TYPE DE DOCUMENT	FICHE DE PROJETS
CYCLE	LICENCE
FILIERE ET NIVEAU	GL 3, GI 3
MATIÈRE	DÉVELOPPEMENT D'APPLICATIONS MOBILES
ENSEIGNANT	Ing. GHOMSI GHOMSI EMMANUEL

CONTEXTE GÉNÉRAL (PÉDAGOGIQUE)

Dans le cadre de votre formation en développement d'applications mobiles, vous allez réaliser un mini-projet d'application météo avec **React Native**.

Ce projet vise à **vous faire pratiquer les compétences suivantes** :

- Création de **composants réutilisables**
- Utilisation de la **navigation entre écrans (Stack)**
- Mise en œuvre de la **navigation par onglets (Tabs)**
- Intégration d'une **API publique** : OpenWeatherMap
- Lecture et manipulation de **données JSON**
- Conception d'une **interface utilisateur inspirée d'une maquette Figma**

Chaque groupe d'étudiant choisira **un projet parmi les six proposés**, avec un contexte d'utilisation précis (tourisme, agriculture, social, sport, etc.).

Projet 1 — WeatherNow

Contexte du projet

Ce projet simule une application météo urbaine, destinée à informer les utilisateurs sur le climat de différentes villes populaires du Cameroun. L'utilisateur sélectionne une ville dans une liste, puis accède à un écran de détails météo pour cette ville. L'objectif est de manipuler des données simples avec un flux clair (sélection → détail), tout en s'exerçant à structurer une application mobile à onglets.

Objectifs pédagogiques

- Créer une liste dynamique de villes
- Naviguer entre écrans (liste → détail)
- Faire un appel API (`fetch`) en passant la latitude/longitude comme paramètres
- Afficher des informations météo (température, humidité, ciel, etc.)
- Organiser les données dans des composants propres

Fonctionnalités attendues

- Accueil (onglet principal) :
  - Liste de villes avec bouton ou carte cliquable
  - Appui sur une ville → navigation vers l'écran de détails météo
- Écran de détails météo :
  - Nom de la ville
  - Température actuelle (°C)
  - Description (ex: "partiellement nuageux")
  - Taux d'humidité
  - Icône de météo selon l'état du ciel

Liste de villes avec coordonnées GPS

Vous allez inclure cette liste statiquement dans votre code (`cities.ts` par exemple) :

```
export const cities = [
  { name: "Yaoundé", lat: 3.848, lon: 11.5021 },
  { name: "Douala", lat: 4.0511, lon: 9.7679 },
  { name: "Garoua", lat: 9.3014, lon: 13.3956 },
  { name: "Maroua", lat: 10.5956, lon: 14.3247 },
  { name: "Bafoussam", lat: 5.4778, lon: 10.4172 },
  { name: "Bertoua", lat: 4.5773, lon: 13.6846 },
  { name: "Ebolowa", lat: 2.9031, lon: 11.1519 },
];
```

Chaque ville aura une fiche météo récupérée via l'API OpenWeatherMap :

```
https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API_KEY}&units=metric
```

Écrans attendus

1. Liste des villes (FlatList ou ScrollView)
2. Détail météo

Projet 2 — TravelBuddy

Contexte du projet

TravelBuddy est une application météo conçue pour les **voyageurs**. Elle permet de gérer une liste personnalisée de **destinations touristiques**, et de consulter rapidement la météo actuelle de chaque site. L'utilisateur peut ajouter ou retirer des lieux de sa liste de favoris, puis explorer les prévisions météorologiques associées. Ce projet est axé sur la gestion de **favoris**, la navigation par **onglets**, et l'intégration dynamique d'une API.

Objectifs pédagogiques

- Utiliser la navigation **par onglets** (Tabs)
- Gérer une **liste de favoris** en mémoire locale
- Utiliser `fetch` pour obtenir la météo à partir de **coordonnées GPS**
- Organiser plusieurs types d'écrans dans un projet structuré
- Créer des composants personnalisés pour les lieux

Fonctionnalités attendues

- **Onglet Accueil** :
  - Liste complète de **sites touristiques** du Cameroun
  - Chaque carte de lieu permet d'ajouter à la liste de favoris
- **Onglet Favoris** :
  - Liste des destinations ajoutées par l'utilisateur
  - En cliquant sur une carte → navigation vers un écran de **détail météo**
- **Onglet Détail** :
  - Affichage météo actuelle du lieu sélectionné
  - Possibilité de retirer des favoris

Liste de sites touristiques avec coordonnées

```
export const touristicSites = [

  { name: "Mont Cameroun", lat: 4.2034, lon: 9.1708 },

  { name: "Lac Tison", lat: 6.0154, lon: 11.4311 },

  { name: "Chutes d'Ekom Nkam", lat: 5.0953, lon: 10.0814 },

  { name: "Parc de Waza", lat: 11.4171, lon: 14.5771 },

  { name: "Musée National du Cameroun", lat: 3.8667, lon: 11.5167 },

  { name: "Kribi Plage", lat: 2.9383, lon: 9.9107 },
```

```
];
```

