Documentation Technique – Info-Tools (Mobile)

Introduction

L'application mobile Info-Tools a été développée avec **.NET MAUI**, une technologie multiplateforme de Microsoft permettant de créer des applications natives Android, iOS, Windows et MacOS avec un seul code C#. Cette version mobile est connectée à l'API Laravel déjà existante, permettant une synchronisation fluide des données.

Outils utilisées :

Stack Technique Complète

Frontend Mobile:

- Framework: .NET MAUI 8.0 (Multiplateforme)
- Langage: C# 11.0
- IDE Principal: Visual Studio 2022 17.8+
- Gestion de dépendances: NuGet

Backend:

- API REST: Laravel 10.x
- Authentification: Sanctum (JetStream)
- Base de données: MySQL 8.0

Outils Complémentaires:

- Postman: Test des endpoints API
- Swagger: Documentation interactive de l'API
- Git: Gestion de version (GitHub/GitLab)

Architecture Technique

Schéma d'Architecture Modulaire

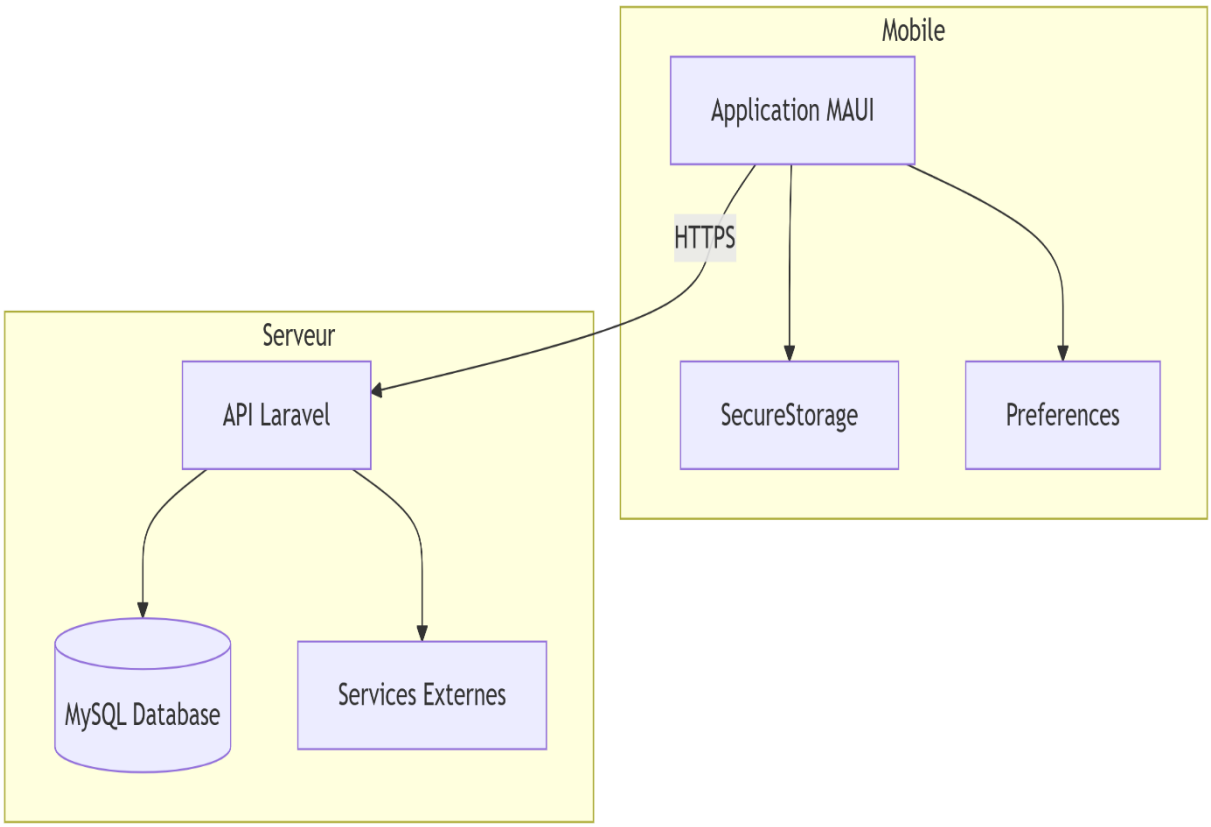
🔐 **Authentification**

Le système d'authentification repose sur **Laravel Sanctum** :

