Partie Théorique / 40 pts

Dossier 1 : Création d'une Application Cloud native (8 pts) :

1. Quels sont les composants principaux de Docker ? **(2pts)**
2. Comment installer Express.js ? **(2pts)**
3. Comment afficher les images Docker ? **(2pts)**
4. Comment fonctionne RabbitMQ ? **(2pts)**

Préliminaire :

La société « **FastFoodDelivery**» fournit à ses employés des vélos électriques pour assurer la livraison des colis auprès de ses clients. Pour garantir la continuité et la qualité de ses services, elle a décidé de mettre en place un système de suivi de l'état des batteries de ces vélos. Ce système permettra d'anticiper le remplacement des batteries avant qu'elles ne se détériorent ou tombent en panne.

Velo(id, #id_batterie, date_derniere_maintenance, date_prochaine_maintenance)
Batterie(id,numero_serie, capacite, sante_batterie, nombre_cycles, statut)
Changement(id, #id_velo, #id_ancienne_batterie, #id_nouvelle_batterie, date_changement, #id_technicien,raison)
Technicien(id, nom, telephone, email, specialite)

NB :La clé primaire est écrite en souligné, la clé étrangère est écrite en #

La clé primaire de la table 'Batterie' est présente deux fois comme clé étrangère dans la table 'Changement' : id_ancienne_batterie, id_nouvelle_batterie

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

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

Le système sera utilisé par deux types d'utilisateurs : **l'administrateur** et **le technicien**

Le technicien peut **signaler** qu'une batterie a été réparée, **enregistrer** les remplacements de batteries et **rédiger** des rapports techniques pour les interventions effectuées

L'administrateur en plus des fonctions du technicien peut gérer les batteries en les **ajoutant** au système, **consulter** la liste des batteries pour surveiller leurs états et planifier des interventions, et **mettre à jour** leurs informations en cas de changement.

L'accès au système est sécurisé, et chaque utilisateur doit s'authentifier pour accéder à ses fonctionnalités respectives.

Scanned with CamScanner

Dossier 3 : Approche Agile (7 pts)

Soit la liste des tâches suivantes :

Tâche	Durée En jours	Tâches Antérieurs	S(n)
A	2	-	B
B	1	A	C, D
C	3	B	E
D	4	B	E
E	2	C, D	F
F	1	E	-

1- Dresser le diagramme PERT. **(3pts)**
2- Trouver le Chemin critique. **(1pt)**
3- Déterminer la durée d'exécution du projet **(1pt)**
4- La tâche C a pris 5 jours au lieu de 3 jours prévus : **(2pts)**
   a. Déterminer le nouveau chemin critique.
   b. Déterminer la nouvelle durée du projet.

Dossier 4 : Gestion de données NOSQL (11 pts)

1) Créer une base de données appelée **FastFoodDelivery** et une collection nommée **batteries. (1pt)**
2) Insérer les documents suivants dans la collection batteries **: (2pts)**

```
[
    {"_id":"1", "numéro_série":"s123", "capacité":"50%",
    "santé_batterie":"bonne", "nombre_cycles":500,"statut":"en service"},

     {"_id":"2", "numéro_série":"s345", "capacité":"70%",
     "santé_batterie":"moyenne", "nombre_cycles":600, "statut":"en panne"}
]
```

3) Ecrire une requête qui Modifier le champ **santé_batterie** à « faible » pour toutes les batteries dont le statut est **en panne** et la **capacité** est inférieure à 50%. **(2 pts)**
4) Afficher les attributs **numéro_série, capacité** et **statut** des batteries dont le nombre de cycle est entre 600 et 1000 classées par la capacité en ordre décroissant. **(2 pts)**
5) Afficher la moyenne des nombres de cycle des batteries par statut. **(2pts)**
6) Supprimer les batteries dont le statut est **en panne** et la capacité est inférieure à 20%. **(2pts)**

