Examen de Fin de Formation (*Session Blanche*)

Préparer par : M. DAIF Othmane

Secteur : Digital et Intelligence Artificielle

Niveau : Technicien Spécialisé

Filière : Développement Digital — Option Web Full Stack

Durée : 4 h 00 **Barème :** /100

Consignes

- Lisez la totalité du sujet et le préliminaire avant de commencer.
- Répondez à toutes les questions ; une réponse cohérente est toujours valorisée.
- Pour les exercices de calcul (PERT), la méthode correcte est prise en compte même si le résultat final est faux ; justifiez toujours votre raisonnement.
- Soignez la lisibilité de vos schémas et de votre code.

Préliminaire — Contexte du sujet

La société **« SmartParking »** gère, pour le compte d'une ville, des parkings publics intelligents répartis par zones. Chaque parking est équipé d'un capteur qui mesure en continu son taux d'occupation et remonte l'état de sa batterie. Lorsqu'un capteur présente une défaillance, un agent enregistre une anomalie. Chaque zone applique un tarif horaire de stationnement. La société souhaite informatiser la supervision de son réseau de parkings.

Schéma du modèle logique de données (MLD) :

Zone(id, nom, ville, tarif_horaire)

Capteur(id, numero_serie, etat_batterie, nombre_mesures, statut)

Parking(id, nom, #id_zone, #id_capteur, nb_places, taux_occupation, date_derniere_maintenance, date_prochaine_maintenance)

Anomalie(id, #id_parking, #id_capteur, #id_agent, date_signalement, description, resolue)

Agent(id, nom, telephone, email)

NB : la clé primaire est soulignée ; la clé étrangère est précédée du signe #. Une zone regroupe plusieurs parkings ; chaque parking est équipé d'un capteur ; un parking peut faire l'objet de plusieurs anomalies.

Partie Théorique (40 points)

Dossier 1 : Création d'une application Cloud native (8 points)

1- À quoi sert l'outil Postman ? (2 pts)

2- Donner la commande permettant de télécharger une image Docker depuis Docker Hub. (2 pts)

3- Donner la commande permettant de lister les images Docker présentes en local. (2 pts)

4- Citer les trois types de services cloud (« as a Service »). (2 pts)

Dossier 2 : Préparation d'un projet web (6 points)

Donner le diagramme de cas d'utilisation correspondant au système suivant : (6 pts)

Le système SmartParking sera utilisé par trois acteurs : le Citoyen, l'Agent et l'Administrateur.

- Le citoyen peut consulter la liste des parkings disponibles avec leur taux d'occupation, et payer son stationnement en ligne.
- L'agent peut signaler une anomalie sur un capteur, consulter la liste des parkings et marquer une anomalie comme résolue.
- L'administrateur peut faire tout ce que fait l'agent, et peut en plus gérer les parkings et les zones (ajouter, supprimer) et consulter les statistiques d'occupation.
- L'accès au système est sécurisé : chaque utilisateur doit s'authentifier pour accéder à ses fonctionnalités.

Dossier 3 : Approche Agile et DevOps (7 points)

1- Citer les trois rôles de la méthode Scrum. (1,5 pt)

2- On veut mesurer la qualité de notre code avec SonarQube : quels types de défauts permet-il d'identifier ? (1,5 pt)

3- On travaille avec l'outil Git :

a- Donner la commande permettant d'enregistrer l'état actuel du code (commit). (1 pt)

b- Donner la commande permettant de fusionner une branche dans la branche actuelle. (1 pt)

4- C'est quoi l'intégration continue (CI) dans une chaîne DevOps ? (2 pts)

Dossier 4 : Gestion de données NoSQL — MongoDB (11 points)

1- Créer la base de données « ParkingDB » et la collection « parkings », puis insérer le document suivant : (3 pts)

```
{
  "_id": "p1",
  "nom": "Parking Centre",
  "nb_places": 200,
  "taux_occupation": 72,
  "statut": "Disponible",
  "zone": { "id_zone": "1", "nom": "Centre-ville", "tarif_horaire": 10 },
  "capteurs": [
    { "_id": "c1", "numero_serie": "CAP-77", "etat_batterie": 88 }
  ]
}
```

On suppose que la collection « parkings » contient un ensemble de documents. Écrire les requêtes MongoDB permettant de :

2- Afficher les parkings triés par taux d'occupation décroissant. (2 pts)

3- Afficher le nombre de parkings dont le statut est « Saturé ». (2 pts)

4- Mettre à jour le statut du parking d'identifiant « p3 » à la valeur « Hors service ». (2 pts)

5- Afficher le taux d'occupation moyen des parkings pour chaque zone. (2 pts)

Dossier 5 : Web dynamique PHP (8 points)

On dispose d'un tableau PHP **$parkings** où chaque élément est un tableau associatif contenant les clés **nom**, **taux_occupation** et **statut**.