Utiliser ces coordonnées dans les appels API météo :

```
https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&
appid={API_KEY}&units=metric
```

Écrans attendus

1. **Accueil (liste des sites)** – avec bouton « ajouter aux favoris »
2. **Favoris** – liste dynamique modifiable
3. **Détail météo** – infos météo en temps réel

Projet 3 — FitWeather

Contexte du projet

FitWeather est une application destinée aux amateurs de sport et de bien-être. Son objectif est de **suggérer automatiquement une activité physique adaptée** aux conditions météo actuelles d'un lieu donné. L'utilisateur choisit un **site sportif ou un espace public extérieur**, et l'application lui propose une activité en fonction de la météo : par exemple, courir s'il fait beau, faire du yoga en intérieur s'il pleut, etc.

Ce projet met l'accent sur l'utilisation **créative de la météo** comme facteur de suggestion, et sur la **logique conditionnelle** dans l'application mobile.

Objectifs pédagogiques

- Implémenter une **logique conditionnelle** à partir des données météo
- Créer des composants visuels pour représenter les activités suggérées
- Naviguer entre une **liste de lieux** et un écran de **suggestion d'activité**
- Organiser les données de manière modulaire
- Utiliser `fetch` pour intégrer la météo en temps réel

Fonctionnalités attendues

- **Écran d'accueil** :
  - Liste de lieux d'activités sportives (espaces publics, stades, parcs)
  - Cliquer sur un lieu → navigation vers une page de suggestion
- **Écran de suggestion** :
  - Affichage de la météo actuelle
  - Suggestion d'une activité selon température, humidité, état du ciel
  - Composant illustré de l'activité (ex : course à pied, stretching, repos)

Logique conditionnelle exemple :

```
if (weather === 'clear') return 'Course à pied'

else if (weather === 'rain') return 'Yoga en salle'

else if (temp < 20) return 'Marche active'
```

Liste de lieux sportifs avec coordonnées

```
export const sportPlaces = [

  { name: "Stade Omnisport de Yaoundé", lat: 3.867, lon: 11.52 },

  { name: "Parc de loisirs d'Ekounou", lat: 3.832, lon: 11.489 },

  { name: "Terrain de sport de Bonapriso", lat: 4.043, lon: 9.719 },

  { name: "Promenade de Kribi", lat: 2.937, lon: 9.906 },

  { name: "Espace sportif de Maroua", lat: 10.59, lon: 14.31 },

  { name: "Stade municipal de Bafoussam", lat: 5.47, lon: 10.42 },

];
```

Écrans attendus

1. **Liste de lieux d'activités** (FlatList ou ScrollView)
2. **Détail de suggestion d'activité**
    - Météo actuelle
    - Nom du lieu
    - Nom de l'activité recommandée
    - Icône ou image

Projet 4 — Climagram

Contexte du projet

Climagram est une application qui mélange **réseau social** et **météo locale**. Elle permet aux utilisateurs de publier leur "humeur météo" du jour dans une ville donnée, sous forme de petites publications (posts) contenant : ville, météo, humeur, commentaire et heure de publication. L'application affiche ensuite un **fil d'actualités météo subjectives**, dans un style proche d'un mini Instagram météo.

Ce projet met l'accent sur la **gestion d'état**, la création de composants de formulaire, et l'affichage structuré sous forme de fil (FlatList).

Objectifs pédagogiques

- Créer un **formulaire de saisie** avec champs texte, dropdown ou sélecteur
- Stocker et afficher des **publications locales**
- Utiliser un `fetch` pour afficher les données météo d'une ville sélectionnée
- Mettre en page un **fil dynamique** (FlatList)
- Gérer un **état local** sans backend ni base de données

Fonctionnalités attendues

- **Onglet "Publier"** :
  - Saisie d'une humeur météo
  - Sélection d'une ville dans une liste
  - Saisie libre d'un commentaire
  - Ajout automatique de la météo actuelle et de l'heure
- **Onglet "Fil"** :
  - Liste chronologique des posts météo des utilisateurs
  - Affichage : météo, humeur, commentaire, heure, ville
- **Onglet "Moi"** :
  - Affichage des publications que l'utilisateur a lui-même postées

Liste de villes avec coordonnées

```
export const citiesForClimagram = [

  { name: "Yaoundé", lat: 3.848, lon: 11.5021 },

  { name: "Douala", lat: 4.0511, lon: 9.7679 },

  { name: "Garoua", lat: 9.3014, lon: 13.3956 },

  { name: "Kribi", lat: 2.9383, lon: 9.9107 },

  { name: "Bafoussam", lat: 5.4778, lon: 10.4172 },

];
```

Chaque publication inclura un appel à l'API météo avec la latitude/longitude de la ville sélectionnée pour afficher la météo au moment du post.

