Examen Blanc de Fin de Formation (Barème à titre indicative)

Métier: Développement Digital (Option : Web FullStack)

Durée : 4H30

Gestion des flux vidéo satellites et des caméras pour un événement sportif (Coupe de Monde 2026)

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

1. Conteneurisation vs Virtualisation (4 pts)

Dans le cadre de la modernisation d'une plateforme de diffusion multimédia en direct, une entreprise souhaite déployer une application de traitement de flux vidéo capable de gérer automatiquement la montée en charge lors des événements à forte audience :

1.1. Expliquer la différence entre *orchestration de conteneurs* (Kubernetes) et *virtualisation matérielle*. Laquelle est la plus adaptée pour une application de traitement de flux vidéo en direct ? (4 pts)

1.2. Vous devez déployer une base de données MySQL pour l'application **SatStream Manager** en utilisant Docker.
La commande suivante est utilisée pour créer et démarrer un conteneur MySQL en mode détaché :

```
docker run -d \
  --name satstream-db \
  -e MYSQL_ROOT_PASSWORD=root123 \
  -e MYSQL_DATABASE=satstream_db \
  -e MYSQL_USER=satstream_user \
  -e MYSQL_PASSWORD=pass456 \
  -p 3306:3306 \
  mysql:8.0
```

Questions

a. Expliquez brièvement le rôle de chaque option dans la commande ci-dessus :

- `-d`
- `--name satstream-db`
- `-e MYSQL_DATABASE=satstream_db`
- `-p 3306:3306`

b. Après avoir démarré le conteneur, vous constatez une erreur de configuration. Donnez la commande Docker pour supprimer complètement le conteneur dont l'ID est `b7f92a33` (conteneur MySQL arrêté).***(1 point)***

c. Quelle commande utiliseriez-vous pour forcer la suppression d'un conteneur en cours d'exécution ?***(0,5 point)***

d. Pourquoi est-il important d'utiliser des variables d'environnement pour les mots de passe plutôt que de les écrire en dur dans l'image Docker **?***(0,5 point)*

2. Scalabilité dans MongoDB (2 pts)

a. Expliquez la différence entre *scaling vertical* et *scaling horizontal* dans le contexte de MongoDB.
b. Quel mécanisme MongoDB utilise-t-il pour le partitionnement horizontal des données ? Décrivez-le brièvement.

3. RabbitMQ : rôle et protocole (3 pts)

a. Quel est le rôle principal d'un système comme RabbitMQ dans une architecture de microservices ?
b. Quel est le protocole natif utilisé par RabbitMQ pour la communication entre producteurs et consommateurs ?
c. Donnez un exemple concret d'utilisation de RabbitMQ dans le traitement des flux vidéo satellites.

4. Principes de l'approche DevOps (3 pts)

a. Citez et expliquez brièvement trois principes fondamentaux de la culture DevOps.
b. En quoi l'automatisation (CI/CD) est-elle essentielle dans une approche DevOps pour une application de streaming vidéo ?

Préliminaire (contexte du dossier 2 et 3)

Dans le cadre d'un événement sportif international, plusieurs caméras capturent des angles différents d'un match. Ces flux sont transmis via des satellites, puis centralisés dans un système de gestion des chaînes.
Chaque **caméra** produit un flux vidéo (débit en Mbps) et peut être fixe ou mobile.
Les **satellites** relaient ces flux avec une latence mesurée en ms.
Les **chaînes** regroupent plusieurs flux et les diffusent selon un planning.

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

Dans le cadre de la gestion des flux vidéo satellites pour un événement sportif international, le système **SatStream Manager** permet aux équipes techniques de :

- Visualiser en temps réel l'ensemble des flux vidéo provenant des différentes caméras relayées par satellites.
- Filtrer les flux par satellite, par type de caméra (fixe/mobile) ou par qualité vidéo (HD, 4K).
- Supprimer un flux défectueux ou inutilisé après confirmation.
- Consulter l'état de santé d'un satellite (latence, bande passante utilisée).
- Assigner un flux à une chaîne de diffusion (préparation du direct).

- Le Technicien vidéo est le responsable de la supervision des flux et de la qualité des diffusions.
- L'ingénieur satellite est le responsable de l'état des satellites et des assignations de bande passante.
- Le Producteur TV est le responsable du choix des flux à diffuser sur les chaînes.

➜ Proposer le **diagramme de cas d'utilisation** correspondant.

