

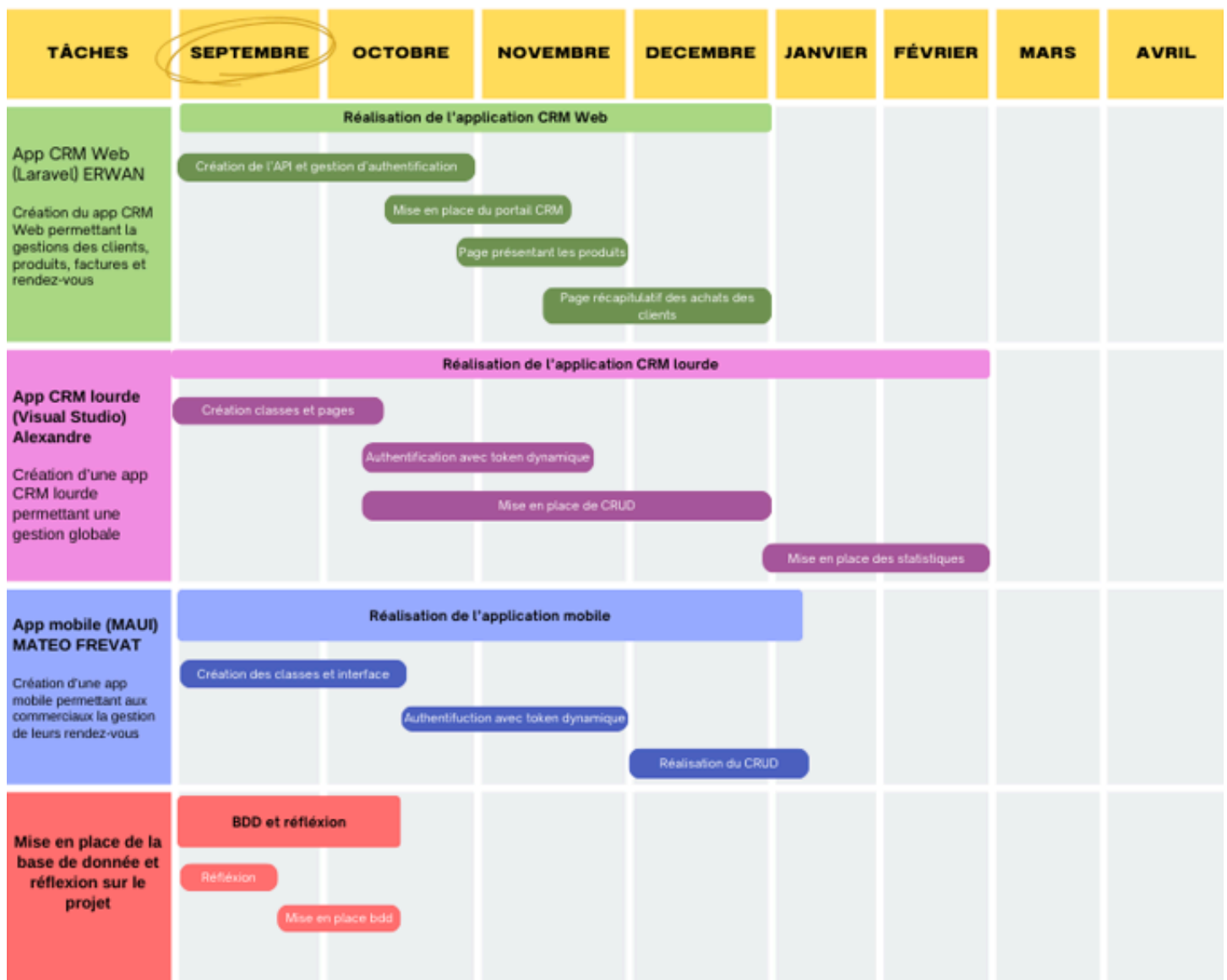
Infotools

Objectif Principal

Réalisation d'une application multiplateforme avec une version Web, mobile et logiciel avec l'utilisation d'une API.

