconnect-mobile-app-xamarin-native'...

Le package du canevas d’application mobile FranceConnect est disponible sous le dossier *franceconnect-mobile-app-xamarin-native* situé sous *%UserProfile%\Documents\GitHub*.

**Le code source associé est alors situé sous le dossier** *%UserProfile%\Documents\GitHub****\franceconnect-mobile-app-xamarin-native-master\Source.***

L’environnement Visual Studio Community 2015 propose également une intégration directe avec GitHub que nous ne développons pas ici.

Quelle que soit l’approche retenue, et une fois, le code source téléchargée, nous vous invitons à passer à la section suivante.

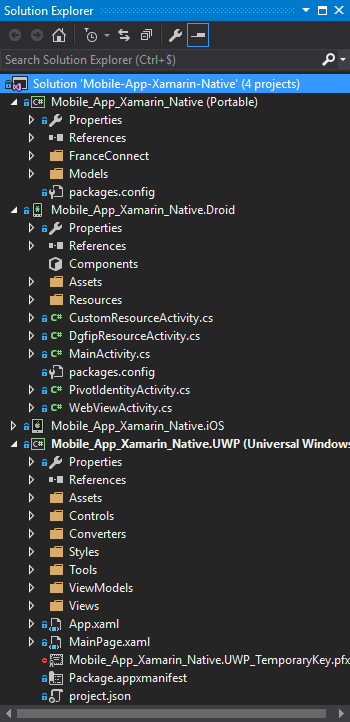
Remarque importante Nous faisons référence au dossier où se situe le code source du canevas d’application mobile comme étant le dossier Kit-Demarrage dans la suite de ce document.

## Organisation du code source du canevas de l’application mobile

A l’instar de ce qui est décrit dans la section § [*Créer une application mobile FranceConnect avec Xamarin*](#_Créer_un_fournisseur), la solution préconfigurée prête à l’emploi est constituée de quatre projet de type **Portable, Android, iOS et Universal Windows**. Ouvrez le fichier solution Visual Studio .sln depuis le dossier **Kit-Demarrage***.*

### Comprendre l’organisation du projet

Le canevas d’application mobile proposé par la solution Visual Studio se compose de la solution suivante :



Architecture du projet

### Appréhender les éléments de configuration

La configuration FranceConnect de l’application se trouve dans le fichier *FranceConnectClient.cs* du projet portable. La configuration des fournisseurs de données se trouve dans le fichier *Provider* présent dans le même projet.

*FranceConnectClient.cs :*

public const string RedirectUri = "http://Mobile-App-Xamarin-Native/signin\_oidc";

public const string BaseConsentRedirectUri = "http://Mobile-App-Xamarin-Native/consent/callback\_";

private const string AuthorizationEndpoint = "https://fcp.integ01.dev-franceconnect.fr/api/v1/authorize";

private const string TokenEndpoint = "https://fcp.integ01.dev-franceconnect.fr/api/v1/token";

private const string UserInfoEndpoint = "https://fcp.integ01.dev-franceconnect.fr/api/v1/userinfo";

private const string ClientId = "dd3d652f1be391d4ae3a055ecd6bce8e5ae7bf1c2fa80539a03c1b180cbc6b47";

private const string ClientSecret = "e359cd837155fb12d0999ef23337113454532198dd26604a85db8c7b51544b9a";

*Provider.cs :*

public static List<string> GetRequiredScope(this Provider provider)

{

switch (provider)

{

case Provider.DGFIP: return new List<string> { "dgfip\_rfr", "dgfip\_sitfam", "dgfip\_nbpac", "dgfip\_nbpart" };

case Provider.Custom: return new List<string> { "value1", "value2" };

default: throw new NotImplementedException();

}

}

public static Uri GetEndpoint(this Provider provider)

{

switch (provider)

{

case Provider.DGFIP: return new Uri("https://fdp.integ01.dev-franceconnect.fr/courtier/apiuniverselle/2016");

case Provider.Custom: return new Uri("http://franceconnect-data-provider-dotnet-webapi-aspnetcore.azurewebsites.net/api/values");

default: throw new NotImplementedException();

}

}

## Publier l’application sur les stores

Pour publier l’application, procédez comme suit :