Dossier 3 : Approche Agile (15 pts)

Tâches du projet :

Tâche	Libellé	Durée (j)	Tâches antérieures
A	Étude des besoins et spécifications techniques	5	-
B	Configuration des récepteurs satellites	7	A
C	Développement du module d'ingestion des flux	10	A
D	Intégration des caméras au système	6	B, C
E	Tests de latence et qualité vidéo	4	D
F	Déploiement et formation	3	E

Début : Lundi 02/09/2026

1. Tracer le diagramme PERT avec dates au plus tôt et au plus tard. (3 pts)
2. Déterminer le chemin critique, en déduire la durée du projet en jours ouvrés (week-ends exclus). (3 pts)

3. Citer deux outils d'intégration et déploiement continu (CI/CD). (3 pts)
4. Donner les trois rôles d'une organisation Agile et les expliquer. (2 pts)
5. Différence entre méthode Agile et méthode classique (ex : cycle en V). (2 pts)
6. Donner la commande git pour fusionner la branche `feature/flux-satellite` dans `main`. (2 pts)

Dossier 4 : Gestion de données NoSQL (11 pts)

Base de données `SatelliteStreaming`.
Collection `FluxVideo` :

```
{
  "_id": "SAT-MOROCCO-01",
  "nom": "Mohammed VI-A",
  "orbite": "GEO",
  "bandeFrequence": "Ku",
  "statut": "actif",
  "dateLancement": ISODate("2022-03-15"),
  "fluxVideo": [
    {
      "idFlux": "FLUX-001",
      "camera": {
        "idCam": "CAM-101",
        "nom": "Caméra Centrale",
        "type": "fixe",
        "resolution": "4K"
      },
      "debitMbps": 85,
      "latenceMs": 120,
      "dateDebut": ISODate("2024-09-10T14:00:00Z"),
      "dateFin": ISODate("2024-09-10T16:30:00Z"),
      "qualite": "HD",
      "statutDiffusion": "en direct"
    },
    {
      "idFlux": "FLUX-002",
      "camera": {
        "idCam": "CAM-102",
        "nom": "Caméra Latérale",
        "type": "mobile",
        "resolution": "HD"
      },
      "debitMbps": 45,
      "latenceMs": 95,
      "dateDebut": ISODate("2024-09-10T14:15:00Z"),
      "dateFin": ISODate("2024-09-10T17:00:00Z"),
      "qualite": "HD",
      "statutDiffusion": "en direct"
    }
  ],
  "bandePassanteUtilisee": 130,
  "bandePassanteMax": 500
}
```

1. Afficher tous les flux dont le débit > 50 Mbps. (2 pts)
2. Compter le nombre de flux par satellite. (3 pts)
3. Ajouter un champ `compression` (booléen) à tous les documents de la collection. (2 pts)
4. Supprimer les flux dont la latence dépasse 200 ms. (4 pts)

5. a. Afficher tous les satellites dont la bande passante utilisée dépasse 70 % de leur capacité maximale. (1 pt)
6. b. Ajouter un nouveau flux vidéo à l'intérieur du tableau `fluxVideo` du satellite `SAT-MOROCCO-01`. (1 pt)
7. c. Supprimer du satellite `SAT-MOROCCO-01` tous les flux dont la latence est supérieure à 150 ms. (1 pt)
8. a. Écrire une requête d'agrégation pour compter, pour chaque satellite, le nombre total de flux vidéo actifs (`statutDiffusion = "en direct"`). (2 pts)
9. b. Écrire une requête d'agrégation pour calculer la **latence moyenne** des flux par type de caméra (`fixe` ou `mobile`). (2 pts)
10. c. Écrire une requête d'agrégation pour afficher les satellites dont la **bande passante utilisée totale** dépasse 400 Mbps, avec la liste des caméras concernées. (1 pt)

11. En utilisant `$unwind`, `$group` et `$sort`, écrivez une requête qui retourne, pour chaque satellite, les **trois flux ayant le plus haut débit (Mbps)**, triés par ordre décroissant.
    Le résultat doit contenir : `nom` du satellite, `idFlux`, `debitMbps`.

Dossier 5 : Développement d'une application Full-stack (60 pts)

Partie 1 : Base de données MySQL (12 pts)

Modèle logique :

- **Satellite** (idSat, nom, bandeFrequence, positionOrbite)
- **Camera** (idCam, nom, type, debitMoyen)
- **Flux** (idFlux, #idSat, #idCam, dateDebut, dateFin, latenceMoyenne)
- **PlanningDiffusion** (#idFlux, chaine, heureDebut, heureFin)

1. Script de création de la table `Flux`. (1 pt)
2. Afficher pour chaque satellite le nombre de caméras associées. (1 pt)
3. Lister les flux du satellite `Sat5G_1` triés par latence. (2 pts)
4. Fonction `estLatenceCritique(idFlux)` retourne VRAI si latence > 150 ms. (3 pts)
5. Trigger empêchant l'ajout d'un flux si la caméra a déjà un flux actif sur le même satellite. (3 pts)
6. Gestion utilisateurs : créer rôle `TechnicienFlux`, lui donner droits SELECT/INSERT sur `Flux`, créer utilisateur `tech1` avec mot de passe `pass123`, lui attribuer le rôle. (2 pts)

