Créer un client WPF C# pour se connecter à une API REST - Projet Restaurant

Guide complet pour créer une application WPF qui se connecte à une API REST. Apprenez à créer des modèles, des services API, centraliser l'accès HTTP et créer une interface utilisateur fonctionnelle.

Programmation Laravel 25 min de lecture Niveau Intermédiaire

Document généré le 25/05/2026 à 20h13 · nouvy.fr/wiki/wpf-client-api-rest-csharp-restaurant

Sommaire

27 section(s) · 25 min de lecture

Ce guide vous apprendra à créer une **application WPF** qui se connecte à une **API REST** créée avec ASP.NET Core. Nous utiliserons le projet de gestion de restaurant comme exemple pratique pour créer un client desktop.

□ Objectifs

À la fin de ce guide, vous serez capable de :

- □ Créer un projet WPF dans Visual Studio
- □ Créer des modèles de données correspondant à l'API
- □ Créer des services pour communiquer avec l'API REST
- □ Centraliser l'accès HTTP pour éviter la duplication de code
- □ Créer une interface utilisateur XAML
- □ Utiliser les services API dans le code-behind
- □ Gérer les opérations CRUD via l'API

□ Prérequis

- **Visual Studio** (2019 ou supérieur) avec le workload .NET Desktop Development
- **API REST** fonctionnelle (voir l'article sur la création d'une API web C#)
- **.NET SDK** (version 8.0)
- Connaissances de base en C# et XAML

Étape 1 : Création du projet WPF

1. Ouvrez **Visual Studio**
2. Cliquez sur **"Créer un nouveau projet"**
3. Sélectionnez le modèle **"Application WPF (.NET)"** ou **"Application WPF"**
4. Configurez votre projet :
    - **Nom du projet** : `RestaurantWPFClient` (ou le nom de votre choix)
    - **Framework** : .NET 8.0
    - **Emplacement** : Choisissez votre dossier de projet
5. Cliquez sur **"Créer"**

Votre projet WPF est maintenant créé avec la structure de base.

Étape 2 : Création des modèles de données

Les modèles doivent correspondre exactement aux modèles de votre API pour que la sérialisation/désérialisation JSON fonctionne correctement.

Création du dossier Models

Créez un dossier **`Models`** dans votre projet (clic droit sur le projet > `Ajouter` > `Nouveau dossier`).

Modèle Article

Créez une nouvelle classe C# dans le dossier `Models` et nommez-la `Article.cs` :

```
using System;
using System.Collections.Generic;

namespace RestaurantWPFClient.Models
{
    public class Article
    {
        public int Id { get; set; }
        public string Nom { get; set; }
        public float PrixHT { get; set; }
        public float PrixTTC { get; set; }
        public float TauxTva { get; set; }
    }
}
```

Copier

Modèle Menu

Créez `Menu.cs` :

```
using System;
using System.Collections.Generic;

namespace RestaurantWPFClient.Models
{
    public class Menu
    {
        public int Id { get; set; }
        public string Nom { get; set; }
        public float PrixHT { get; set; }
        public float PrixTTC { get; set; }
        public float TauxTva { get; set; }
        public List<Article> Articles { get; set; }
    }
}
```

Copier

Modèle Commande

Créez `Commande.cs` :

```
using System;
using System.Collections.Generic;

namespace RestaurantWPFClient.Models
{
    public class Commande
    {
        public int Id { get; set; }
        public string Numero { get; set; }
        public float TotalHT { get; set; }
        public float TotalTTC { get; set; }
        public List<Menu> Menus { get; set; }
        public List<Article> Articles { get; set; }
        public int TableId { get; set; }
        public int UtilisateurId { get; set; }
    }
}
```

Copier

Modèle Table

Créez `Table.cs` :

```
using System;
using System.Collections.Generic;

namespace RestaurantWPFClient.Models
{
    public class Table
    {
        public int Id { get; set; }
        public int Numero { get; set; }
        public int Places { get; set; }
    }
}
```

