



Projet de Synthèse N° : 01

Année : 2024-2025

Filière : Développement Digital (Option : Web FullStack)

Module 206 : Création application Cloud

Chapitre II : Gestion des données avec mongoose

Objectifs pédagogiques Opérationnelles

Développer une application web simple en utilisant Express.js, en intégrant les concepts fondamentaux tels que la gestion des routes, les middlewares et la manipulation des requêtes et réponses HTTP.

Objectifs pédagogiques intermédiaires

- ◊ Comprendre le rôle et les fonctionnalités d'Express.js
- ◊ Installer et configurer un projet Express.js
- ◊ Gérer les routes et les requêtes http
- ◊ Utiliser les middlewares
- ◊ Interagir avec une base de données
- ◊ Déployer une application Express.js

Contexte du Projet :

beIN SPORTS est un réseau international de chaînes de télévision sportives détenu par le groupe **beIN Media Group**, basé au Qatar. Il diffuse des événements sportifs majeurs (football, tennis, NBA, F1, etc.) dans plusieurs pays, notamment en France, au Moyen-Orient, en Amérique du Nord et en Asie.

Pour accéder à ses contenus premium, beIN SPORTS propose des **abonnements payants** via différentes offres (mensuelles, annuelles, forfaits famille, etc.). Une gestion efficace de ces abonnements est cruciale pour :

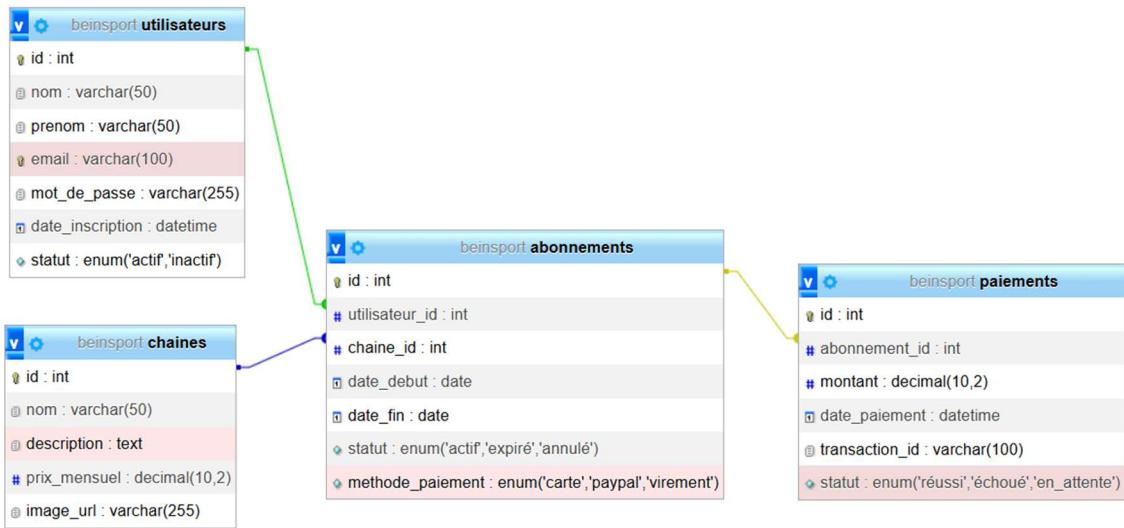
- Maximiser la rétention des abonnés
- Automatiser les processus de paiement et renouvellement
- Lutter contre le piratage et le partage illégal de comptes
- Proposer des offres personnalisées

PARTIE I : Schéma Relationnel pour la Gestion d'Abonnements BeinSport

Le Modèle Logique de Données :

UTILISATEURS (_id_utilisateur_, nom, prenom, email, mot_de_passe, date_inscription, statut)
CHAINES (_id_chaine_, nom, description, prix_mensuel, image_url)
ABONNEMENTS (_id_abonnement_, #id_utilisateur, #id_chaine, date_debut, date_fin, statut, methode_paiement)
PAIEMENTS (_id_paiement_, #id_abonnement, montant, date_paiement, transaction_id, statut)

Le Modèle physique de Données :



1. Contraintes de domaine:

- prix_mensuel > 0
- date_fin > date_debut
- email doit être valide (vérification par regex)

2. Contraintes temporelles:

- date_inscription ≤ date du jour
- date_paiement ≤ date du jour

Travail à Faire :

Ecrire le Script SQL qui permet de créer les tables tenant compte des contraintes cités ci-dessus ?

Question 1

-- Quelle est la syntaxe correcte pour créer une fonction qui calcule le nombre d'abonnés actifs pour une chaîne donnée ?