Partie 2 : Interface React.js / Redux (24 pts)

Contexte métier

Vous développez l'interface utilisateur du système **SatStream Manager**. L'application doit permettre aux techniciens vidéo de :

- Consulter la liste des flux vidéo disponibles
- Visualiser les détails d'un satellite et de ses flux associés
- Ajouter un nouveau flux vidéo à un satellite
- Supprimer un flux après confirmation

Les données sont gérées globalement via **Redux** pour assurer la cohérence entre les composants.

Structure du store Redux

Le store contient l'état suivant :

```
{
  satellites: [
    {
      id: "SAT-MOROCCO-01",
      nom: "Mohammed VI-A",
      orbite: "GEO",
      bandePassanteMax: 500,
      bandePassanteUtilisee: 130,
      flux: [
        {
          id: "FLUX-001",
          nomCamera: "Caméra Centrale",
          typeCamera: "fixe",
          debitMbps: 85,
          latenceMs: 120,
          statut: "actif"
        },
        {
          id: "FLUX-002",
          nomCamera: "Caméra Latérale",
          typeCamera: "mobile",
          debitMbps: 45,
          latenceMs: 95,
          statut: "actif"
        }
      ]
    }
  ],
  selectedSatelliteId: "SAT-MOROCCO-01",
  loading: false,
  error: null
}
```

Actions Redux définies

```
const ADD_FLUX = "ADD_FLUX";
const REMOVE_FLUX = "REMOVE_FLUX";
const SELECT_SATELLITE = "SELECT_SATELLITE";
const SET_LOADING = "SET_LOADING";
const SET_ERROR = "SET_ERROR";
```

1. Définition du store et des reducers (6 pts)

a. Écrire le **reducer principal** `satelliteReducer` qui gère les actions suivantes :

- `SELECT_SATELLITE` : met à jour `selectedSatelliteId`
- `ADD_FLUX` : ajoute un nouveau flux au satellite concerné
- `REMOVE_FLUX` : supprime un flux d'un satellite donné

b. Écrire la fonction **`configureStore`** pour créer le store Redux avec ce reducer.

c. Écrire les **action creators** pour les trois actions ci-dessus.

2. Connexion du store aux composants (6 pts)

a. Écrire le composant **`Menu.js`** (3 pts)

- Doit contenir des liens vers :
  - **`/listeFlux`** → **`ListeFlux.js`**
  - **`/detailsSatellite`** → **`DetailsSatellite.js`**
  - **`/ajouterFlux`** → **`AjouterFlux.js`**

b. Écrire `App.js` (3 pts)

- Utiliser `BrowserRouter`, `Routes`, `Route`
- Intégrer le `Provider` Redux
- Appeler le composant `Menu.js`

3. Composant `DetailsSatellite.js` (6 pts)

Ce composant doit afficher les informations du satellite sélectionné (voir figure ci-dessous).

Informations à afficher (lues depuis le store Redux) :

- Nom du satellite
- Orbite
- Bande passante utilisée / maximale (avec une barre de progression)
- Liste des flux associés avec leur débit et latence

Contraintes :

- Utiliser `useSelector` pour lire les données
- Si aucun satellite n'est sélectionné, afficher un message "Aucun satellite sélectionné"

4. Composant `AjouterFlux.js` (6 pts)

Ce composant affiche un formulaire permettant d'ajouter un nouveau flux au satellite sélectionné.

Champs du formulaire :

- `nomCamera` (texte)
- `typeCamera` (radio : fixe / mobile)
- `debitMbps` (nombre)
- `latenceMs` (nombre)

Comportement :

- Les valeurs sont stockées dans un `state` local du composant
- À la soumission, dispatcher l'action `ADD_FLUX` avec les données du formulaire
- Réinitialiser le formulaire après ajout
- Afficher une notification de succès (simple `alert` ou message temporaire)

Contrainte technique :

- Utiliser `useDispatch` et `useSelector` pour récupérer l'ID du satellite sélectionné

Partie 3 : Backend & Architecture Microservices (30 pts)

Contexte général

Le système **SatStream Manager** repose sur une architecture hybride :

- Une **base de données relationnelle (MySQL)** gérée par une API Laravel pour les données administratives (satellites, caméras, assignations).
- Un **microservice dédié aux flux vidéo** connecté à MongoDB (collection `Satellites` du Dossier 4) pour la gestion en temps réel des flux.