Copier

Modèle Utilisateur

Créez `Utilisateur.cs` :

```
using System;
using System.Collections.Generic;

namespace RestaurantWPFClient.Models
{
    public class Utilisateur
    {
        public int Id { get; set; }
        public string Nom { get; set; }
        public string Prenom { get; set; }
        public string Email { get; set; }
        public string Password { get; set; }
    }
}
```

Copier

> 💡 ***Note*** *: Les modèles doivent correspondre exactement aux modèles de votre API pour que la sérialisation JSON fonctionne correctement.*

Étape 3 : Installation du package HttpClient

Pour communiquer avec l'API REST, nous avons besoin du package `System.Net.Http.Json`. Il est généralement inclus dans les projets .NET modernes, mais si nécessaire, vous pouvez l'ajouter :

```
dotnet add package System.Net.Http.Json
```

Copier

Étape 4 : Création des services API

Création du dossier Services

Créez un dossier **`Services`** dans votre projet (clic droit sur le projet > `Ajouter` > `Nouveau dossier`).

Centralisation de l'accès API

Pour éviter de répéter l'URL de base dans chaque service, créons un client API centralisé.

Créez `ApiClient.cs` dans le dossier `Services` :

```
using System;
using System.Net.Http;

namespace RestaurantWPFClient.Services
{
    public class ApiClient
    {
        private static readonly HttpClient _httpClient = new HttpClient
        {
            BaseAddress = new Uri('https://localhost:7181/api/')
        };

        public static HttpClient HttpClient => _httpClient;
    }
}
```

Copier

> ⚠ ***Important*** *: Remplacez* `https://localhost:7181/api/` *par l'URL de votre API. Vous pouvez également utiliser un fichier de configuration (*`app.config` *ou* `appsettings.json`*) pour stocker cette URL.*

Service ApiArticleService

Créez `ApiArticleService.cs` dans le dossier `Services` :

```
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Net.Http.Json;
using System.Threading.Tasks;
using RestaurantWPFClient.Models;

namespace RestaurantWPFClient.Services
{
    public class ApiArticleService
    {
        private readonly HttpClient _httpClient;

        public ApiArticleService()
        {
            _httpClient = ApiClient.HttpClient;
        }

        public async Task<List<Article>> GetArticlesAsync()
        {
            return await _httpClient.GetFromJsonAsync<List<Article>>('Article');
        }

        public async Task<Article> GetArticleByIdAsync(int id)
        {
            return await _httpClient.GetFromJsonAsync<Article>($'Article/{id}');
        }

        public async Task AddArticleAsync(Article article)
        {
            var response = await _httpClient.PostAsJsonAsync('Article', article);
            response.EnsureSuccessStatusCode();
        }

        public async Task UpdateArticleAsync(int id, Article article)
        {
            var response = await _httpClient.PutAsJsonAsync($'Article/{id}', article);
            response.EnsureSuccessStatusCode();
        }

        public async Task DeleteArticleAsync(int id)
        {
            var response = await _httpClient.DeleteAsync($'Article/{id}');
            response.EnsureSuccessStatusCode();
        }
    }
}
```

Copier

Service ApiMenuService

Créez `ApiMenuService.cs` :

```
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Net.Http.Json;
using System.Threading.Tasks;
using RestaurantWPFClient.Models;

namespace RestaurantWPFClient.Services
{
    public class ApiMenuService
    {
        private readonly HttpClient _httpClient;

        public ApiMenuService()
        {
            _httpClient = ApiClient.HttpClient;
        }

        public async Task<List<Menu>> GetMenusAsync()
        {
            return await _httpClient.GetFromJsonAsync<List<Menu>>('Menu');
        }

        public async Task<Menu> GetMenuByIdAsync(int id)
        {
            return await _httpClient.GetFromJsonAsync<Menu>($'Menu/{id}');
        }

        public async Task AddMenuAsync(Menu menu)
        {
            var response = await _httpClient.PostAsJsonAsync('Menu', menu);
            response.EnsureSuccessStatusCode();
        }

        public async Task UpdateMenuAsync(int id, Menu menu)
        {
            var response = await _httpClient.PutAsJsonAsync($'Menu/{id}', menu);
            response.EnsureSuccessStatusCode();
        }

        public async Task DeleteMenuAsync(int id)
        {
            var response = await _httpClient.DeleteAsync($'Menu/{id}');
            response.EnsureSuccessStatusCode();
        }
    }
}
```