Les diagrammes de GANTT

Prévisionnel



Réel

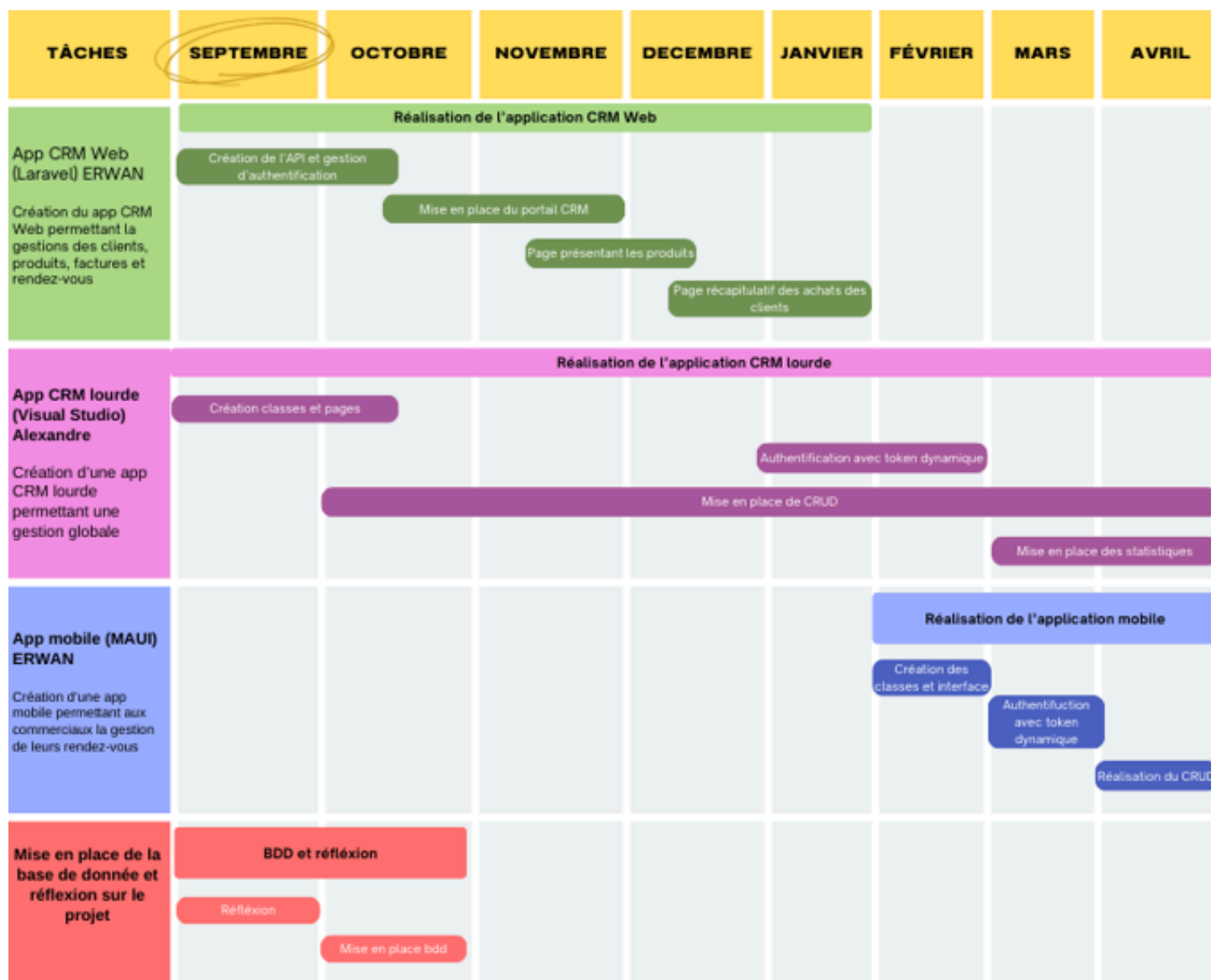
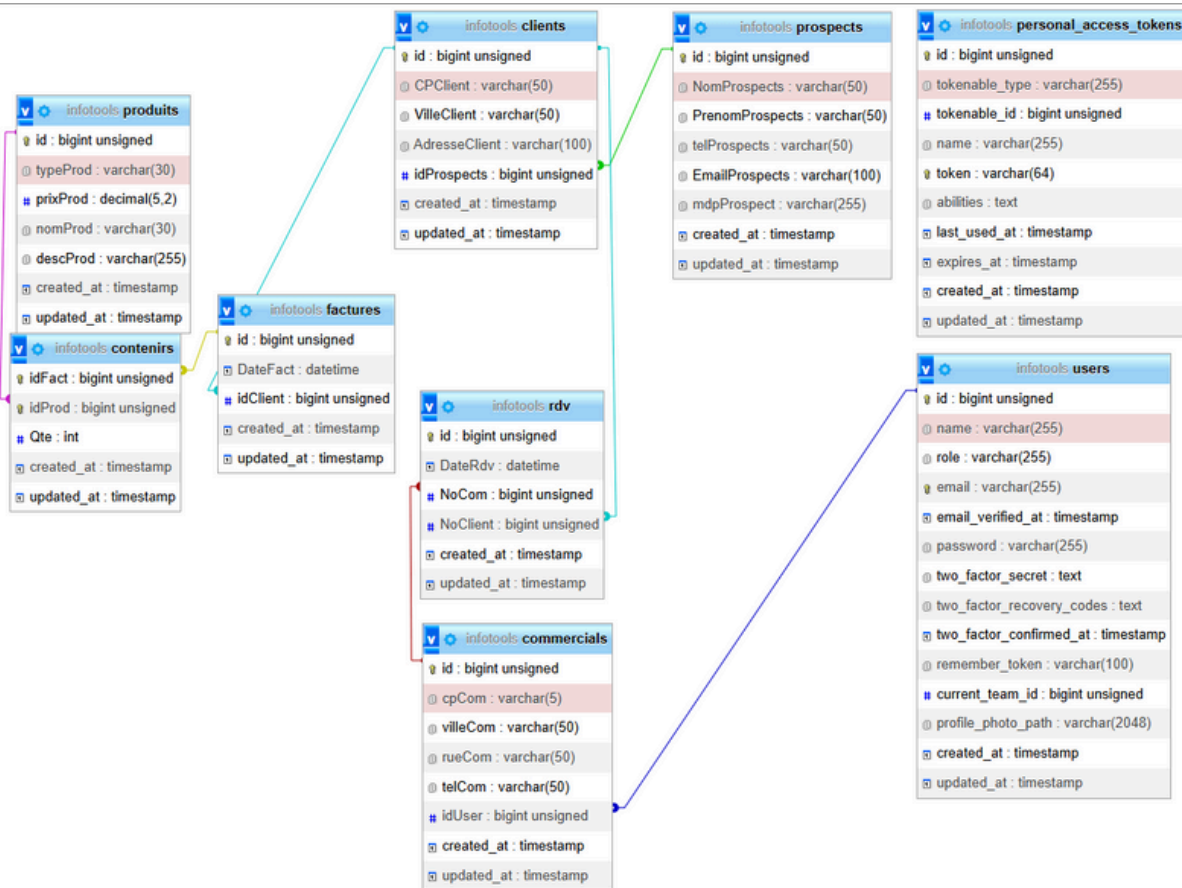


