Contrôle terminal BUT2 / Semestre 3

R3.01 : Développement d'applications Web

Nom du responsable :	**Nicolas Le Sommer**
Date du contrôle :	**20 octobre 2021**
Durée du contrôle :	**1h30**
Nombre total de pages :	**3**
Impression :	**Recto-Verso**
Documents autorisés :	**Feuille A4 de notes personnelles**
Calculatrice autorisée :	**non**
Réponses :	**Les réponses devront être parfaitement lisibles et soignées. Les exercices devront être rendus sur des copies séparées.**

Dans ce sujet de contrôle, nous considérons une application Web de gestion de stock d'un parc automobile et de vente en ligne d'automobiles. Cette application comporte deux parties : une partie publique accessible aux clients et une partie privée accessible de manière sécurisée aux vendeurs qui gèrent le stock du parc automobile. Cette application repose sur une base de données dont le script de création de table est donné ci-dessous. Les vendeurs peuvent ajouter des véhicules dans le parc automobile, et en retirer. Les véhicules sont classés par catégorie (berline, break, cabriolet, monospace, etc.), par marque (Peugeot, Citroën, Renault, etc.) et par type de carburant. Les clients doivent peuvent consulter le catalogue selon ces critères, ainsi qu'en fonction du prix de vente et du kilométrage des véhicules. Le système de gestion de base de données utilisé est SQLite3.

```
CREATE TABLE IF NOT EXISTS Categorie(
id  INTEGER   CONSTRAINT pk_categorie PRIMARY KEY AUTOINCREMENT,
nom TEXT(100) CONSTRAINT nn_categorie_nom NOT NULL
              CONSTRAINT uq_categorie_nom UNIQUE
);

CREATE TABLE IF NOT EXISTS Carburant(
id  INTEGER  CONSTRAINT pk_carburant PRIMARY KEY AUTOINCREMENT,
nom TEXT(20) CONSTRAINT nn_carburant_nom NOT NULL
             CONSTRAINT uq_carburant_nom UNIQUE
);

CREATE TABLE IF NOT EXISTS Marque(
id  INTEGER   CONSTRAINT pk_marque PRIMARY KEY AUTOINCREMENT,
nom TEXT(100) CONSTRAINT nn_marque_nom NOT NULL
              CONSTRAINT uq_marque_nom UNIQUE
);

CREATE TABLE IF NOT EXISTS Vehicule(
id     INTEGER  CONSTRAINT pk_vehicule PRIMARY KEY AUTOINCREMENT,
modele TEXT(40) CONSTRAINT nn_model NOT NULL,
annee  DATE     CONSTRAINT nn_amc NOT NULL,
immatriculation TEXT (20) CONSTRAINT nn_immat NOT NULL
                          CONSTRAINT uq_immat UNIQUE,
prix   INTEGER  CONSTRAINT nn_prix NOT NULL
                CONSTRAINT ck_prix CHECK(prix >=0),
km     INTEGER  CONSTRAINT nn_km NOT NULL
                CONSTRAINT ck_km CHECK(km >=0),
marque INTEGER  CONSTRAINT nn_marque NOT NULL
                CONSTRAINT fk_marque REFERENCES Marque(id),
carburant INTEGER CONSTRAINT nn_carburant NOT NULL
                  CONSTRAINT fk_carburant REFERENCES Carburant(id),
categorie INTEGER CONSTRAINT nn_categorie NOT NULL
                  CONSTRAINT fk_categorie REFERENCES Categorie(id)
);
```

Exercice 1 : PHP (8 points)