Copier

Service ApiCommandeService

Créez `ApiCommandeService.cs` :

```
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Net.Http.Json;
using System.Threading.Tasks;
using RestaurantWPFClient.Models;

namespace RestaurantWPFClient.Services
{
    public class ApiCommandeService
    {
        private readonly HttpClient _httpClient;

        public ApiCommandeService()
        {
            _httpClient = ApiClient.HttpClient;
        }

        public async Task<List<Commande>> GetCommandesAsync()
        {
            return await _httpClient.GetFromJsonAsync<List<Commande>>('Commande');
        }

        public async Task AddCommandeAsync(Commande commande)
        {
            var response = await _httpClient.PostAsJsonAsync('Commande', commande);
            response.EnsureSuccessStatusCode();
        }
    }
}
```

Copier

> **Avantage de la centralisation** : En utilisant `ApiClient.HttpClient`, tous les services partagent la même instance de `HttpClient` avec la même `BaseAddress`. Cela évite la duplication de code et facilite la maintenance.

Étape 5 : Création de l'interface utilisateur XAML

Modification de MainWindow.xaml

Ouvrez `MainWindow.xaml` et remplacez le contenu par :

```
<Window x:Class='RestaurantWPFClient.MainWindow'
        xmlns='http://schemas.microsoft.com/winfx/2006/xaml/presentation'
        xmlns:x='http://schemas.microsoft.com/winfx/2006/xaml'
        xmlns:d='http://schemas.microsoft.com/expression/blend/2008'
        xmlns:mc='http://schemas.openxmlformats.org/markup-compatibility/2006'
        xmlns:local='clr-namespace:RestaurantWPFClient'
        mc:Ignorable='d'
        Title='Gestion Restaurant' Height='450' Width='800'>
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width='0*'/>
            <ColumnDefinition Width='393*'/>
            <ColumnDefinition Width='407*'/>
        </Grid.ColumnDefinitions>
```

```
        <!-- ComboBox pour choisir le type d'entité -->
        <ComboBox x:Name='Cbx_Choix'
                  Grid.ColumnSpan='2'
                  HorizontalAlignment='Left'
                  Margin='76,123,0,0'
                  VerticalAlignment='Top'
                  Width='120'
                  SelectionChanged='Cbx_Choix_SelectionChanged'>
            <ComboBoxItem Content='Article'/>
            <ComboBoxItem Content='Menu'/>
            <ComboBoxItem Content='Commande'/>
            <ComboBoxItem Content='Table'/>
            <ComboBoxItem Content='Utilisateur'/>
        </ComboBox>
        <!-- Champs de saisie -->
        <Label Content='Nom'
               Grid.ColumnSpan='2'
               HorizontalAlignment='Left'
               Margin='10,179,0,0'
               VerticalAlignment='Top'
               Height='29'
               Width='50'/>
        <TextBox x:Name='tbx_nom'
                 Grid.ColumnSpan='2'
                 HorizontalAlignment='Left'
                 Margin='76,185,0,0'
                 TextWrapping='Wrap'
                 Text=''
                 VerticalAlignment='Top'
                 Width='120'/>
        <Label Content='Prix TTC'
               Grid.ColumnSpan='2'
               HorizontalAlignment='Left'
               Margin='10,0,0,0'
               VerticalAlignment='Center'
               Height='29'
               Width='50'/>
        <TextBox x:Name='tbx_prix_ttc'
                 Grid.ColumnSpan='2'
                 HorizontalAlignment='Left'
                 Margin='76,0,0,0'
                 TextWrapping='Wrap'
                 Text=''
                 VerticalAlignment='Center'
                 Width='120'/>
        <Label Content='Taux TVA'
               Grid.ColumnSpan='2'
               HorizontalAlignment='Left'
               Margin='10,225,0,0'
               VerticalAlignment='Top'
               Height='24'
               Width='61'/>
        <TextBox x:Name='tbx_taux_tva'
                 Grid.ColumnSpan='2'
                 HorizontalAlignment='Left'
                 Margin='76,231,0,0'
                 TextWrapping='Wrap'
                 Text=''
                 VerticalAlignment='Top'
                 Width='120'/>
        <!-- Liste des articles (visible uniquement pour les menus) -->
        <ListBox x:Name='Lbx_Articles'
                 Grid.ColumnSpan='2'
                 Margin='205,137,10,137'
                 Visibility='Collapsed'/>
```

```
        <!-- Bouton de validation -->
        <Button Content='Valider'
                Grid.ColumnSpan='2'
                HorizontalAlignment='Left'
                Margin='76,303,0,0'
                VerticalAlignment='Top'
                Height='21'
                Width='120'
                Click='Valider_Click'/>
    </Grid>
</Window>
```

