Dashboard des surfaces aquatiques Sabkha Zarzis

1. Aperçu du projet

1.1 Contexte

Le projet vise à visualiser l'évolution temporelle des surfaces d'eau dans la **Sabkha de Zaris** à l'aide d'un tableau de bord géospatial interactif. La solution repose sur **TerriaMap / TerriaJS** pour la visualisation et une API backend personnalisée pour fournir des données géospatiales temporelles.

Chaque instantané (snapshots) de surface de l'eau est fourni sous forme de fichier GeoJSON représentant une **date unique**. Ces instantanés doivent être consolidés dans un **jeu de données temporel** pour permettre la visualisation des chronologies.

1.2 Objectifs

- Visualisez les surfaces d'eau sur **Sabkha de Zarzis** sur une carte interactive
- Permettre **l'exploration basée sur la** chronologie de l'évolution de la surface de l'eau
- Prise en charge **du téléversement des GeoJSON** et **de la consommation de données basée sur API**
- Fournir un backend capable de fusionner et de servir GeoJSON temporel
- Garantir la scalabilité et la précision des données géospatiales temporelles

1.3 Utilisateurs ciblés

- Chercheurs environnementaux
- Analystes en hydrologie et climatologie
- Spécialistes des SIG
- Décideurs et institutions publiques

2. Exigences fonctionnelles

2.1 Visualisation cartographique

- Afficher les géométries de surface de l'eau (Polygone / MultiPolygone)
- Carte centrale de la Sabkha de Zarzis
- Autoriser l'inspection panoramique, zoomée et des fonctionnalités
- Style de support (couleur, opacité) pour les plans d'eau

2.2 Fonctionnalité de la chronologie

- Afficher automatiquement une timeline lorsque les données temporelles sont chargées
- Autoriser:
    - Sélection manuelle de l'heure
    - Lecture/Pause Animation
    - Navigation étape par étape
- Filtrez les caractéristiques visibles en fonction du temps sélectionné

2.3 Modes de chargement des données

2.3.1 Téléversement GeoJSON local

- L'utilisateur peut télécharger un ou plusieurs fichiers GeoJSON
- Le système valide :
    - Structure GeoJSON
    - Présence d'informations sur le temps
- Les fichiers téléchargés sont fusionnés en une seule couche compatible avec la timeline

2.3.2 Consommation d'API en backend

- Dashboard peut charger les données directement depuis les points de terminaison backend
- Le backend retourne le GeoJSON fusionné, sensible au temps

2.4 Fonctionnalités du tableau de bord

- Bascule de calque (activé/désactivé)
- Fenêtre contextuelle de fonctionnalités montrant les attributs :
    - Date
    - Surface
    - Classification (lagune, zone type, etc.)
- Panneau de statistiques optionnelles :
    - Surface totale de l'eau par date
    - Évolution au fil du temps

3. Spécification des données

3.1 Structure GeoJSON

Chaque caractéristique **doit** inclure une propriété temporelle.

```
{
  "type": "Feature",
  "geometry": {
    "type": "MultiPolygon",
    "coordonnées": [...]
  },
  "propriétés": {
    "date": "2025-07-18",
    "surface": 4137.16,
    "classe": "Lagune",
    "zone_type": "autre"
  }
}
```

3.2 Règles temporelles

- Format de date : ISO 8601 (`YYY-MM-DD`)
- Le temps est stocké **par fonctionnalité**, pas par fichier
- Toutes les fonctionnalités appartenant au même instantané partagent la même date

3.3 Hypothèses au niveau du fichier

- Chaque fichier GeoJSON d'entrée représente **une date**
- La date est dérivée de :
    1. `Propriété date` (préférentielle)
    2. Nom de fichier (de secours, validé)

4. Spécification backend

4.1 Responsabilités

- Lire plusieurs fichiers instantanés GeoJSON
- Propriétés d'injection ou de validation temporelle
- Fusion des fonctionnalités en une seule collection
- GeoJSON avec temps de service vers le frontend
- Persistez optionellement les fichiers téléchargés