A)

```
CREATE FUNCTION compter_abonnes_actifs(id_chaine INT)
RETURNS INT
BEGIN
    DECLARE nb_abonnes INT;
    SELECT COUNT(*) INTO nb_abonnes FROM ABONNEMENTS
    WHERE id_chaine = id_chaine AND statut = 'actif' AND date_fin > CURDATE();
    RETURN nb_abonnes;
END
```

B)

```
CREATE PROCEDURE compter_abonnes_actifs(IN id_chaine INT, OUT nb_abonnes INT)
BEGIN
    SELECT COUNT(*) INTO nb_abonnes FROM ABONNEMENTS
    WHERE id_chaine = id_chaine AND statut = 'actif';
END
```

```

C)
CREATE TRIGGER compter_abonnes_actifs AFTER INSERT ON ABONNEMENTS
FOR EACH ROW
BEGIN
    UPDATE CHAINES SET abonnes_actifs = abonnes_actifs + 1
    WHERE id_chaine = NEW.id_chaine;
END

```

Question 2

Quel serait le déclencheur approprié pour mettre à jour automatiquement le statut d'un abonnement lorsque la date de fin est dépassée ?

- A) Un trigger BEFORE INSERT sur la table ABONNEMENTS
- B) Un trigger AFTER UPDATE sur la table PAIEMENTS
- C) Un événement planifié qui s'exécute quotidiennement
- D) Un trigger BEFORE SELECT sur la table UTILISATEURS

Question 3

Complétez la procédure stockée qui renouvelle un abonnement

```

CREATE PROCEDURE renouveler_abonnement(
    IN p_id_abonnement INT,
    IN p_duree_mois INT
)
BEGIN
    DECLARE date_fin_actuelle DATE;
    DECLARE nouvelle_date_fin DATE;

    -- Récupérer la date de fin actuelle
    SELECT date_fin INTO date_fin_actuelle
    FROM ABONNEMENTS
    WHERE id_abonnement = p_id_abonnement;

    -- Calculer la nouvelle date de fin
    IF date_fin_actuelle < CURDATE() THEN
        SET nouvelle_date_fin = _____;
    ELSE
        SET nouvelle_date_fin = _____;
    END IF;

    -- Mettre à jour l'abonnement
    UPDATE ABONNEMENTS
    SET date_fin = nouvelle_date_fin,
        statut = 'actif'
    WHERE id_abonnement = p_id_abonnement;
END

```

Quelles expressions complètent correctement la procédure ?

1. DATE_ADD(CURDATE(), INTERVAL p_duree_mois MONTH) et DATE_ADD(date_fin_actuelle, INTERVAL p_duree_mois MONTH)
2. CURDATE() + p_duree_mois et date_fin_actuelle + p_duree_mois
3. DATE_ADD(NOW(), INTERVAL p_duree_mois MONTHS) et DATE_ADD(date_fin_actuelle, p_duree_mois MONTHS)
4. DATE_ADD(CURDATE(), INTERVAL p_duree_mois MONTH) et DATE_ADD(date_fin_actuelle, INTERVAL p_duree_mois MONTH)

Question 4

Indiquez si ces affirmations sont vraies ou fausses concernant les triggers sur la base BeinSport :

1. Un trigger AFTER INSERT sur la table ABONNEMENTS pourrait servir à créer automatiquement le premier paiement.
2. Une fonction peut modifier les données des tables UTILISATEURS, CHAINES et ABONNEMENTS.
3. Un trigger BEFORE DELETE sur la table CHAINES pourrait empêcher la suppression d'une chaîne ayant des abonnés actifs.
4. Une procédure stockée ne peut pas appeler une autre procédure stockée.

Question 5

Quel serait le code d'un trigger qui met à jour le statut d'un abonnement en 'inactif' lorsqu'un paiement échoue (statut = 'échoué' dans la table PAIEMENTS) ?