Copier

Étape 6 : Code-behind de MainWindow

Ouvrez `MainWindow.xaml.cs` et remplacez le contenu par :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using RestaurantWPFClient.Models;
using RestaurantWPFClient.Services;

namespace RestaurantWPFClient
{
    public partial class MainWindow : Window
    {
        private List<Article> _allArticles = new List<Article>();
        private List<CheckBox> _checkboxArticles = new List<CheckBox>();
        private readonly ApiArticleService _apiArticle = new ApiArticleService();
        private readonly ApiMenuService _apiMenu = new ApiMenuService();

        public MainWindow()
        {
            InitializeComponent();
        }

        private async void Valider_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                var selectedChoice =
((ComboBoxItem)Cbx_Choix.SelectedItem)?.Content?.ToString();

                if (string.IsNullOrEmpty(selectedChoice))
                {
                    MessageBox.Show('Veuillez sélectionner un type d'entité.', 'Erreur',
MessageBoxButton.OK, MessageBoxImage.Warning);
                    return;
                }

                string nom = '';
                float prixTtc = 0;
                float tauxTva = 0;
                float prixHt = 0;

                if (selectedChoice == 'Menu')
```

```
                {
                    nom = tbx_nom.Text;
                    prixTtc = Convert.ToSingle(tbx_prix_ttc.Text);
                    tauxTva = Convert.ToSingle(tbx_taux_tva.Text);
                    prixHt = (float)Math.Round(prixTtc / (1 + tauxTva / 100), 2);

                    List<Article> selectedArticles = new List<Article>();

                    foreach (var checkbox in _checkboxArticles)
                    {
                        if (checkbox.IsChecked == true)
                        {
                            selectedArticles.Add((Article)checkbox.Tag);
                        }
                    }

                    Menu menu = new Menu
                    {
                        Nom = nom,
                        PrixTTC = prixTtc,
                        PrixHT = prixHt,
                        TauxTva = tauxTva,
                        Articles = selectedArticles
                    };

                    await _apiMenu.AddMenuAsync(menu);

                    // Réinitialiser les champs
                    tbx_nom.Clear();
                    tbx_prix_ttc.Clear();
                    tbx_taux_tva.Clear();

                    foreach (var checkbox in _checkboxArticles)
                    {
                        if (checkbox.IsChecked == true)
                        {
                            checkbox.IsChecked = false;
                        }
                    }

                    MessageBox.Show('Votre menu a bien été ajouté !', 'Succès',
MessageBoxButton.OK, MessageBoxImage.Information);
                }

                if (selectedChoice == 'Article')
                {
                    nom = tbx_nom.Text;
                    prixTtc = Convert.ToSingle(tbx_prix_ttc.Text);
                    tauxTva = Convert.ToSingle(tbx_taux_tva.Text);
                    prixHt = (float)Math.Round(prixTtc / (1 + tauxTva / 100), 2);

                    Article article = new Article
                    {
                        Nom = nom,
                        PrixTTC = prixTtc,
                        PrixHT = prixHt,
                        TauxTva = tauxTva
                    };

                    await _apiArticle.AddArticleAsync(article);