Schéma relationnel de la base de donnée



L'application mobile (MAUI)

1. Objectif

Réaliser une application mobile permettant aux commerciaux d'avoir accès à leurs rendez-vous, de pouvoir les consulter, les modifier, supprimer ou encore en ajouter.

Ici on a donc utilisé l'API Laravel créé en amont et MAUI sur Visual Studio 2022 pour faire un CRUD (create, read, update, delete) sur les rendez-vous.

De plus, un système d'authentification a été fait avec une gestion de token dynamique permettant aux commerciaux de se connecter en toute sécurité.

2. Login

Interface

Login

Email

Mot de passe

Se connecter

Code

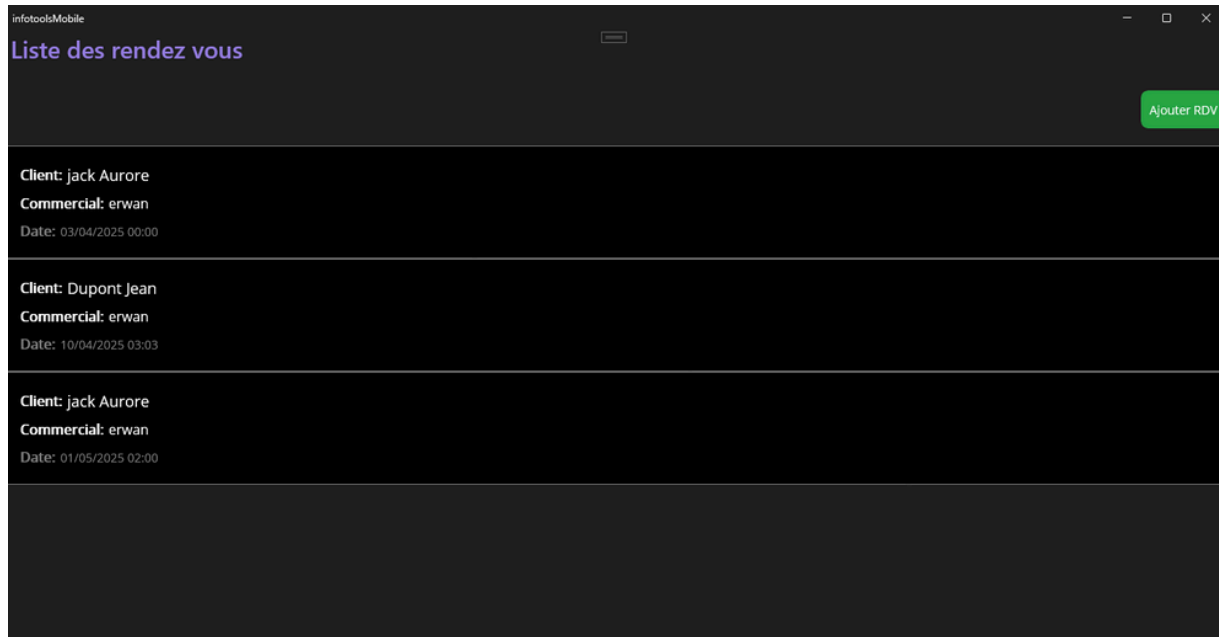
```
var jsonContent = new StringContent(JsonConvert.SerializeObject(LoginData), Encoding.UTF8, "application/json");
try
{
    HttpResponseMessage response = await client.PostAsync(apiUrl, jsonContent);
    if (response.IsSuccessStatusCode)
    {
        string jsonResponse = await response.Content.ReadAsStringAsync();
        var apiResponse = JsonConvert.DeserializeObject<dynamic>(jsonResponse);

        if (apiResponse.success == true)
        {
            return new user
            {
                Id = apiResponse.data.idUser,
                name = apiResponse.data.name.ToString(),
                token = apiResponse.data.token.ToString()
            };
        }
    }
    else
    {
        Console.WriteLine("Échec de la connexion, statut: " + response.StatusCode);
    }
}
```