Dossier 5 : Web Dynamique PHP (8 pts)

1) Donner le code PHP de la création de la classe Batterie avec le champ capacité définit privée **(1 pt)**
2) Implémenter le getter **getCapacité** et le setter **setCapacité** pour la propriété capacité **(1pt)** :
3) Ajouter à la classe Batterie une méthode **verifierEntretienRequis,** Cette méthode doit vérifier si l'entretien est requis pour la batterie selon les cas suivant :
   - *Retourner **True** (entretien requis) si la capacité est inférieure ou égale à 30%, ou Le nombre de cycles est supérieur ou égal à 800.*
   - *Sinon, retourne **False** (entretien non requis).* **(2pts)**
4) Créer un tableau de deux objets Batteries avec des données de votre choix. **(2pts)**
5) Parcourir le tableau de la question 4 pour afficher le nombre de batterie nécessitant un entretien en utilisant la méthode **verifierEntretienRequis. (2pts)**

Scanned with CamScanner

Partie Pratique / 60 pts

Dossier 1 : Gestion des données MySQL (12 pts)

En se basant sur le schéma de base de données ci-dessus, écrire les scripts MySQL qui répondent aux questions suivantes :

1- Ecrire le script de la création de la table **'Vélo'. (2pts)**
2- Modifier la table **'Vélo'** en ajoutant la colonne **'nb_batteries'** qui est de type entier positif et non nul. **(2pts)**
3- Ecrire un trigger **TR1** qui contrôle la valeur de la colonne **'sante_batterie'** qui doit être un nombre compris entre 0 et 100, lors de l'ajout d'une nouvelle batterie. **(2pts)**
4- Ecrire un trigger **TR2,** qui modifie la valeur de la colonne statut de **'En Utilisation'** à **'Retiré'.** Lors d'un ajout dans la table **'Changement' (2pts)**
5- Ecrire une fonction **FCT1** qui retourne le nombre total de batteries consommées par un vélo dont l'id est passé en paramètre. **(2pts)**
6- Gestion des utilisateurs/Rôles : **(2pts)**
   a. Créer le rôle **'RoleTechnicien'**
   b. Donner les droits d'ajout, suppression et modification sur les tables **'Vélo'** et **'Changement'** au rôle 'Role*Technicien*'
   c. Créer l'utilisateur 'ali' avec le mot de passe 0000
   d. Attribuer le rôle '*RoleTechnicien*' à l'utilisateur '*Ali*' déjà crée

Dossier 2 : Développement Front End (24 pts)

Soit l'état initial du ***store*** redux :

```
InitialState = {
velo :{
 id: 14,
 Matricule: "1 A 4321",
 Date_derniere_maintenance: "01/06/2025",
 Date_prochaine_maintenance: "01/08/2025",
 BatterieUtilisee: {
      Id : 135,
      Capacite : 73,
            Numero_serie : "BAT-202",
      sante_batterie: 95,
      nombre_cycles: 2127,
      statut: "En Utilisation",
       }
,Techniciens :[ {id :...,Nom :...}, {id :...,Nom :...},...],
,Batteries :[ {id :...,Numero_serie :...}, {id :..., Numero_serie:...},...],
,StatutBatterie :[ "En Stock","En Utilisation","Retiré"]
};
```

Remarque : Il n'est pas demandé à vous de créer le store Redux (actions, reducer, store).

