Rapport - Application météo iOS avec SwiftUI

Aya EL HADDAJ

Introduction

Ce rapport présente une analyse complète de l'application iOS WeatherApp. Cette application est une reproduction du design de l'application Météo d'Apple, construite avec **SwiftUI** et utilisant l'API **Openweathermap**. Le projet démontre l'utilisation des technologies modernes de développement iOS pour créer une interface utilisateur fluide et responsive pour afficher les données météorologiques.

Structure et architecture

L'application est conçue selon le modèle d'architecture **MVVM** (Model-View-ViewModel), qui offre une séparation claire entre les données, la logique métier et l'interface utilisateur.

Modèles de données

L'application utilise plusieurs structures de données pour représenter différents aspects des informations météorologiques :

- **CurrentWeather** : Contient les données météorologiques actuelles d'une localité
- **ForecastWeather** : Représente les prévisions météorologiques pour une période spécifique
- **ForecastWeatherResponse** : Contient la réponse complète de l'API pour les prévisions
- **WeatherElement** : Décrit un élément météorologique spécifique (comme la pluie, les nuages, etc.)
- **Coordinate** : Représente les coordonnées géographiques d'une localité
- **WeatherWind** : Contient les informations sur le vent
- **WeatherClouds** : Contient les informations sur la couverture nuageuse

Toutes ces structures implémentent le protocole **Codable**, ce qui facilite leur décodage à partir des réponses JSON de l'API.

Réseau et API

La classe OpenweatherAPIClient est responsable de toutes les communications avec l'API Openweathermap. Elle gère :

- La construction des URLs d'API
- L'envoi des requêtes réseau

- Le décodage des réponses JSON en modèles de données

L'application utilise deux endpoints principaux :

1. **/weather** pour obtenir les conditions météorologiques actuelles
2. **/forecast** pour obtenir les prévisions météorologiques

ViewModel

La classe **WeatherViewModel** sert d'intermédiaire entre les modèles de données et les vues. Elle :

- Gère l'état de l'application (chargement, succès, échec)
- Maintient les données météorologiques actuelles et les prévisions
- Transforme les données brutes en formats plus facilement utilisables par les vues
- Initialise et coordonne les appels réseau

Ce ViewModel suit le pattern Observable, permettant aux vues de réagir automatiquement aux changements de données.

Vues

L'application comprend plusieurs composants de vue réutilisables :

- **ContentView** : Vue principale qui organise l'ensemble de l'interface
- **BackgroundView** : Fournit un dégradé de couleur bleu pour l'arrière-plan
- **LocationAndTemperatureHeaderView** : Affiche le nom de la localité et la température actuelle
- **HourlyWeatherView** : Affiche les prévisions météorologiques horaires
- **DailyWeatherView** : Affiche les prévisions météorologiques quotidiennes

- **DetailsCurrentWeatherView** : Affiche des détails supplémentaires sur les conditions actuelles

Les vues utilisent des composants **SwiftUI** comme **VStack**, **HStack**, **ScrollView** et **Image** pour créer une interface utilisateur responsive et visuellement attrayante.

Fonctionnalités

Météo actuelle

L'application affiche les informations météorologiques actuelles, notamment :

- Température actuelle
- Conditions météorologiques (ensoleillé, nuageux, etc.)
- Température ressentie
- Pression atmosphérique
- Humidité
- Visibilité
- Lever et coucher du soleil

Prévisions horaires

Une vue horizontale scrollable affiche les prévisions météorologiques par heure, avec :

- Heure
- Icône représentant les conditions météorologiques
- Température
- Humidité

Prévisions quotidiennes

Une liste verticale montre les prévisions météorologiques pour les prochains jours, avec :

- Jour de la semaine
- Icône représentant les conditions météorologiques
- Températures maximales et minimales

Gestion des erreurs

L'application gère les erreurs réseau et fournit un bouton de réessai en cas d'échec lors de la récupération des données.

