**1. Analyse des besoins et définition des fonctionnalités**

* **Page d'authentification** : Un espace sécurisé où les gérants et toi (le DG) pourrez vous connecter. Chaque gérant a son propre compte.
* **Formulaire de saisie de données** : Les gérants entreront quotidiennement les informations sur l'entrée, la sortie et le stock d'essence et de gasoil.
* **Tableau de bord pour le DG** : Un tableau de bord où tu pourras consulter les rapports des gérants, suivre l'évolution des stocks et vérifier les entrées/sorties.
* **Base de données** : Stockage des informations dans une base de données pour faciliter les requêtes, les rapports et les exports.
* **Notifications** : Rappels automatiques pour que les gérants soumettent leurs rapports.

**2. Choix des technologies**

Voici les technologies que je recommande pour ce projet :

* **Frontend (interface utilisateur)** :
  + **HTML/CSS** pour la structure et le style de la page.
  + **JavaScript** avec **React** pour une interface interactive et réactive.
  + **Bootstrap** pour rendre le site responsive et moderne.
* **Backend (serveur et logique)** :
  + **Php** pour gérer les requêtes HTTP et la logique côté serveur.
* **Base de données** :
  + **MySQL** pour stocker les rapports et les données des stations.
* **Notifications et rappels** :
  + Utilisation de **cron jobs** (pour les rappels automatiques) et de services comme **Nodemailer** pour l’envoi d’emails ou **Twilio** pour les notifications par SMS.

**3. Structure du projet**

Voici un exemple de structure de dossier pour ton projet web :

bash

Copier le code

**4. Fonctionnalités à développer**

* **Authentification** : Implémenter la gestion des utilisateurs avec des rôles (par exemple, gérant ou DG).
* **Gestion des rapports** : Créer des formulaires où les gérants peuvent entrer les données chaque jour.
* **Tableau de bord** : Permettre au DG de consulter les rapports sous forme de tableau ou de graphiques.
* **Alertes et notifications** : Ajouter un système de rappels pour les gérants et des alertes pour les anomalies dans les données.

**5. Sécurité**

Assure-toi que toutes les informations sensibles (comme les identifiants des utilisateurs et les rapports) sont protégées :

* Utilise **HTTPS** pour sécuriser les communications.
* **Hashage des mots de passe** avec un algorithme comme **bcrypt**.
* Protéger les routes sensibles avec des **middleware** de vérification des rôles d'utilisateur.

**6. Déploiement**

Une fois que ton site est prêt, il faut le déployer. Tu peux utiliser des services comme :

* **Heroku** ou **Vercel** pour le déploiement de ton backend et frontend.
* **DigitalOcean** ou **AWS** pour une solution plus robuste si nécessaire.

**7. Tests**

Teste bien toutes les fonctionnalités avant de mettre en production :

* Tests unitaires pour vérifier la logique du backend (utilisation de **Mocha** ou **Jest**).
* Tests d'intégration pour vérifier que les différentes parties du site fonctionnent ensemble.

8. Taches par importance

**Commencons d’abord par la conception pour la gestion de vente.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Numéro | Tache | Date debut | Durée | Priorité |
|  |  |  |  |  |
| 1 | Gestion de vente(Front et back end) |  | ………………… | 1 |
| 2 | Gestion de stock(Front et back end) |  | ………………… | 2 |
| 3 | Autres(Front et back end) |  | ………………… | 3 |
| 4 | Gestion des utilisateur(coté Admin) (Front et back end) |  | ………………… | 4 |
| 5 | Tableau de bord avec les boutons(Admin et gerant) |  | ………………… | 5 |
| 6 | Authentification |  | ………………… | 6 |

Pour un début, on commence par Gestion de vente jusqu’à terminé et après on entame la tache avec la seconde priorité et ainsi de suite.