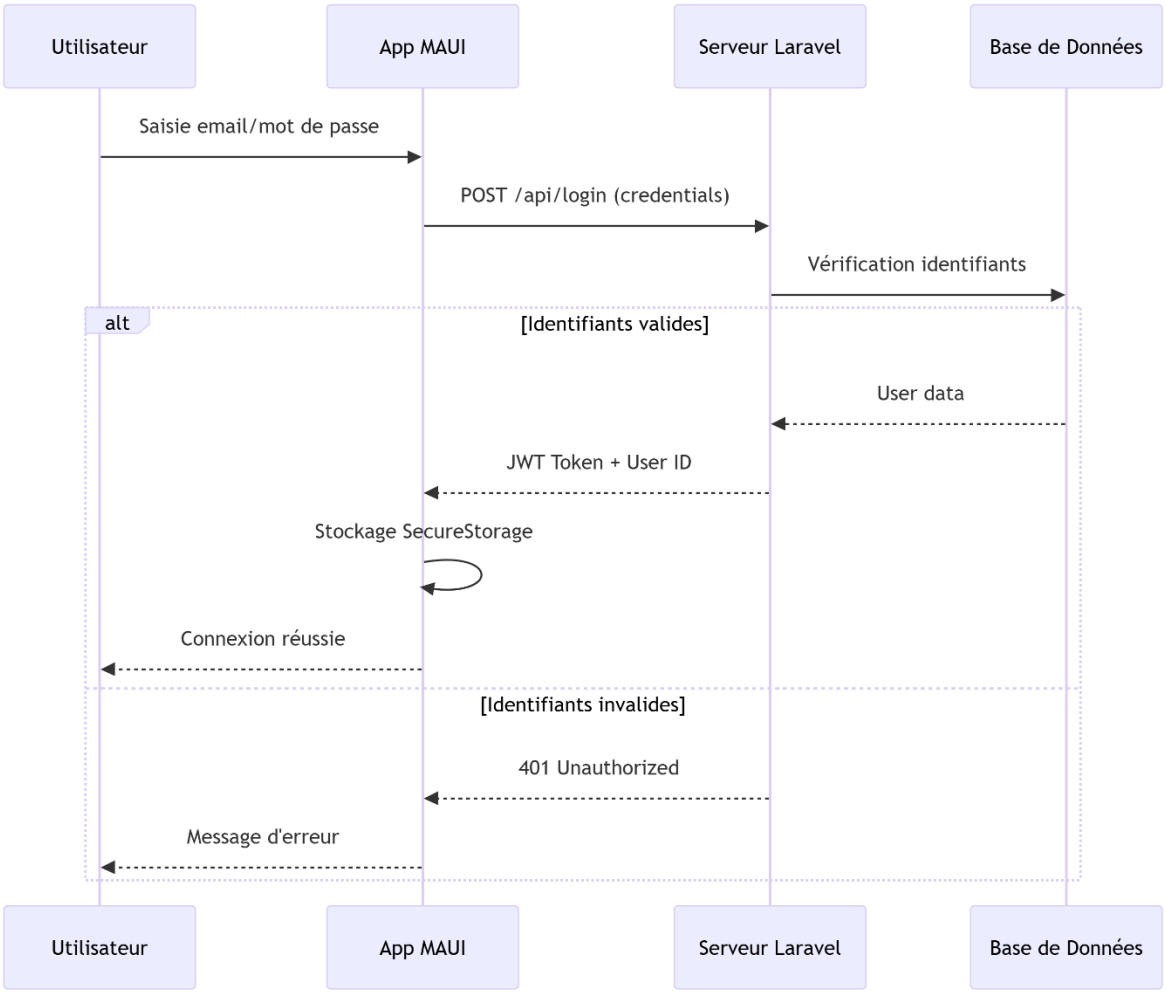
- Lors de la connexion, les identifiants sont transmis de manière sécurisée à l'API Laravel via `HttpClient`.
- En cas de succès, un **jeton d'accès (token)** est généré par le serveur.
- Ce token est ensuite stocké localement de manière sécurisée à l'aide de la classe `SecureStorage` fournie par MAUI, permettant une authentification persistante et fiable.