1. Sélectionnez l’application web que vous venez de créer.
2. Cliquez sur **Publish**

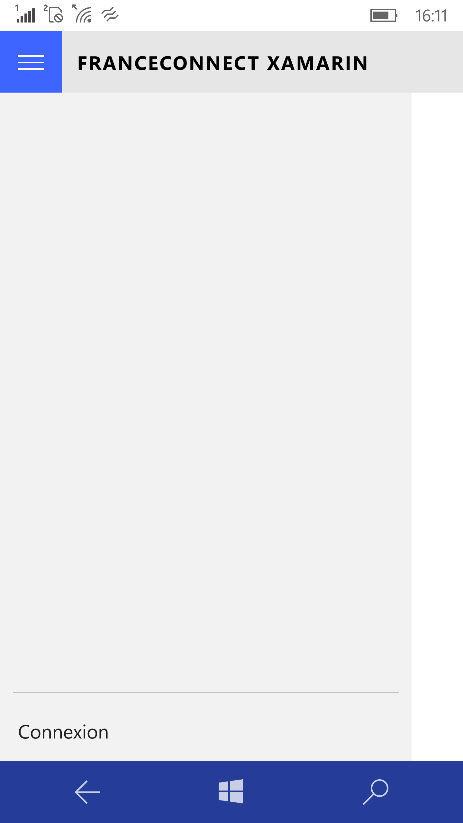
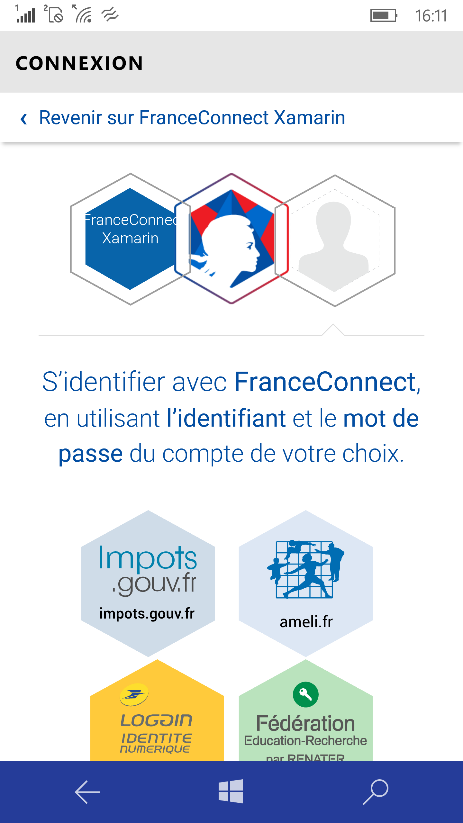
# Tester l’application mobile

## Configurer le compte FranceConnect

Vous êtes obligé de renseigner vos URL de callback pour que l’authentification via FranceConnect fonctionne.

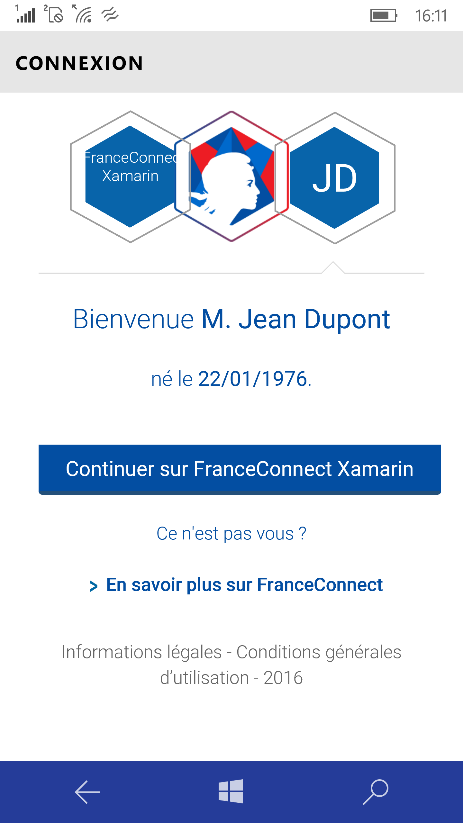
1. Rendez-vous sur [votre compte FranceConnect](https://doc.integ01.dev-franceconnect.fr/client)
2. Ajoutez les URL de callback suivantes :
   1. <http://Mobile-App-Xamarin-Native/signin_oidc>
   2. http://Mobile-App-Xamarin-Native/consent/callback\_<*nom du FD*> (pour chaque fournisseur de données que vous utilisez)