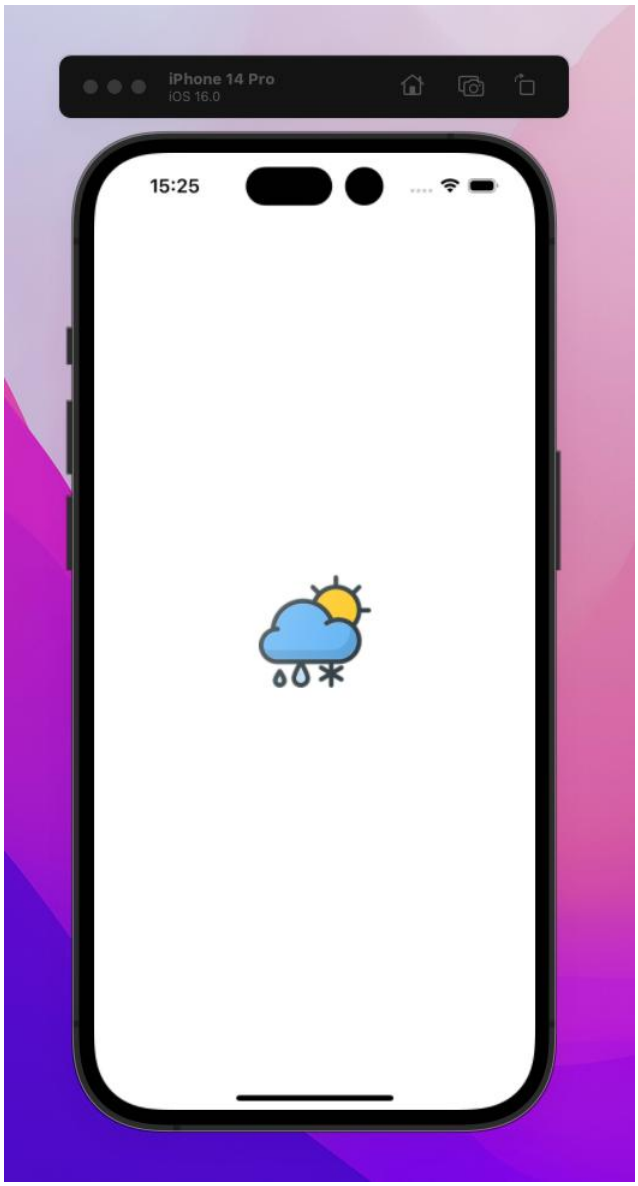
Interface utilisateur

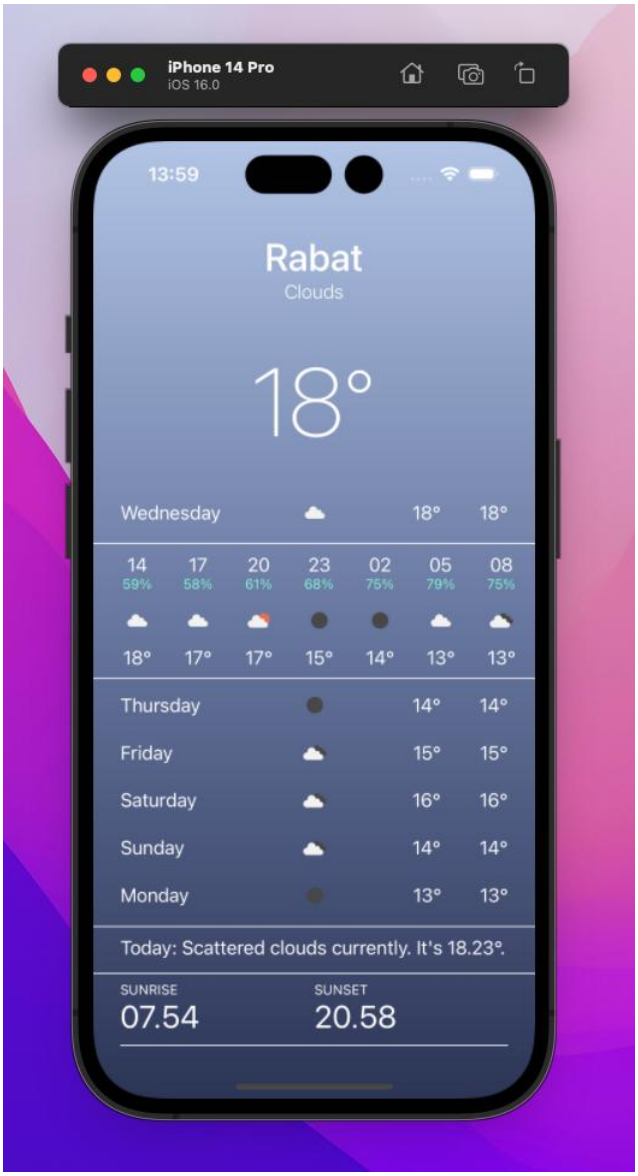
L'interface utilisateur est inspirée de l'application Météo d'Apple, avec un design minimaliste et élégant. La palette de couleurs utilise différentes nuances de bleu pour l'arrière-plan, créant une ambiance calme et intuitive pour une application météo.

Les composants UI sont bien organisés avec :

- Une en-tête affichant le nom de la ville et la température actuelle
- Une vue défilante verticale pour les prévisions et les détails
- Des séparateurs visuels entre les différentes sections d'information

L'application utilise des effets de défilement à la fois horizontaux et verticaux pour organiser les différentes données météorologiques de manière intuitive et accessible.





Aspects techniques

Gestion des dates

L'application utilise des extensions sur Int et Date pour faciliter la conversion et le formatage des timestamps Unix fournis par l'API. Ces extensions permettent d'extraire facilement les jours de la semaine, les heures et d'autres informations temporelles.

Customisation des composants UIKit

La vue **ActivityIndicatorView** est un exemple d'intégration d'un composant **UIKit** dans **SwiftUI**, démontrant comment les deux frameworks peuvent coexister dans une même application.

État de l'application

L'application utilise une énumération **StateView** pour gérer son état interne (chargement, succès, échec), ce qui permet d'afficher différentes interfaces en fonction de l'état actuel.

Configuration et déploiement

Pour utiliser l'application, il faut :

1. Obtenir une clé API d'Openweathermap
2. Configurer l'ID de la ville (dans ce projet, Rabat avec l'ID 2538475)
3. Lancer l'application sur un simulateur ou un appareil iOS

Perspectives d'amélioration

Plusieurs améliorations pourraient être envisagées :

- Ajout de la géolocalisation pour obtenir la météo de l'emplacement de l'utilisateur
- Intégration de notifications pour les alertes météorologiques
- Ajout de la possibilité de rechercher et sauvegarder plusieurs villes
- Implémentation d'un widget pour l'écran d'accueil

Conclusion

L'application iOS WeatherApp est un exemple bien structuré d'une application météo moderne utilisant **SwiftUI** et l'architecture **MVVM**. Elle démontre efficacement l'utilisation des technologies Apple les plus récentes pour créer une expérience utilisateur fluide et intuitive tout en maintenant un code propre et organisé.

La séparation des préoccupations entre les modèles, les vues et les viewmodels permet une maintenance facile et une extensibilité du code, tandis que l'utilisation de SwiftUI permet de créer rapidement une interface utilisateur réactive et visuellement attrayante.