Architecture Technique

Schéma d'Architecture Modulaire :



Fonctionnement de l'Authentification :



Implémentation Technique :

Couche Réseau (API Communication) :

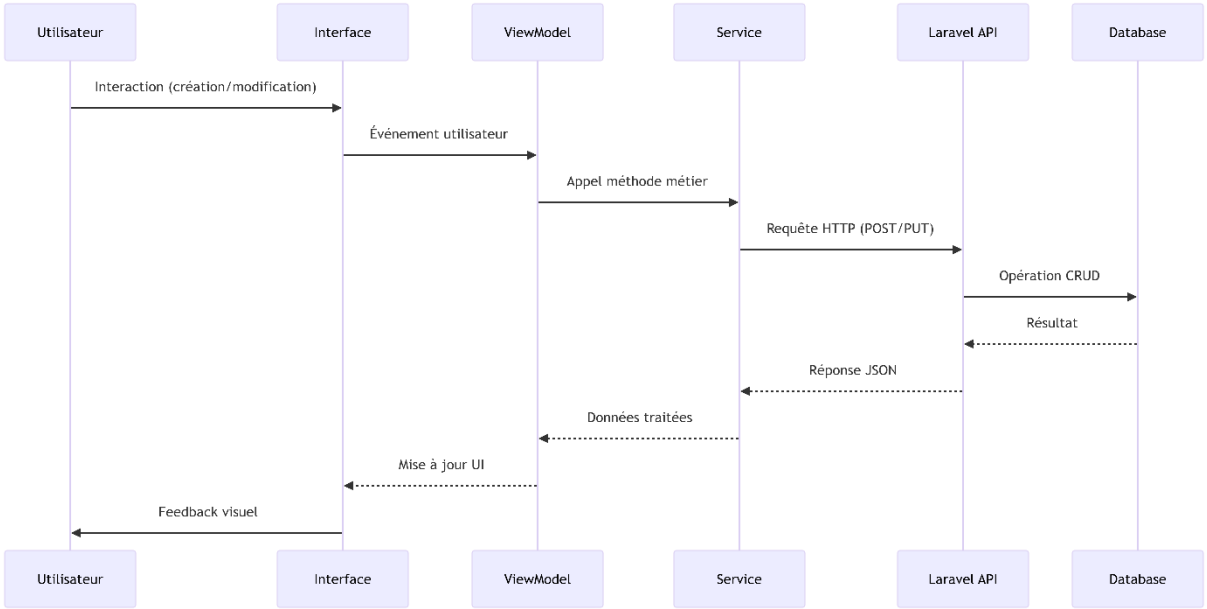
```
using System.Text;
using System.Threading.Tasks;
using Newtonsoft.Json;

namespace Inftools.App.Services
{
    4 références
    public class AuthService
    {
        private readonly HttpClient _httpClient;

        1 référence
        public AuthService()
        {
            _httpClient = new HttpClient();
        }

        // Méthode pour se connecter et obtenir le token et commercial_id
        1 référence
        public async Task<LoginResponse> LoginAsync(string email, string password)
        {
            var url = "http://127.0.0.1:8001/api/login";
            var data = new StringContent($"{{\"email\": \"{email}\", \"password\": \"{password}\"}}", Encoding.UTF8, "application/json");

            var response = await _httpClient.PostAsync(url, data);

            if (response.IsSuccessStatusCode)
            {
                var responseData = await response.Content.ReadAsStringAsync();
                // Affichage de la réponse brute pour voir son contenu
                Console.WriteLine($"Réponse brute de l'API : {responseData}");

                // Désérialiser la réponse en objet LoginResponse
                var loginResponse = JsonConvert.DeserializeObject<LoginResponse>(responseData);

                // Affichage du contenu de loginResponse pour débogage
                Console.WriteLine($"Réponse de connexion : Token = {loginResponse.Token}, ID = {loginResponse.id}");

                if (loginResponse != null && !string.IsNullOrEmpty(loginResponse.Token))
                {
                    // Sauvegarder le token et l'ID de l'utilisateur dans les préférences
                    Preferences.Set("AuthToken", loginResponse.Token);
                    Preferences.Set("UserId", loginResponse.id);  // Sauvegarder l'ID sous "UserId"
                    //teste  verification de l id et le token
                    //await Application.Current.MainPage.DisplayAlert(
                    //        "Bienvenue",
                    //        $"Connexion réussie !\n\nVotre token : {loginResponse.Token}\nVotre ID utilisateur : {loginResponse.id}",
                    //        "OK"
                    //    );
                    return loginResponse; // Retourner l'objet avec le token et l'ID
                }
            }

            return null; // Si la réponse ne contient pas de token, retourner null
        }


        // Méthode pour vérifier si le token est valide
        0 références
        public async Task<bool> IsTokenValidAsync(string token)
        {
            var url = "http://127.0.0.1:8001/api/user"; // Remplacez par l'URL de votre endpoint
            _httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);

            var response = await _httpClient.GetAsync(url);
            return response.IsSuccessStatusCode;
```

```
        return null; // Si la réponse ne contient pas de token, retourner null
    }


    // Méthode pour vérifier si le token est valide
    0 références
    public async Task<bool> IsTokenValidAsync(string token)
    {
        var url = "http://127.0.0.1:8001/api/user"; // Remplacez par l'URL de votre endpoint
        _httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);

        var response = await _httpClient.GetAsync(url);
        return response.IsSuccessStatusCode;

    }
    0 références
    public async Task<UserDetails> GetUserDetailsAsync(string token)
    {
        var url = "http://127.0.0.1:8001/api/user"; // Remplacez par l'URL de votre API
        _httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);

        var response = await _httpClient.GetAsync(url);

        if (response.IsSuccessStatusCode)
        {
            var responseData = await response.Content.ReadAsStringAsync();
            return JsonConvert.DeserializeObject<UserDetails>(responseData);
        }

        return null;
    }
    2 références
```