Partie 3.1 : API Laravel (Base de données relationnelle) – 15 pts

Questions (15 pts)

1. Création des migrations (3 pts)

Écrire les migrations Laravel pour créer les trois tables ci-dessus, en respectant :

- Clés primaires auto-incrémentées
- Clés étrangères avec contraintes d'intégrité référentielle
- Index sur `date_debut` et `statut` dans `AssignationFlux`

2. Modèles et relations Eloquent (2 pts)

Définir les modèles `Satellite`, `Camera` et `AssignationFlux` avec les méthodes de relation appropriées.

- Un satellite peut avoir plusieurs assignations de flux
- Une caméra peut être assignée plusieurs fois

3. Contrôleur `SatelliteController` – CRUD (4 pts)

Écrire un contrôleur `SatelliteController` avec les méthodes suivantes :

a. `index()` : retourne la liste de tous les satellites avec leur charge actuelle (nombre de flux actifs)
b. `store(Request $request)` : valide et crée un nouveau satellite (nom requis, orbite valide : GEO/MEO/LEO)
c. `show($id)` : retourne un satellite avec ses assignations de flux actives
d. `destroy($id)` : supprime un satellite seulement s'il n'a pas d'assignation de flux en cours

4. Routes API (2 pts)

Ajouter les routes RESTful pour `SatelliteController` dans `routes/api.php`. Utiliser le préfixe `api/v1/satellites`.

5. Requêtes avancées avec Query Builder (4 pts)

a. Écrire la requête Eloquent pour obtenir tous les satellites avec plus de 5 flux actifs (statut = "en cours"). (2 pts)

b. Écrire la requête SQL brute via `DB::select` pour obtenir la caméra la plus utilisée (nombre d'assignations) en juillet 2024. (2 pts)

Partie 3.2 : Microservice de gestion des flux vidéo (Node.js + MongoDB) – 15 pts

Contexte

Un microservice indépendant est développé en **Node.js avec Express** pour interagir directement avec la collection MongoDB `Satellites` du Dossier 4.
Ce microservice expose des endpoints REST pour la gestion fine des flux vidéo imbriqués.

Questions (15 pts)

1. Configuration de la connexion MongoDB (2 pts)

Écrire un fichier `db.js` qui :

- Utilise `MongoClient` pour se connecter à MongoDB
- Exporte une fonction `getDb()` retournant l'objet `db`
- Gère les erreurs de connexion avec un message approprié

2. Contrôleur `fluxController.js` – Opérations sur les flux imbriqués (5 pts)

Implémenter les méthodes suivantes
(avec `req.params.idSatellite` et `req.params.idFlux`) :

a. `ajouterFlux(req, res)` : ajoute un nouveau flux vidéo dans le tableau `fluxVideo` d'un satellite donné (utiliser `$push`)
b. `supprimerFlux(req, res)` : supprime un flux spécifique (utiliser `$pull` avec filtrage sur `idFlux`)
c. `mettreAJourFlux(req, res)` : met à jour un champ (ex: `debitMbps`) d'un flux sans modifier les autres (utiliser `$set` avec opérateur positionnel `$`)
d. `listerFlux(req, res)` : retourne uniquement le tableau `fluxVideo` d'un satellite (projection MongoDB)
e. `statistiquesFlux(req, res)` : retourne le nombre total de flux, le débit moyen et la latence maximale pour un satellite (utiliser `$aggregate` avec `$unwind`)

3. Routes du microservice (2 pts)

Définir les routes Express pour les méthodes ci-dessus avec les URL suivantes :

```
POST   /api/satellites/:idSatellite/flux
DELETE /api/satellites/:idSatellite/flux/:idFlux
PUT    /api/satellites/:idSatellite/flux/:idFlux
GET    /api/satellites/:idSatellite/flux
GET    /api/satellites/:idSatellite/flux/stats
```

4. Communication entre Laravel et le microservice (3 pts)

a. Dans Laravel, écrire une
méthode `syncFluxFromMicroservice()` dans `SatelliteController` qui appelle le microservice (via `Http::get()`) pour récupérer les flux d'un satellite et les enregistrer localement dans la table `AssignationFlux`. (2 pts)

b. Quel problème d'architecture peut survenir si le microservice est indisponible ? Proposez une solution de résilience. (1 pt)

5. Validation des entrées (3 pts)

Dans le microservice, ajouter une validation avec `Joi` ou `express-validator` pour l'ajout d'un flux :

- `nomCamera` : requis, chaîne de caractères
- `debitMbps` : requis, nombre entre 10 et 500
- `latenceMs` : requis, nombre entre 0 et 500

Écrire le middleware de validation correspondant.