Reçoit deux chaînes de caractères qui l'envoie à l'API pour recevoir un token.

3. Page d'accueil

Interface



Code

```
try
{
    HttpResponseMessage response = await client.GetAsync(apiUrl);
    if (response.IsSuccessStatusCode)
    {
        string jsonResponse = await response.Content.ReadAsStringAsync();
        var apiResponse = JsonConvert.DeserializeObject<dynamic>(jsonResponse);

        // Vérification du succès de la réponse
        if (apiResponse.succes == true)
        {
            List<Rdv> rdvList = new List<Rdv>();

            // Boucle sur chaque rendez-vous retourné dans "data"
            foreach (var rdvData in apiResponse.data)
            {
                string dateFormat = "yyyy-MM-dd HH:mm:ss";
                DateTime dateRdv;

                if (DateTime.TryParseExact(rdvData.date.ToString(), dateFormat, CultureInfo.InvariantCulture, DateTimeStyles.None, out dateRdv))
                {
                    rdvList.Add(new Rdv
                    {
                        Id = rdvData.id,
                        Client = rdvData.client.nom + " " + rdvData.client.prenom,
                        NameCom = rdvData.commercial.name,
                        DateRdv = dateRdv
                    });
                }
                else
                {
                    Console.WriteLine($"Erreur de format pour la date du rendez-vous ID: {rdvData.id}");
                }
            }

            return rdvList;
        }
    }
}
```

Permet de récupérer la liste des rendez-vous du commercial connecté.

4. Page d'ajout

Interface

The screenshot shows a mobile application interface titled 'Ajouter un rendez-vous' (Add Appointment). The interface is dark-themed. At the top, there's a header with the title. Below it, a section labeled 'Sélectionnez un client :'. Under this label is a search bar with the placeholder text 'Rechercher un client...'. Below the search bar is a dropdown menu labeled 'Choisissez un client'. Further down, there's a section labeled 'Date et heure du rendez-vous :'. This section contains a date picker showing '4/10/2025' and a time picker showing '12:00 AM'. At the bottom of the form is a large purple button labeled 'Ajouter le rendez-vous'.