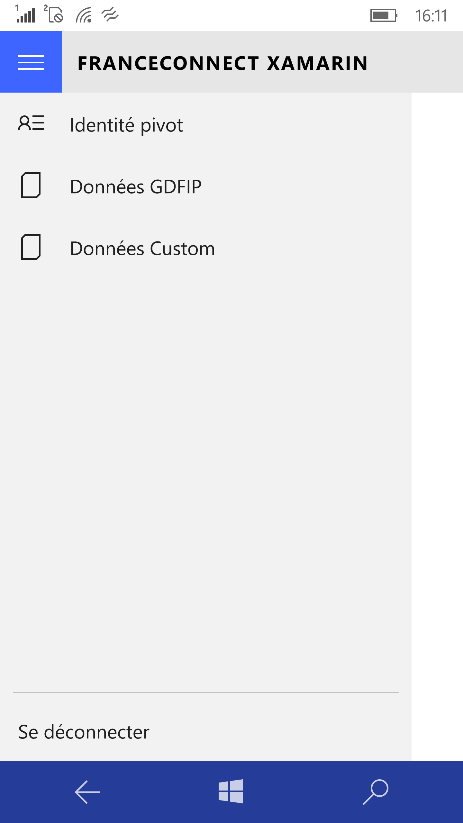
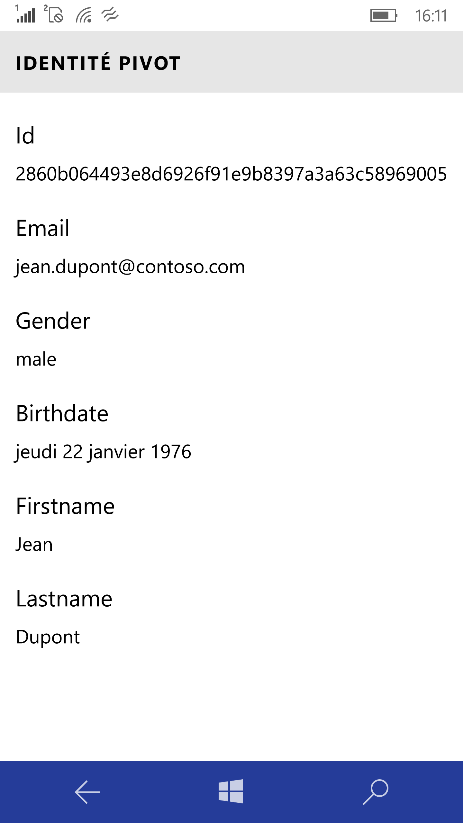
## Se connecter avec le système FranceConnect

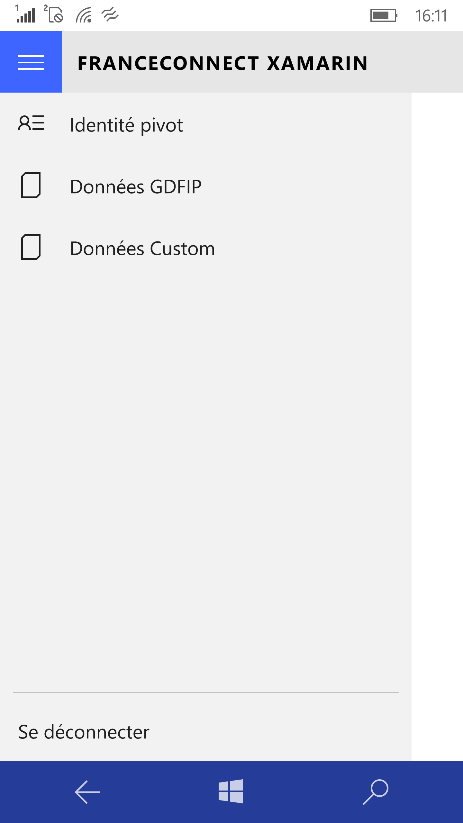
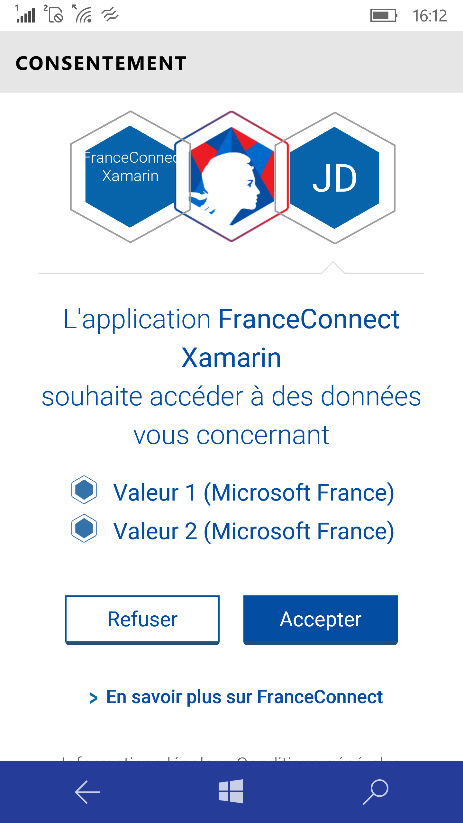
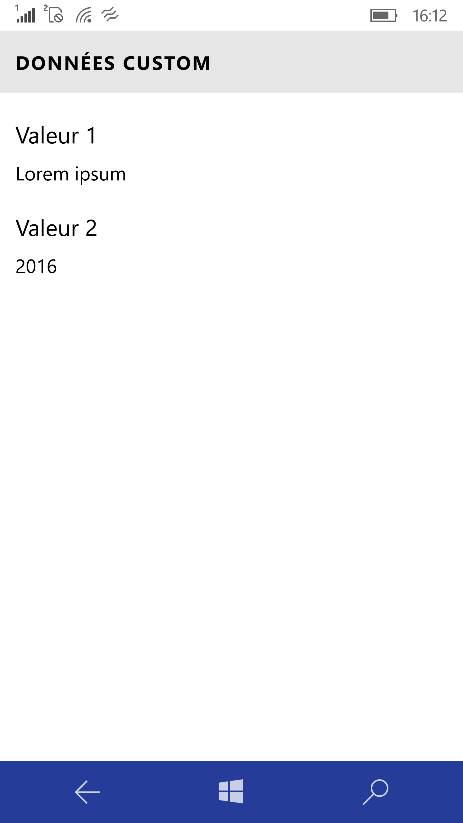
Pour le fournisseur d’identité factice Impôts (utilisateur Jean Dupont)