                    // Réinitialiser les champs
                    tbx_nom.Text = '';
                    tbx_prix_ttc.Text = '';
                    tbx_taux_tva.Text = '';
```

```
                    MessageBox.Show('Votre article a bien été ajouté !', 'Succès',
MessageBoxButton.OK, MessageBoxImage.Information);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($'Une erreur est survenue : {ex.Message}', 'Erreur',
MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }

        private async void Cbx_Choix_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            try
            {
                var selectedChoice =
((ComboBoxItem)Cbx_Choix.SelectedItem)?.Content?.ToString();

                Lbx_Articles.Items.Clear();
                _checkboxArticles.Clear();

                if (selectedChoice == 'Menu')
                {
                    _allArticles = await _apiArticle.GetArticlesAsync();

                    foreach (var article in _allArticles)
                    {
                        var checkbox = new CheckBox
                        {
                            Content = article.Nom,
                            Tag = article,
                        };

                        _checkboxArticles.Add(checkbox);
                        Lbx_Articles.Items.Add(checkbox);
                    }

                    Lbx_Articles.Visibility = Visibility.Visible;
                }
                else
                {
                    Lbx_Articles.Visibility = Visibility.Collapsed;
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($'Erreur lors du chargement des articles : {ex.Message}',
'Erreur', MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
    }
}
```

Copier

Points importants

Gestion des erreurs

Toujours encapsuler les appels API dans des blocs `try-catch` pour gérer les erreurs réseau ou de sérialisation :

```
try
{
    await _apiArticle.AddArticleAsync(article);
    MessageBox.Show('Article ajouté avec succès !');
}
catch (HttpRequestException ex)
{
    MessageBox.Show($'Erreur de connexion à l'API : {ex.Message}');
}
catch (Exception ex)
{
    MessageBox.Show($'Une erreur est survenue : {ex.Message}');
}
```

Copier

Calcul du prix HT

Le prix HT est calculé à partir du prix TTC et du taux de TVA :

```
prixHt = (float)Math.Round(prixTtc / (1 + tauxTva / 100), 2);
```

Copier

Sérialisation JSON

Les objets C# sont automatiquement sérialisés en JSON lors de l'envoi à l'API grâce à `PostAsJsonAsync` et `PutAsJsonAsync`. La désérialisation se fait automatiquement avec `GetFromJsonAsync`.

Prochaines étapes

Maintenant que votre application WPF de base est fonctionnelle, vous pouvez :

1. **Améliorer l'interface utilisateur** : Utiliser des styles, des templates et une meilleure mise en page
2. **Ajouter la validation** : Valider les données avant l'envoi à l'API
3. **Implémenter la mise à jour et la suppression** : Ajouter des boutons pour modifier et supprimer les entités
4. **Ajouter la gestion d'erreurs** : Implémenter une gestion d'erreurs plus robuste
5. **Utiliser l'injection de dépendances** : Utiliser un conteneur IoC pour gérer les services
6. **Ajouter la configuration** : Utiliser `appsettings.json` pour stocker l'URL de l'API
7. **Implémenter MVVM** : Utiliser le pattern MVVM pour séparer la logique de la vue
8. **Ajouter des tests unitaires** : Tester les services API

⚠ Points d'attention

- **URL de l'API** : Assurez-vous que l'URL de l'API est correcte et que l'API est démarrée
- **Certificat SSL** : Si vous utilisez HTTPS avec un certificat auto-signé, vous devrez peut-être désactiver la validation SSL (uniquement en développement)
- **Thread UI** : Les opérations asynchrones doivent être gérées correctement pour éviter de bloquer l'interface utilisateur
- **Gestion des erreurs** : Toujours gérer les erreurs réseau et de sérialisation

---

□ Conclusion

Vous avez maintenant créé une application WPF qui se connecte à une API REST ! Votre application peut créer des articles et des menus, et les envoyer à l'API pour être stockés en base de données.

Pour toute question ou problème, consultez la documentation officielle :

- WPF Documentation
- HttpClient Documentation
- ASP.NET Core Web API