1. En utilisant l'API PDO de PHP et le patron d'accès aux données « *Data Access Object* », écrivez des classes modélisant les données contenues dans la base de données de l'application, ainsi que les classes d'accès aux données associées à celles-ci. Vous n'implanterez que la méthode *findAll()* pour la classe *CarburantDAO*. Cette méthode devra retourner toutes les données contenues dans la table Carburant. Dans la classe *VehiculeDAO*, vous implanterez les méthodes *findAll()* et *insert()*. Dans le reste de l'exercice, vous supposerez l'existence des classes *Categorie*, *CategorieDAO*, Marque et *MarqueDAO* qui sont très semblables à celles des classes Carburant et *CarburantDAO*. Vous supposerez que les classes *CategorieDAO* et *MarqueDAO* offrent une méthode *findAll()* à l'instar de la classe *CarburantDAO*. Vous supposerez également l'existence d'une classe *SQLiteConnection* qui permet d'établir une connexion à la base de données. Cette classe met en œuvre le patron de conception « Singleton » et définit 3 méthodes, une méthode statique *getInstance()*, une méthode *getConnection()* et une méthode *close()*. Ces méthodes permettent respectivement d'obtenir une

instance de la classe *SQLiteConnection*, d'obtenir une référence vers l'objet PDO créé par la classe *SQLiteConnection* et de clore la connexion avec la base de données.

2. Écrivez un script PHP *ajout_vehicule_form.php* permettant de construire et de retourner aux clients Web un formulaire HTML5 permettant aux vendeurs d'ajouter des véhicules dans le stock. Cette vue devra permettre aux vendeurs de sélectionner le carburant, la catégorie et la marque du véhicule dans des listes déroulantes. Les vendeurs devront également renseigner la date de mise en circulation, la plaque d'immatriculation, le kilométrage et le prix de vente du véhicule. Ce formulaire sera transmis pour traitement à l'URL http://localhost:8080/ajout_vehicule.php.

Exercice 2 : Javascript / Express (12 points)

Dans cet exercice, nous considérons une nouvelle fois l'application de dépôt de travaux pratiques présentée en début de sujet de contrôle. Cette application sera développée dans cet exercice avec le langage Javascript, l'environnement Node.js et l'intergiciel Express.js.

1. En utilisant le module node.js *sqlite3* vu en travaux pratiques, écrivez une classe Javascript *VehiculeDAO* définissant une méthode *insert()* qui permet d'insérer un véhicule dans la base de données. Cette classe doit également définir une méthode *findAll()* permettant d'obtenir la liste des véhicules contenus dans la base de données. La méthode *findAll()* devra être mise en œuvre en utilisant une promesse. La méthode *insert()* devra quant à elle être mise en œuvre en utilisant le mécanisme de *callback*.
2. Écrivez un fichier Javascript */routes/vehicule.js* contenant une instance de la classe *Router* d'Express qui est capable de traiter les requêtes HTTP de type POST et d'insérer dans la base de données un véhicule à l'aide de la méthode *insert()* de la classe *VehiculeDAO*.
3. Donnez les lignes à ajouter dans le fichier *app.js* de l'application Express pour que le code contenu dans le fichier */routes/vehicule.js* puisse être exécuté lorsqu'une requête HTTP POST est adressée à l'application Web à l'URL *http://localhost:3000/vehicules*
4. Ajoutez dans le fichier Javascript */routes/vehicules.js* une méthode permettant de traiter les requêtes de type HTTP GET. Cette méthode devra utiliser la méthode *findAll()* de la classe *VehiculeDAO*. Les données obtenues par cette méthode devront être passées à la vue */views/vehicules_list.jade*. Vous n'écrirez pas la vue *vehicules_list.jade*. Celle-ci est présentée ci-dessous.

```
doctype html
    html
        head
            title="Catalogue véhicules"
        body
            table(border='1')
                thead
                    tr
                        th Modèle
                        th Marque
                        th Date de mise en circulation
                        th Kilométrage
                        th Carburant
                        th Prix
                tbody
                    each val in v_lst
                     tr
                        td= val.modele
                        td= val.marque
                        td= val.annee
                        td= val.km
                        td= val.carburant
                        td= val.prix
```