* Numéro fiscal : 248163264128
* Mot de passe : 123

## Invoquer un service de données via le système FranceConnect

Ceci conclut ce guide.

# Annexe A. Références techniques

Le document présent s’appuie sur les diverses ressources d’intégration mises à disposition par la DINSIC et en particulier de la [documentation technique d’intégration](https://doc.integ01.dev-franceconnect.fr)[[14]](#footnote-14).

Il repose plus particulièrement pour la définition des éléments de mise en œuvre d’application mobile FranceConnect ainsi proposé sur la [documentation](https://doc.integ01.dev-franceconnect.fr/fournisseur-service)[[15]](#footnote-15).

1. Site portail Web Microsoft Secteur Public : http://www.microsoft.com/france/entreprises/secteur-public/solutions-pour-secteur-public.aspx [↑](#footnote-ref-1)
2. Xamarin.mobile : http://www.dotnetfoundation.org/xamarin.mobile [↑](#footnote-ref-2)
3. Fondation .NET : http://www.dotnetfoundation.org/ [↑](#footnote-ref-3)
4. Visual Studio Community 2015 : https://go.microsoft.com/fwlink/?LinkId=691978&clcid=0x40c [↑](#footnote-ref-4)
5. Introduction to ASP.NET Core : https://docs.asp.net/en/latest/conceptual-overview/aspnet.html [↑](#footnote-ref-5)
6. ASP.NET Core Documentation : https://docs.asp.net/en/latest/ [↑](#footnote-ref-6)
7. franceconnect-mobile-app-xamarin-native : https://github.com/FranceConnectSamples/franceconnect-mobile-app-xamarin-native [↑](#footnote-ref-7)
8. Inscription FranceConnect - Fournisseur de service : https://doc.integ01.dev-franceconnect.fr/inscription [↑](#footnote-ref-8)
9. franceconnect-mobile-app-xamarin-native : https://github.com/FranceConnectSamples/franceconnect-mobile-app-xamarin-native [↑](#footnote-ref-9)
10. Git : http://git-scm.com/ [↑](#footnote-ref-10)
11. GitHub : https://github.com/FranceConnectSamples [↑](#footnote-ref-11)
12. Visual Studio : https://msdn.microsoft.com/vstudio/aa718325.aspx [↑](#footnote-ref-12)
13. Set up Git : https://help.github.com/articles/set-up-git/ [↑](#footnote-ref-13)
14. FranceConnect : https://doc.integ01.dev-franceconnect.fr/ [↑](#footnote-ref-14)
15. Documentation – Fournisseur de service : https://doc.integ01.dev-franceconnect.fr/fournisseur-service [↑](#footnote-ref-15)