TP : Géolocalisation et Photos géolocalisées

1. Objectifs :

Objectif 1 – Géolocalisation

Mettre en place dans une application Ionic/Capacitor :

- l'installation du plugin **Geolocation**,
- la gestion des **permissions** Android,
- l'obtention de la **position unique** avec getCurrentPosition,
- le **suivi continu** avec watchPosition,
- l'affichage **basique des coordonnées** (latitude, longitude, précision).

Objectif 2 – Photos géolocalisées

Permettre de :

- prendre des **photos de preuve** (cours, bibliothèque, stage) avec le plugin **Camera**,
- associer chaque photo à **date/heure/position GPS**,
- afficher une **liste d'événements** avec coordonnées et photo correspondante.

2. Mise en place du projet

2.1. Installer les outils globalement

- ✓ npm install -g @ionic/cli
- ✓ npm install -g @capacitor/cli

Vérifier l'installation :

- ✓ ionic --version
- ✓ node --version
- ✓ npm --version

2.2. Créer le projet Ionic

- ✓ ionic start geotrackeur-etudiant tabs --type=angular –capacitor
- ✓ cd geotrackeur-etudiant

2.3. Installation des plugins nécessaires

Installer les plugins Capacitor Geolocation et Camera :

- ✓ npm install @capacitor/geolocation @capacitor/camera
- ✓ npx cap sync

- @capacitor/geolocation : accès GPS (position unique + suivi).
- @capacitor/camera : prise de photos, récupération d'une URL webPath pour affichage.

2.4. Build initial et ajout de la plateforme Android

- ✓ ionic build
- ✓ npx cap sync
- ✓ ionic capacitor add android

3. Configuration des permissions Android

3.1 Permissions de géolocalisation

Ouvrir `android/app/src/main/AndroidManifest.xml` et ajouter les permissions avant la balise `</manifest>` :

```
<!-- Geolocation -->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-feature android:name="android.hardware.location.gps" />
```

Ces lignes autorisent l'accès à la localisation approximative et précise, et déclarent l'usage du GPS.

3.2 Permissions caméra (photos)

Toujours dans AndroidManifest.xml, ajouter :

```
<!-- Camera -->
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Suivant le niveau de SDK, WRITE_EXTERNAL_STORAGE peut être optionnelle, mais on la laisse pour simplifier.

4. Mise en place de l'application

4.1 Logique TypeScript (Tab1)

Remplacer le contenu de src/app/tab1/tab1.page.ts par le code suivant :

```
import { Component } from '@angular/core';
import { Geolocation } from '@capacitor/geolocation';
import { Camera, CameraResultType, CameraSource } from '@capacitor/camera';
@Component({
  selector: 'app-tab1',
  templateUrl: 'tab1.page.html',
  styleUrls: ['tab1.page.scss'],
})
export class Tab1Page {
  currentPosition: any = null;
  positions: any[] = [];      // historique des positions + photos
  watchId: string | null = null;
  isTracking = false;
  // 1) Position unique
  async getCurrentPosition() {
    try {
      const permission = await Geolocation.requestPermissions();
      if (permission.location !== 'granted') {
        alert('Permission GPS refusée.');
        return;
      }
      const coordinates = await Geolocation.getCurrentPosition({
        enableHighAccuracy: true,
        timeout: 15000,
```

```
      maximumAge: 0,
    });  // Promise<GeolocationPosition>
    this.currentPosition = coordinates;
    this.positions.unshift({
      coords: coordinates.coords,
      timestamp: new Date().toLocaleString('fr-FR'),
      photo: null,
    });
  } catch (error: any) {
    alert('Erreur GPS : ' + (error?.message || 'Inconnue'));
  }
}
// 2) Suivi continu
async toggleTracking() {
  try {
    // Si déjà en suivi → on arrête
    if (this.isTracking) {
      if (this.watchId) {
        await Geolocation.clearWatch({ id: this.watchId });
      }
      this.isTracking = false;
      this.watchId = null;
      return;
    }
    const permission = await Geolocation.requestPermissions();
    if (permission.location !== 'granted') {
      alert('Permission GPS refusée.');
      return;
    }
    const watchPromise = Geolocation.watchPosition(
      { enableHighAccuracy: true, timeout: 10000 },
      (position, err) => {
        if (err) {
          console.error('Erreur watchPosition:', err);
          return;
        }
        if (position) {
          this.currentPosition = position;
          this.positions.unshift({
            coords: position.coords,
            timestamp: new Date().toLocaleString('fr-FR'),
            photo: null,
          });
        }
      }
    ); // retourne un ID de watch
```