Architecture des Rendez-vous

Gestion des Rendez-vous dans Info-Tools Mobile :

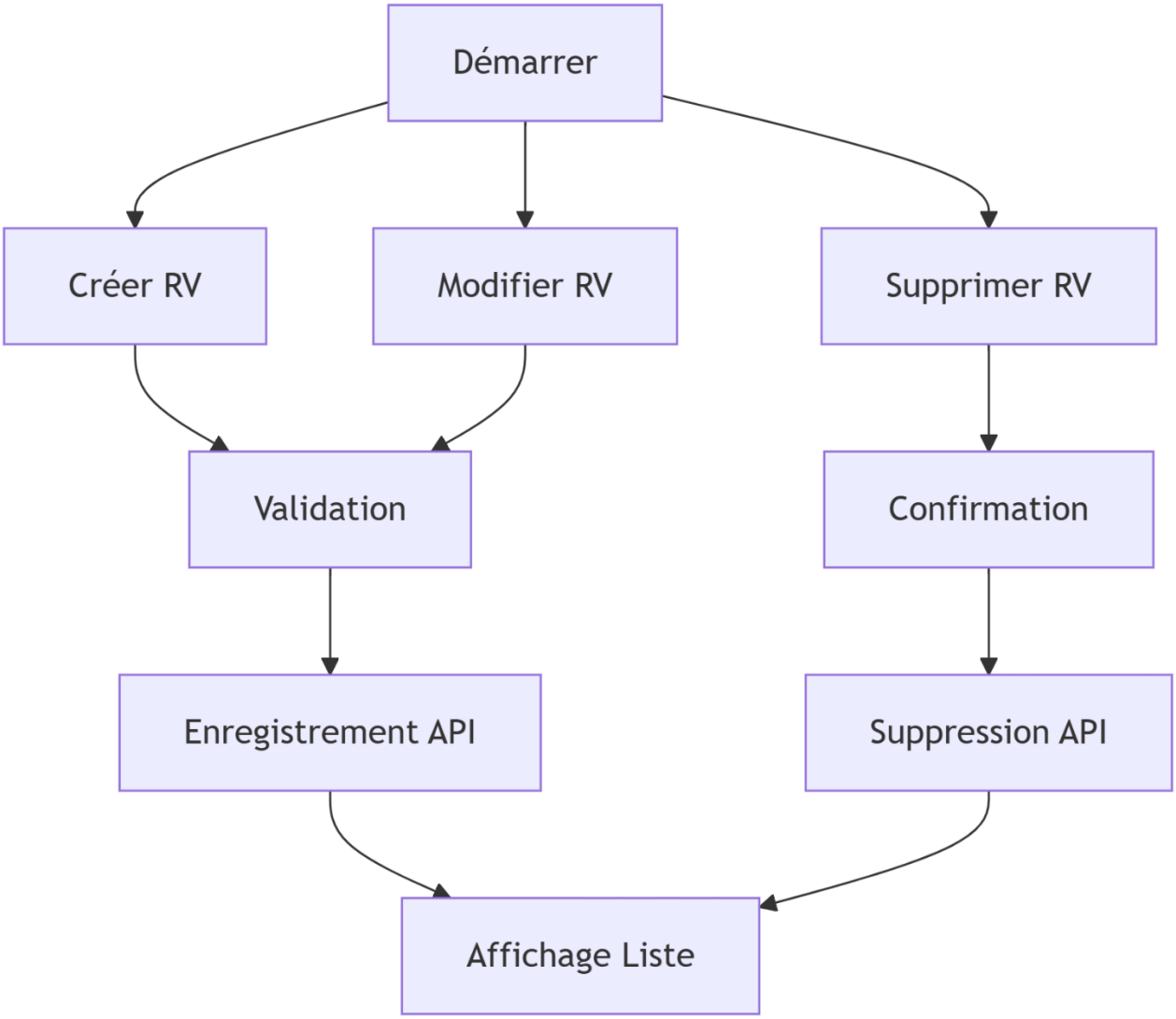
Workflow Complet :