Scanned with CamScanner

1. Ecrire la composante **DetailsVelo.js** qui lit les informations depuis le store Redux -En utilisant useSelector- et les afficher comme indiqué le *figure 1*. **(4pts)**

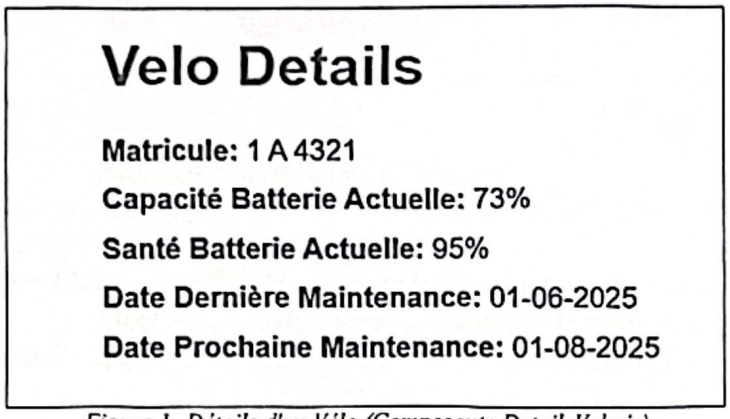


Figure 1- Détails d'un Vélo (Composante DetailsVelo.js)

2. Ecrire le code de la composante **AjouterChangement.js** (voir figure 2) qui ajoute un changement de la batterie dans la base de données à travers l'API et en utilisant les données du sotre pour remplir le formulaire d'un nouveau changement de batterie : **(6pts)**

Méthode http	POST
URL de l'API	http://localhost:8000/api/AjouterChangement
Structure Objet	{ " id_velo":..., "id_ancienne_batterie":..., "id_nouvelle_batterie":..., "date_changement":..., "id_technicien":..., "raison":... }

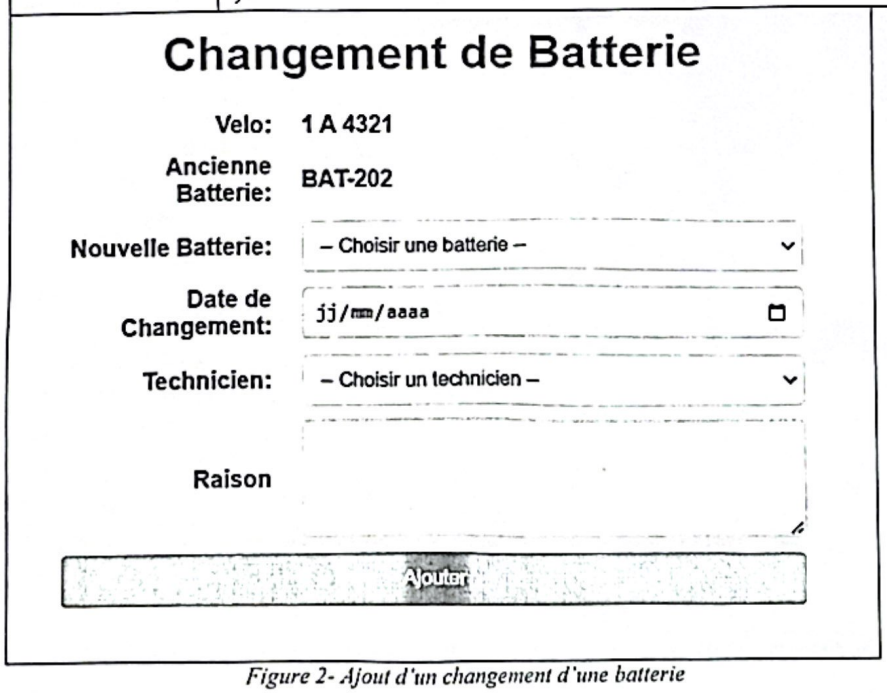


Figure 2- Ajout d'un changement d'une batterie

- Tous les champs sont obligatoires
- Les valeurs du champs (vélo, ancienne batterie,liste batteries,liste techniciens) sont lues depuis le store redux

Scanned with CamScanner

3. Créer la composante **ListeBatteries**.js qui lit la liste des batteries depuis le store redux et l'affiche sous la forme de la *figure 3* : **(6pts)**

Liste Batteries

Choisir le Statut de la Batterie:

En Stock | Filtrer

Id	Numero Serie	Statut	Nombre Cycles
232	BAT-11	En Stock	6
16	BAT-19	En Stock	24

2 batteries Trouvées

La moyenne des nombres de cycles est 15

Figure 3 : Liste des Batteries - Lue depuis le store Redux

- La liste déroulante (selecte) lit les données depuis le store redux (StatutBatterie).
-Apres clique sur le bouton 'Filtrer' Les données sont filtrées selon l'option choisie au niveau de la liste déroulante.
-En Bas du tableau, on affiche le nombre total d'éléments trouvés ansi que la moyenne des nombres de cycles

4. Ecrire la composante **Menu.js** qui définit les liens qui mènent vers les composantes : **(4pts)**

Lien	Composante	Titre
/AjouterChgmt	AjouterChangement.js	Nouveau Changement
/ListeBatteries	ListeBatteries.js	Liste Batteries
/DetailsVélo	DetailsVelo.js	Détails Vélo

5. Ecrire le code **App.js** qui : **(4pts)**
   - Définit le routage (BrowserRouter, Routes, …)
   - Appel la composante Menu.js crée ci-dessus.
   - Integer le provider du store redux

Scanned with CamScanner

Dossier 3 : Développement Back End (24 pts)

1) Donner la commande de la création de la migration pour la table batteries. **(1pt)**
2) Donner le code des méthodes ***up*** et ***down*** du fichier de migration de la table ***batteries*** en définissant le champ ***numéro_série*** comme unique et le champ ***statut*** par défaut à « En service ». **(2pts)**
3) Donner la commande pour appliquer la migration. **(1pt)**
4) Donner la commande de la création de la migration pour une modification de la structure de la table batteries. **(1pt)**
5) Donner le code de la méthode **up** et **down** pour ajouter un index sur la colonne **statut** et changer le type de donnée de la colonne **capacité** en chaine de caractères de taille 15 caractères. **(2pts)**
6) Créer les modèles des tables **batteries** et **vélos** avec leurs attributs et fonctions de relations. **(2pts)**
7) Donner la commande de la création du contrôleur **VéloController**. **(2pts)**
8) Créer la méthode **GetBatterie** dans le contrôleur **VéloController** qui retourne les informations d'une batterie d'un vélo dont l'id est passé en paramètre. **(2pts)**
9) Créer la méthode **AfficherBatterie** dans le contrôleur **VéloController** qui retourner une vue nommée **InfoBatterie (serra créer dans la question 10)** avec les données de la méthode **GetBatterie** de la question 8. **(2pts)**

10) Créer la vue **InfoBatterie** dans le contrôleur **VéloController** qui affiche les informations de la Batterie sous forme d'un tableau comme le montre l'exemple ci-dessous avec les liens modifier et supprimer. **(2pts)**

Id	Numéro série	Capacité	Santé batterie	Nombre cycle	Statut	Modifier	Supprimer
1	S1	300	Bonne	500	En service	Modifier	Supprimer

11) Créer la méthode **SuuprimerBattrie** dans le contrôleur **VéloController** qui supprime une batterie dont l'id est passé en paramètre et qui redirige l'utilisateur vers la vue **Welcome. (2pts)**

Dans ce qui suit on suppose qu'un système d'authentification est déjà installé sur le projet et que les utilisateurs sont déjà créés dans la base de données

12) Donner la commande pour créer le middleware sous le nom «**autorisationSuppression**». **(1pt)**
13) Donner le corps de la méthode **handle** pour que le middleware **autorisationSuppression** accepte les requêtes provenant de l'utilisateur « toto » dans le cas contraire on affiche un code d'erreur 403 avec le message « vous n'êtes pas autorisé à effectuer cette opération de suppression ». **(2pts)**
14) Donner le code de la route qui mène vers l'action de suppression de batterie en la protégeant par le middleware «**autorisationSuppression**». **(2pts)**

Scanned with CamScanner