4.2 Points d'accès API

4.2.1 Obtenir des surfaces d'eau fusionnées

`GET /api/water-surfaces`

Réponse :

```
{
  "type": "FeatureCollection",
  "features": [ ... ]
}
```

Comportement :

- Agrège tous les instantanés disponibles
- Inclure la propriété **`date`** sur chaque fonctionnalité

4.2.2 Obtenir des dates disponibles

`GET /api/water-surfaces/dates`

Réponse :

```
["2025-06-15", "2025-07-01", "2025-07-18"]
```

4.2.3 Téléversement des fichiers GeoJSON

```
POST /api/water-surfaces/upload
Type de contenu : multipart/form-data
```

Comportement :

- Accepter un ou plusieurs fichiers
- Valider la structure et la date
- Stocker les fichiers sur disque ou base de données

4.3 Logique de traitement en arrière-plan

1. Scanner le répertoire GeoJSON
2. Pour chaque fichier :
    - Parse GeoJSON
    - Date d'extrait
    - Injectez la date dans toutes les fonctionnalités
3. Fusionner toutes les fonctionnalités

4. Retourner la collection unifiée de fonctionnalités

4.4 Stratégie de stockage

Option A : Système de fichiers (phase initiale)

```
/data/water-surfaces/
├── 2025-06-15.geojson
├── 2025-07-01.geojson
└── 2025-07-18.geojson
```

Option B : Base de données (évolutive)

- Post-GIS
- Colonnes du tableau :
    - Géométrie
    - Date
    - Surface
    - Métadonnées

5. Spécification frontend (TerriaJS)

5.1 Composants de base

- Instance TerriaMap
- Configuration du catalogue
- Chronologie (Terria intégré)

5.2 Exemple de configuration de catalogue

```
{
  "type": "geojson",
  "name": "Zarzis Sabkha Water Surfaces",
  "url": "http://localhost:8080/api/water-surfaces",
  "timeProperty": "date"
}
```

5.3 Comportement de la chronologie

- Apparaît automatiquement lorsque `timeProperty` est défini
- Filtres les fonctionnalités en fonction du temps sélectionné
- Supporte l'animation

6. Règles de validation

6.1 Validation GeoJSON

- JSON valide
- Types GeoJSON corrects
- Tableau de `caractéristiques` non vide

6.2 Validation temporelle

- `date` existe sur toutes les caractéristiques
- Format ISO-8601
- Dénomination cohérente des propriétés

7. Gestion des erreurs

- Date manquante → rejeter le fichier ou déduire à partir du nom du fichier
- GeoJSON invalide → retour 400 mauvaise demande
- Jeu de données vide → retourner FeatureCollection vide

8. Considérations de sécurité

- Limites de taille de téléchargement de fichiers
- Validation de type MIME
- Aucune exécution arbitraire de fichiers

9. Stratégie de test

9.1 Tests unitaires

- Analyse GeoJSON
- Logique d'extraction de dates
- Correction de la fusion

9.2 Tests d'intégration

- Validation de la réponse API
- Apparition dans la chronologie dans Terria

9.3 Tests manuels

- Téléverser plusieurs instantanés
- Lecture de l'animation de la timeline
- Inspecter les attributs de la caractéristique

10. Considérations de déploiement

- Backend déployé en tant que service REST
- Hébergement frontend statique
- Configuration CORS pour l'accès API

11. Critères de réussite

- La chronologie apparaît et s'anime correctement
- Les surfaces de l'eau changent avec le temps
- Le système gère les nouveaux instantanés sans modification de code

12. Améliorations futures

- Prise en charge de la plage de temps (début/fin)
- Graphiques statistiques liés à la chronologie
- Superpositions d'images satellites
- Filtrage de dates défini par l'utilisateur
- Optimisations de performance (pavage, tuiles vectorielles)