A)

```
CREATE TRIGGER maj_statut_abonnement
AFTER UPDATE ON PAIEMENTS
FOR EACH ROW
BEGIN
    IF NEW.statut = 'échoué' THEN
        UPDATE ABONNEMENTS SET statut = 'inactif'
        WHERE id_abonnement = NEW.id_abonnement;
    END IF;
END
```

B)

```
CREATE PROCEDURE maj_statut_abonnement(IN p_id_paiement INT)
BEGIN
    DECLARE p_statut VARCHAR(20);
    DECLARE p_id_abonnement INT;

    SELECT statut, id_abonnement INTO p_statut, p_id_abonnement
    FROM PAIEMENTS WHERE id_paiement = p_id_paiement;
```

```

IF p_statut = 'échoué' THEN
    UPDATE ABONNEMENTS SET statut = 'inactif'
    WHERE id_abonnement = p_id_abonnement;
END IF;
END

```

C)

```

CREATE FUNCTION maj_statut_abonnement(p_id_paiement INT)
RETURNS VARCHAR(20)
BEGIN
    UPDATE ABONNEMENTS a JOIN PAIEMENTS p ON a.id_abonnement = p.id_abonnement
    SET a.statut = 'inactif'
    WHERE p.id_paiement = p_id_paiement AND p.statut = 'échoué';

    RETURN 'Statut mis à jour';
END

```

PARTIE II : Schéma MongoDB pour la Gestion d'Abonnements BeinSport

```
// Collection UTILISATEURS
{
    _id: ObjectId("..."),
    nom: "Dupont",
    prenom: "Jean",
    email: "jean.dupont@example.com",
    mot_de_passe: "$2a$10$...", // hashé
    date_inscription: ISODate("2023-05-15T00:00:00Z"),
    statut: "actif",
    abonnements: [
        {
            id_abonnement: ObjectId("..."),
            id_chaine: ObjectId("..."),
            date_debut: ISODate("2023-06-01T00:00:00Z"),
            date_fin: ISODate("2023-07-01T00:00:00Z"),
            statut: "actif"
        }
    ]
}

// Collection CHAINES
{
    _id: ObjectId("..."),
    nom: "BeinSports 1",
    description: "Chaîne sportive premium",
    prix_mensuel: 19.99,
    image_url: "http://.../bein1.jpg",
    stats: {
        abonnes_actifs: 1250,
        revenu_mensuel: 24987.50
    }
}

// Collection PAIEMENTS
{
    _id: ObjectId("..."),
    id_utilisateur: ObjectId("..."),
    id_abonnement: ObjectId("..."),
    montant: 19.99,
    date_paiement: ISODate("2023-06-01T10:30:00Z"),
    transaction_id: "PAY123456789",
    statut: "completé",
    methode: "carte_credit"
}
```

Répondez aux questions suivantes :

1. Quelle commande utiliser pour insérer un nouvel utilisateur dans la collection UTILISATEURS ?
2. Comment trouver tous les abonnements actifs d'un utilisateur spécifique ?
3. Quelle opération met à jour le prix mensuel de toutes les chaînes de plus de 15€ à 16.99€ ?

4. Comment calculer le revenu total mensuel de toutes les chaînes ?
5. Quelle requête trouve les utilisateurs avec au moins un abonnement expiré ?
6. Comment créer un index pour optimiser les recherches par email ?
7. Quelle opération supprime tous les paiements échoués de plus de 30 jours ?
8. Comment ajouter un nouvel abonnement à un utilisateur existant ?
9. Quelle requête trouve les chaînes avec un prix entre 10€ et 20€, triées par nombre d'abonnés décroissant ?
10. Quelle requête d'agrégation permet de calculer le chiffre d'affaires total par chaîne, trié par ordre décroissant ?
11. Comment trouver les utilisateurs dont **au moins un abonnement est expiré**, en affichant leur nom, email et la liste des abonnements expirés ?
12. Quelle requête calcule la **moyenne des montants des paiements groupés par méthode de paiement**, en excluant les paiements échoués ?

PARTIE III : Développement Backend

1- Développement Backend avec expressJS (Microservice)

Contexte :

Vous devez créer un **microservice** en **Express.js** qui permet de récupérer la liste des abonnements actifs pour une chaîne donnée (via son `id_chaine`).

- 1- Ecrire le code config.js qui permet d'établir la connexion à MongoDB
- 2- Ecrire le code app.js qui permet de créer un serveur http et déclarer une route vers l'endpoint « `abonnements/chaine/:id_chaine` » en utilisant le middleware **Router** d'express.
- 3- Quel code **Express.js** implémente correctement un endpoint **GET** `/abonnements/chaine/:id_chaine` qui renvoie les abonnements actifs d'une chaîne, en utilisant **MongoDB** comme base de données ?

2- Développement Backend avec Laravel (Microservice)

- a- Créer les migrations de définition de schéma de chaque table
- b- Créer les Modèles en implémentant les méthodes définissant des relations
- c- Implémenter les actions CRUD de gestion des abonnements
- d- Créer une API **GET** `/abonnements/chaine/:id_chaine` qui retourne en format json la liste des abonnées d'une chaîne