Code

```
public static async Task<bool> AddRdv(Client unClient , string uneDate)
{
    HttpClient client = new HttpClient();
    string apiUrl = "http://infotools.test/api/rdvs";
    string token = await SecureStorage.GetAsync("UserToken");
    client.DefaultRequestHeaders.Add("Authorization", "Bearer " + token);

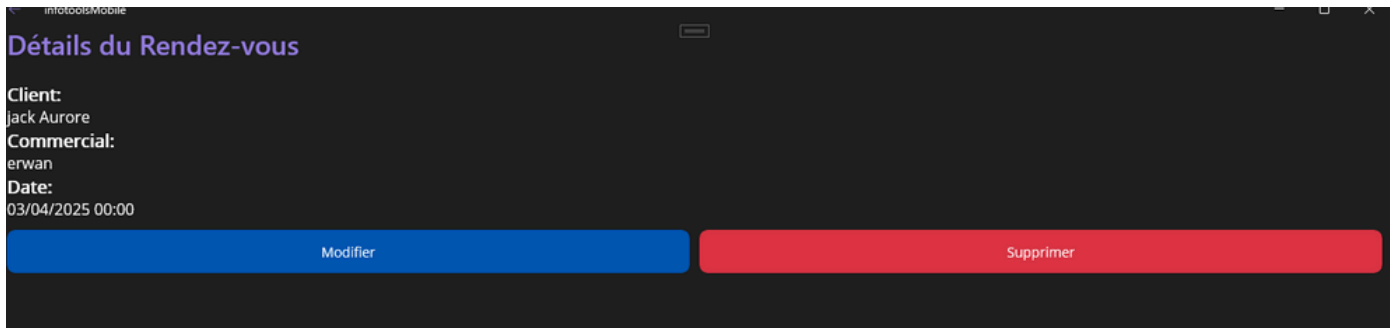
    var value = new
    {
        DateRdv = uneDate,
        NoClient = Convert.ToString(unClient.id)
    };
    var jsonContent = new StringContent(JsonConvert.SerializeObject(value), Encoding.UTF8, "application/json");
    try
    {
        HttpResponseMessage response = await client.PostAsync(apiUrl, jsonContent);

        if (response.IsSuccessStatusCode)
        {
            string jsonResponse = await response.Content.ReadAsStringAsync();
            var apiResponse = JsonConvert.DeserializeObject<dynamic>(jsonResponse);
            return apiResponse.success;
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Erreur de connexion: {ex.Message}");
    }
    return false;
}
```

Permet l'ajout d'un rendez-vous en contactant l'API.

5. Page de détail

Interface



Code suppression

```
public static async Task<bool> DeleteRdv(int id)
{
    string apiUrl = "http://infotools.test/api/rdvs";
    string token = await SecureStorage.GetAsync("UserToken");

    try
    {
        using (HttpClient client = new HttpClient())
        {
            // Ajouter le token d'authentification
            client.DefaultRequestHeaders.Add("Authorization", "Bearer " + token);

            // Effectuer la requête DELETE pour supprimer le rendez-vous
            HttpResponseMessage response = await client.DeleteAsync($"{apiUrl}/{id}");

            // Vérifier si la réponse est réussie
            return response.IsSuccessStatusCode;
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Erreur lors de la suppression du rendez-vous : " + ex.Message);
        return false;
    }
}
```

Permettant la suppression du rendez-vous en passant l'id du rdv à l'API.

6. Page de modification

Interface

The screenshot shows a mobile application window titled 'infotoolsMobile'. The main heading is 'Mettre à jour le rendez-vous' in purple. Below it, the text 'Sélectionnez un client :' is followed by a search bar with the placeholder 'Rechercher un client...'. Underneath is a dropdown menu labeled 'Choisissez un client' with 'Jack Aurore' selected. The next section is 'Date du rendez-vous :' with a date picker showing '4/3/2025'. Below that is 'Heure du rendez-vous :' with a time picker showing '12:00 AM'. At the bottom is a large blue button labeled 'Mettre à jour le rendez-vous'.

Code

```
public static async Task<bool> UpdateRdv(Client unClient, string uneDate, int rdvId)
{
    HttpClient client = new HttpClient();
    string apiUrl = $"http://infotools.test/api/rdvs/{rdvId}"; // Utiliser l'ID du rendez-vous pour l'URL
    string token = await SecureStorage.GetAsync("UserToken");
    client.DefaultRequestHeaders.Add("Authorization", "Bearer " + token);

    var value = new
    {
        DateRdv = uneDate, // La nouvelle date du rendez-vous
        NoClient = Convert.ToString(unClient.id) // ID du client associé au rendez-vous
    };

    var jsonContent = new StringContent(JsonConvert.SerializeObject(value), Encoding.UTF8, "application/json");

    try
    {
        // Utilisation de la méthode PUT pour mettre à jour le rendez-vous existant
        HttpResponseMessage response = await client.PutAsync(apiUrl, jsonContent);

        if (response.IsSuccessStatusCode)
        {
            string jsonResponse = await response.Content.ReadAsStringAsync();
            var apiResponse = JsonConvert.DeserializeObject<dynamic>(jsonResponse);

            return apiResponse.success; // Retourner true si la mise à jour a réussi
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Erreur de connexion: {ex.Message}");
    }

    return false; // Retourner false en cas d'échec
}
```

Permet la modification en passant comme paramètre l'id dans l'url et les valeurs à modifier dans la requête.