Structure d'un post

```
{
```

```
  id: string;
  city: string;
  lat: number;
  lon: number;
  mood: "heureux" | "fatigué" | "apathique" | "énergique";
  comment: string;
  weather: { temp: number; description: string };
  timestamp: string;
}
```

Écrans attendus

1. **Écran de publication**
2. **Fil d'actualités**
3. **Mon journal météo**

Projet 5 — FarmCast

Contexte du projet

FarmCast est une application conçue pour les **agriculteurs et cultivateurs locaux**. Elle fournit des **prévisions météorologiques à 5 jours** pour des **zones rurales spécifiques** du Cameroun, afin d'aider à la planification des semis, des récoltes ou des traitements. Ce projet est centré sur la lecture de **données structurées sur plusieurs jours** issues de l'API, et leur affichage dans un tableau ou une grille claire.

Objectifs pédagogiques

- Consommer une API complexe (`/forecast`) avec plusieurs blocs temporels
- Extraire les prévisions météo quotidiennes
- Afficher des données dans une structure **multi-jour** (grille, tableau, ou liste)
- Naviguer d'une **liste de zones rurales** à une page de **prévisions météo**
- Travailler sur la lisibilité de l'interface

Fonctionnalités attendues

- **Écran de sélection** :
  - Liste de zones agricoles prédéfinies

    - Appui sur une zone → navigation vers les prévisions météo
- **Écran de prévisions** :
    - Affichage météo sur **5 jours**
    - Pour chaque jour : température min/max, météo, humidité, icône
    - Format recommandé : tableau ou cards horizontales scrollables

Liste de zones rurales avec coordonnées

```
export const ruralZones = [
  { name: "Zone agricole de Tonga", lat: 5.017, lon: 10.650 },
  { name: "Zone maraîchère de Njombé", lat: 4.637, lon: 9.640 },
  { name: "Zone rizicole de Maga", lat: 10.737, lon: 14.940 },
  { name: "Zone vivrière de Foumbot", lat: 5.514, lon: 10.634 },
  { name: "Zone horticole de Mbankomo", lat: 3.677, lon: 11.297 },
];
```

API utilisée

Endpoint :
`https://api.openweathermap.org/data/2.5/forecast?lat={lat}&lon={lon}&appid={API_KEY}&units=metric`

- Réponse : prévisions toutes les 3h pendant 5 jours
- Il faudra **regrouper les données par jour** (par exemple, en filtrant 12h00 chaque jour ou en moyennant)

Écrans attendus

1. **Liste de zones agricoles**
2. **Prévisions 5 jours**
    - Carte ou tableau pour chaque jour

Projet 6 — MoodWeather

Contexte du projet

MoodWeather est une application d'inspiration émotionnelle et musicale. Elle propose à l'utilisateur une **ambiance musicale ou une "humeur sonore"** en fonction de la météo actuelle d'un lieu donné. Par exemple, un ciel orageux pourra suggérer du jazz doux ou du lo-fi ; un soleil éclatant, une playlist pop ou dance.

Ce projet est basé sur la **créativité**, l'UX design, l'interprétation des données météo, et la **personnalisation de l'interface utilisateur**. Il permet de créer une application immersive, stylisée, tout en intégrant des appels API.

Objectifs pédagogiques

- Utiliser `fetch` pour obtenir la météo en temps réel
- Mapper des conditions météo vers des "moods" ou ambiances musicales
- Créer une **interface UX expressive**, agréable à utiliser
- Naviguer d'un lieu vers une ambiance
- Travailler l'organisation visuelle, les couleurs, les effets

Fonctionnalités attendues

- **Écran d'accueil** :
  - Liste de **villes culturelles** ou lieux "inspirants"
  - Chaque lieu mène vers une ambiance météo
- **Écran d'ambiance** :
  - Affichage météo actuelle
  - Humeur déduite automatiquement
  - Suggestion d'ambiance ou de type musical fictif
  - Design libre (icônes, fonds dégradés, musique imaginaire)

Liste de villes artistiques / culturelles avec coordonnées

```
export const moodPlaces = [
  { name: "Yaoundé - Centre culturel", lat: 3.870, lon: 11.512 },
  { name: "Buea - Université", lat: 4.152, lon: 9.243 },
  { name: "Foumban - Ville artistique", lat: 5.729, lon: 10.902 },
  { name: "Limbe - Bord de mer", lat: 4.017, lon: 9.217 },
  { name: "Bamenda - Hauts plateaux", lat: 5.963, lon: 10.159 },
];
```

Exemple de logique "météo → ambiance"

```
function getMood(weather, temp) {
  if (weather.includes('rain')) return 'Lo-fi chill';
```

```
    if (weather.includes('clear') && temp > 30) return 'Dance tropicale';

  if (weather.includes('clouds')) return 'Indie acoustique';

  if (temp < 20) return 'Jazz doux';

  return 'Ambiance zen';

}
```

Écrans attendus

1. **Liste de lieux “artistiques”**
2. **Page ambiance météo**
    - Données météo (température, description)
    - Humeur musicale
    - Style visuel personnalisé selon la météo