Guide de déploiement du site CREPSITE

🔗 Site en ligne : https://pierregb1.github.io/crepsite/

1. Introduction

Ce document explique comment créer et héberger un site web interactif permettant d'afficher dynamiquement des graphiques Python générés depuis des modèles climatiques. Le projet s'appuie sur deux plateformes : GitHub Pages pour le front-end (HTML/CSS/JS) et Render pour le back-end (Python/Flask).

2. Hébergement Front-End : GitHub Pages

GitHub Pages permet d'héberger gratuitement des fichiers HTML/CSS/JS depuis un dépôt public GitHub.

• Il suffit de placer les fichiers HTML à la racine du dépôt ou dans un dossier nommé `/docs`.

• Activez GitHub Pages dans les paramètres du dépôt (Settings > Pages > Source : branch `main`, folder `/root`).

3. Hébergement Back-End : Render (Flask)

Render est utilisé pour exécuter le code Python des modèles lorsque l'utilisateur clique sur un bouton sur le site.

Étapes :

1. Créez un compte sur https://render.com et créez un nouveau Web Service.

2. Déposez tous les fichiers Python (ex : `main.py`, `modele2p.py`...) + `requirements.txt` + `render.yaml` dans un dépôt GitHub.

3. Dans le fichier `main.py`, définissez les routes Flask pour chaque modèle (`/modele2`, `/modele3`, etc.).

4. Render exécute le serveur Flask avec Gunicorn (commande définie dans `render.yaml`).

4. Exemple de route Flask

Le serveur Flask est hébergé sur render ce qui permet d'utiliser des codes python.

```python
@app.route('/modele2', methods=['GET'])
def modele2():
    lat = float(request.args.get('lat'))
    long = float(request.args.get('long'))
    image_path = generate_model2_graph(lat, long)
    return send_file(image_path, mimetype='image/png')
```

5. Exemple HTML (bouton pour appeler le modèle)

```html
<form action="https://crepsite.onrender.com/modele2" method="get">
  Latitude: <input name="lat" value="48.85">
  Longitude: <input name="long" value="2.35">
  <input type="submit" value="Afficher le graphique">
</form>
<img src="https://crepsite.onrender.com/modele2?lat=48.85&long=2.35" alt="Graphique">
```

6. Conclusion

En combinant GitHub Pages pour l'interface statique et Render pour le traitement dynamique en Python, on obtient un site interactif, léger, et évolutif, idéal pour visualiser les résultats de simulations scientifiques.