1- Écrire un script PHP qui parcourt le tableau $parkings et affiche le nombre de parkings saturés (taux d'occupation supérieur à 90). (4 pts)

2- Écrire un script PHP qui calcule et affiche le taux d'occupation moyen de l'ensemble des parkings. (4 pts)

Partie Pratique (60 points)

Dossier 1 : Gestion des données MySQL (12 points)

En vous basant sur le MLD du préliminaire, écrire les scripts MySQL répondant aux questions suivantes :

1- Écrire le script de création de la table « Parking ». (2 pts)

2- Écrire la requête qui affiche, pour chaque zone, son nom et le taux d'occupation moyen de ses parkings. (2 pts)

3- Écrire le trigger « TR_batterie » qui contrôle, lors de l'insertion d'un capteur, que la valeur de « etat_batterie » est comprise entre 0 et 100. (2 pts)

4- Écrire la fonction « fct_batterieFaible » qui reçoit l'identifiant d'un capteur et retourne 1 si l'état de sa batterie est inférieur à 20, sinon 0. (2 pts)

5- Écrire le trigger « TR_anomalie » qui empêche l'insertion d'une anomalie si le capteur concerné a une batterie critique, en utilisant la fonction « fct_batterieFaible » créée à la question précédente. (2 pts)

6- Gestion des utilisateurs et des rôles : (2 pts)

a- Créer le rôle « RoleAgent ».

b- Donner les droits d'ajout, de modification et de suppression sur les tables « Parking » et « Anomalie » au rôle « RoleAgent ».

c- Créer l'utilisateur « omar » avec le mot de passe « 1234 ».

d- Attribuer le rôle « RoleAgent » à l'utilisateur « omar ».

Dossier 2 : Développement Front-End — React et Redux (24 points)

Soit l'état initial du store Redux suivant (le store, les actions et les reducers sont déjà créés ; il ne vous est pas demandé de les écrire) :

```
InitialState = {
  parking: {
    id: 5,
    nom: "Parking Centre",
    nb_places: 200,
    taux_occupation: 72,
    statut: "Disponible",
    date_derniere_maintenance: "01/05/2026",
    date_prochaine_maintenance: "01/08/2026",
    Zone: { id: 1, nom: "Centre-ville", ville: "Rabat", tarif_horaire: 10 },
    CapteurPrincipal: {
      id: 30, numero_serie: "CAP-77",
      etat_batterie: 88, statut: "En service"
    }
  },
  Agents:        [ { id: ..., nom: ... }, ... ],
  Parkings:      [ { id: ..., nom: ... }, ... ],
  StatutParking: [ "Disponible", "Saturé", "Hors service" ]
};
```

1- Écrire la composante « DetailsParking.js » qui lit les informations du parking depuis le store Redux (en utilisant useSelector) et affiche : le nom du parking, son taux d'occupation, son statut, le nom de sa zone et son tarif horaire, ainsi que le numéro de série et l'état de la batterie du capteur. (4 pts)

2- Écrire la composante « Menu.js » qui définit, à l'aide de NavLink, les liens menant aux composantes : « Détails Parking » (/DetailsParking), « Liste Parkings » (/ListeParkings) et « Calcul Tarif » (/Tarif). (4 pts)

3- Écrire le code de « App.js » qui : définit le routage (BrowserRouter, Routes, Route), appelle la composante Menu.js, et intègre le Provider du store Redux. (4 pts)

4- Écrire une composante « CalculTarif.js » permettant de saisir le nombre d'heures de stationnement. Au clic sur le bouton « Calculer », le montant à payer est affiché ; il est égal au nombre d'heures multiplié par le tarif horaire de la zone (tarif_horaire = 10 DH). (4 pts)

5- Initialiser la variable du state « parkings » de la composante principale App.js en appelant l'API du back-end suivante : (4 pts)

Méthode HTTP : GET

URL de l'API : http://localhost:8000/parkings

Résultat : un tableau d'objets parking au format JSON (id, nom, nb_places, taux_occupation, statut).

6- Créer la composante « ListeParkings.js » qui reçoit le tableau des parkings et l'affiche sous forme de liste à puces (nom, taux d'occupation et statut de chaque parking). (4 pts)

Dossier 3 : Développement Back-End — Laravel (24 points)

1- Donner les commandes de création des fichiers de migration : « create_zones_table », « create_capteurs_table », « create_parkings_table ». (1,5 pt)

2- Donner le code des méthodes up() et down() du fichier de migration de la table « parkings » (nom : chaîne ; nb_places : entier ; taux_occupation : entier avec la valeur par défaut 0 ; id_zone et id_capteur : clés étrangères vers les tables zones et capteurs). (3 pts)

3- Donner la commande permettant d'appliquer les migrations. (1 pt)

4- Donner les commandes de création des modèles : Zone, Capteur, Parking. (1,5 pt)

5- Donner le code du modèle « Parking » sachant qu'un parking appartient à une zone, qu'un parking est équipé d'un capteur, et qu'un parking possède plusieurs anomalies (préciser la propriété fillable et les relations). (4 pts)

6- Créer le contrôleur de ressource « ParkingController » en écrivant les méthodes index (afficher tous les parkings), store (ajouter un parking) et destroy (supprimer un parking). (6 pts)

7- Créer un fichier seeder pour la table « parkings » et écrire la méthode run() permettant d'ajouter un parking avec des données de votre choix. (3 pts)

8- Donner la route de type resource permettant d'associer toutes les actions du ParkingController. (2 pts)

9- Donner le code de la méthode du modèle « Zone » permettant de récupérer la liste de tous les parkings d'une zone donnée. (2 pts)