Contrôle terminal info2 / Semestre 3

M3104 : Développement d'applications Web côté serveur

Nom du responsable :	**Nicolas Le Sommer**
Date du contrôle :	**18 octobre 2021**
Durée du contrôle :	**1h30**
Nombre total de pages :	**4**
Impression :	**Recto-Verso**
Documents autorisés :	**Feuille A4 de notes personnelles**
Calculatrice autorisée :	**non**
Réponses :	

Dans ce sujet de contrôle, nous considérons une application Web permettant à des étudiants de déposer leurs travaux pratiques sous la forme d'une archive (au format zip), et à leurs enseignants de consulter la liste des travaux déposés et de télécharger ceux-ci. Les comptes utilisateurs (enseignants et étudiants) sont créés par un utilisateur administrateur. Cet administrateur précise notamment l'identifiant et le mot de passe d'un étudiant ou d'un enseignant lors de la création d'un compte (i.e. lors de l'ajout en base de données). Il indique également quel est le profil utilisateur (i.e. enseignant ou étudiant). Cet administrateur définit également la liste des matières pour lesquelles les étudiants déposeront des archives. Lors d'un dépôt, les étudiants préciseront la matière dans laquelle s'inscrit leur travail, en choisissant la matière dans une liste déroulante. Les enseignants pourront voir l'ensemble des travaux déposés par les étudiants. Ces travaux seront classés par matière. La base de données utilisée par cette application Web est une base SQLite contenue dans le fichier « tpapp.db ». Le script de création de cette base de données est le suivant :

```
CREATE TABLE IF NOT EXISTS User(
    id INTEGER CONSTRAINT pkUser PRIMARY KEY AUTOINCREMENT,
    lastname TEXT(100) CONSTRAINT nnUserLastname NOT NULL,
    firstname TEXT(100) CONSTRAINT nnUserFirstname NOT NULL,
    password TEXT(100) CONSTRAINT nnUserPassword NOT NULL,
    profile TEXT(8) CONSTRAINT nnUserProfile NOT NULL
                    CONSTRAINT CHECK (profile = 'ETUDIANT' or profile = 'ENSEIGNANT'),
    email TEXT(100) CONSTRAINT nnUserEmail NOT NULL
                    CONSTRAINT ckUserEmail CHECK(email LIKE '%_@__%.__%')
                    CONSTRAINT uqUserEmail UNIQUE
);

CREATE TABLE IF NOT EXISTS Course(
    id TEXT(8) CONSTRAINT pkCourse PRIMARY KEY,
    description TEXT(100) CONSTRAINT nnCourseDesc NOT NULL
);

CREATE TABLE IF NOT EXISTS Files(
    id INTEGER CONSTRAINT pkUpload PRIMARY KEY AUTOINCREMENT,
    uploadDate DATE CONSTRAINT nnUploadDate NOT NULL,
    fileURL TEXT(200) CONSTRAINT nnUploadFile NOT NULL,
    uid INTEGER CONSTRAINT nnUploadUID NOT NULL
                CONSTRAINT fkUploadUser REFERENCES User(id),
    uCourse TEXT(8) CONSTRAINT nnUploadUc NOT NULL
                CONSTRAINT fkUploadCourse REFERENCES Course(id)
);
```

Exercice 1 : HTML5 (3 points)

En utilisant le langage HTML5, écrivez une page HTML *user_add_form.html* contenant un formulaire permettant à l'administrateur de l'application Web décrite précédemment de créer un compte utilisateur. Vous supposerez que ce formulaire sera traité par un contrôleur accessible à l'URL http://localhost:3000/users. Vous veillerez à utiliser les bonnes balises et les bons attributs HTML pour fiabiliser la saisie des données.

Exercice 2 : PHP (9 points)

1. En utilisant l'API PDO de PHP et le patron d'accès aux données « *Data Access Object* », écrivez une classe *UserDAO* contenant une fonction *insert()*, fonction qui permet d'insérer un utilisateur dans la base de données SQLite. Vous écrirez toutes les classes supplémentaires nécessaires.
2. Écrivez une classe *UserController* qui étend la classe abstraite *Controller* ci-dessous. La fonction *post()* de cette classe *UserController* devra permettre d'ajouter un utilisateur dans la base de données en utilisant la classe *UserDAO* définie précédemment et les données saisies via le formulaire élaboré dans l'exercice 1. La fonction *get()* devra quant à elle permettre de retourner aux clients Web le contenu du fichier *user_add_form.html*. Dans cet exercice, les fonctions *put()* et *delete()* n'effectueront aucun traitement.
3. Écrivez un fichier *index.php* qui sera le point d'entrée unique de l'application Web de dépôt de travaux pratiques, et qui jouera le rôle de contrôle de façade de cette application. Ce contrôleur de façade devra invoquer le contrôleur approprié en fonction du chemin qui sera indiqué dans l'URL de la requête HTTP émise par le client Web. Côté serveur, vous pouvez obtenir ce chemin via la propriété *PATH_INFO* de la variable globale *$_SERVER*.

```
<?php
abstract class Controller{
  private $route;
  public static $controllers = array();

  function __construct($r){
    $this->route = $r;
    self::$controllers[$this->route]= $this;
  }
  public function getRoute(){
    return $this->route;
  }
  public function process($request){
    switch($_SERVER['REQUEST_METHOD']){
      case 'GET':
      this.get($request);
      break;
      case 'POST':
      this.post($request);
      break;
```

```
      case 'PUT':
      this.put($request);
      break;
      case 'DELETE':
      this.delete($request);
      break;
    }
  }

  public abstract function get($request);
  public abstract function post($request);
  public abstract function put($request);
  public abstract function delete($request);
}
?>
```

Exercice 3 : Javascript / Express (8 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 mettra en œuvre une API REST, et sera développée dans cet exercice avec le langage Javascript, l'environnement Node.js et l'intergiciel Express.js.

1. En utilisant le module *npm sqlite3* vu en travaux pratiques, écrivez une classe Javascript *FilesDAO* définissant une méthode *findAll()* qui permet d'obtenir les données contenues dans la table *Files* de la base de données SQLite.
2. Écrivez un fichier Javascript /*routes/files.js* contenant une instance de la classe *Router* d'Express qui est capable de répondre à une méthode HTTP de type GET et de retourner un tableau JSON contenant des objets JSON correspondant aux tuples de la table *Files* de la base de données SQLite. L'URL pour accéder à ces données sera : *http://localhost:3000/files*
3. Donnez les lignes à ajouter dans le fichier *app.js* de l'application Express pour que le code contenu dans le fichier /*routes/files.js* puisse être exécuté lorsqu'une requête HTTP GET est adressée à l'application Web à l'URL *http://localhost:3000/files*.