```
    this.watchId = await watchPromise;
    this.isTracking = true;
   } catch (error: any) {
    alert('Erreur suivi : ' + (error?.message || 'Inconnue'));
   }
  }
  // 3) Prendre une photo pour une position donnée (Objectif 2)
  async takePhoto(index: number) {
   try {
    const image = await Camera.getPhoto({
     quality: 90,
     allowEditing: false,
     resultType: CameraResultType.Uri,
     source: CameraSource.Camera,
    }); // image.webPath pour <img>
    this.positions[index].photo = image.webPath;
   } catch (error: any) {
    alert('Erreur photo : ' + (error?.message || 'Inconnue'));
   }
  }
}
```

- getCurrentPosition() : obtient la position unique et l'ajoute à l'historique.

  toggleTracking() : démarre/arrête watchPosition (suivi continu).

- takePhoto() : utilise Camera.getPhoto et associe webPath à la position.

4.2 Interface HTML (affichage des coordonnées)

Remplacer src/app/tab1/tab1.page.html :

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>Traqueur Étudiant GPS</ion-title>
  </ion-toolbar>
</ion-header>
<ion-content [fullscreen]="true" class="ion-padding">
  <!-- Boutons géolocalisation -->
  <ion-button expand="block" color="primary" (click)="getCurrentPosition()">
    Position actuelle
  </ion-button>
  <ion-button expand="block" [color]="isTracking ? 'danger' : 'success'"
(click)="toggleTracking()">
    {{ isTracking ? ' Arrêter le suivi' : ' Démarrer le suivi' }}
  </ion-button>
```

```
  <!-- Affichage position actuelle -->
  <ion-card *ngIf="currentPosition">
    <ion-card-header>
      <ion-card-title>Position actuelle</ion-card-title>
    </ion-card-header>
    <ion-card-content>
      <p><strong>Latitude :</strong> {{ currentPosition.coords.latitude |
number:'1.6-6' }}</p>
      <p><strong>Longitude :</strong> {{ currentPosition.coords.longitude |
number:'1.6-6' }}</p>
      <p><strong>Précision :</strong> {{ currentPosition.coords.accuracy |
number:'1.0-0' }} m</p>
    </ion-card-content>
  </ion-card>
  <!-- Historique des positions + bouton photo -->
  <ion-list>
    <ion-list-header>
      <ion-label>Historique des positions ({{ positions.length }})</ion-label>
    </ion-list-header>
    <ion-item *ngFor="let pos of positions.slice(0,10); let i = index">
      <ion-thumbnail slot="start">
        <ion-img *ngIf="pos.photo" [src]="pos.photo"></ion-img>
        <ion-icon *ngIf="!pos.photo" name="camera-outline" color="medium"></ion-
icon>
      </ion-thumbnail>
      <ion-label>
        <h3>{{ i + 1 }}. {{ pos.timestamp }}</h3>
        <p>{{ pos.coords.latitude | number:'1.6-6' }}, {{ pos.coords.longitude |
number:'1.6-6' }}</p>
      </ion-label>
      <ion-button slot="end" fill="clear" (click)="takePhoto(i)">
         Photo
      </ion-button>
    </ion-item>
  </ion-list>
</ion-content>
```

- Affiche les coordonnées et la précision pour la position actuelle.
- Pour chaque point de l'historique, un bouton **« Photo »** permet de prendre une photo et de l'associer à ce point.

5. Test et déploiement

5.1 Test en mode web (pour valider la logique)

- ✓ ionic serve

- Le navigateur demandera la permission d'accès à la localisation.
- getCurrentPosition et watchPosition fonctionnent, mais la précision est limitée (GPS simulé).

5.2 Test sur Android (GPS réel + caméra)

- ✓ ionic build
- ✓ npx cap sync
- ✓ ionic cap run android -l

- Tester sur un émulateur ou un téléphone Android avec GPS et caméra.
- Accepter les permissions de localisation et de caméra lors du premier lancement.

6. Amélioration de l'application

Proposer une amélioration de l'application et déposer la version finale sur la plateforme avec un compte rendu.