Implémentation du CRUD

1. Modèle de Données :

```
public class Appointment
{
    [JsonProperty("id")]
    public int Id { get; set; }

    [JsonProperty("date_time")]
    public DateTime DateTime { get; set; }

    [JsonProperty("location")]
    public string Location { get; set; }

    [JsonProperty("status")]
    public string Status { get; set; } // "planned", "completed", "canceled"

    [JsonProperty("client")]
    public Client Client { get; set; }

    [JsonProperty("notes")]
    public string Notes { get; set; }

    [JsonIgnore]
    public string FormattedDate => DateTime.ToString("dd/MM/yyyy
HH:mm");
}
```



2. Ajout d'un Rendez-vous

```
private async Task CreateAppointment()
{
    if (ClientPicker.SelectedItem is Client selectedClient)
    {
        var dateTime = DatePicker.Date.Add(TimePicker.Time);
        string formattedDateTime = dateTime.ToString("yyyy-MM-dd'T'HH:mm:ss");

        var location = LocationEntry.Text;
        var status = StatusPicker.SelectedItem?.ToString();

        var appointment = new
        {
            user_id = Preferences.Get("UserId", string.Empty),
            id = selectedClient.id,
            date_time = formattedDateTime,
            location = location,
            status = status
        };

        var url = "http://127.0.0.1:8001/api/appointments";
        bool success = await SendAppointmentRequest(url, HttpMethod.Post, appointment);

        if (success)
        {
            await DisplayAlert("Succès", "Le rendez-vous a été créé avec succès.", "OK");
            await Navigation.PopAsync();
        }
    }
    else
    {
        await DisplayAlert("Erreur", "Veuillez sélectionner un client.", "OK");
    }
}
```

3. Modification d'un Rendez-vous

```
private async Task UpdateAppointment()
{
    if (AppointmentToEdit != null && ClientPicker.SelectedItem is Client selectedClient)
    {
        var dateTime = DatePicker.Date.Add(TimePicker.Time);
        string formattedDateTime = dateTime.ToString("yyyy-MM-dd'T'HH:mm:ss");

        var appointmentData = new
        {
            user_id = Preferences.Get("UserId", string.Empty),
            id = selectedClient.id,
            date_time = formattedDateTime,
            location = LocationEntry.Text,
            status = StatusPicker.SelectedItem?.ToString()
        };

        var url = $"http://127.0.0.1:8001/api/appointments/{AppointmentToEdit.appointment_id}";
        bool success = await SendAppointmentRequest(url, HttpMethod.Put, appointmentData);

        if (success)
        {
            await DisplayAlert("Succès", "Le rendez-vous a été modifié avec succès.", "OK");
            await Navigation.PopAsync();
        }
        else
        {
            await DisplayAlert("Erreur", "Impossible de modifier le rendez-vous.", "OK");
        }
    }
    else
    {
        await DisplayAlert("Erreur", "Veuillez sélectionner un client valide.", "OK");
    }
}
```

4. Suppression d'un Rendez-vous
Code d'Implémentation :

```
0 references
private async void OnDeleteButtonClicked(object sender, EventArgs e)
{
    Button button = sender as Button;
    Appointment appointmentToDelete = button?.CommandParameter as Appointment;

    if (appointmentToDelete != null)
    {
        bool confirm = await DisplayAlert("Supprimer", "Voulez-vous vraiment supprimer ce rendez-vous ?", "Oui", "Non");

        if (confirm)
        {
            string token = Preferences.Get("AuthToken", string.Empty);
            if (string.IsNullOrEmpty(token))
            {
                await DisplayAlert("Erreur", "Token manquant. Veuillez vous reconnecter.", "OK");
                return;
            }

            bool success = await _appointmentService.DeleteAppointmentAsync(appointmentToDelete.appointment_id, token);

            if (success)
            {
                await LoadAppointmentsAsync(token);
                await DisplayAlert("Succès", "Le rendez-vous a été supprimé avec succès.", "OK");
            }
            else
            {
                await DisplayAlert("Erreur", "Échec de la suppression du rendez-vous.", "OK");
            }
        }
    }
}
```

Conclusion

Le projet **Info-Tools** est une solution fonctionnelle et complète pour la gestion des rendez-vous, intégrant une authentification sécurisée avec **JetStream**, une API REST efficace et un contrôle des accès via **Policies Laravel**. Il répond aux besoins essentiels, mais des améliorations sont possibles, notamment sur la sécurité avancée et la récupération des comptes. Globalement, l'application est stable, évolutive et prête à être optimisée selon les futurs besoins.