مكتب التكوين المهني وإنعاش الشغل

Office de la Formation Professionnelle et de la Promotion du Travail

Direction de la Recherche et de l'Ingénierie de la Formation
Division Conception des Examens

# Examen National de Fin de Formation
## Session de Juin 2024
# Examen de Fin de Formation (Epreuve de synthèse)

## Éléments de correction

| | | | |
|---|---|---|---|
| **Secteur :** | **Digital et Intelligence Artificielle** | **Niveau :** | **Technicien Spécialisé** |
| **Filière :** | **Développement Digital option Web Full Stack** | | |

| | | | | | |
|---|---|---|---|---|---|
| **Variante** | **V2** | **Durée :** | **4H00** | **Barème** | **100** |

**Consignes et Précisions aux correcteurs :**

Veuillez respecter impérativement les consignes suivantes :

- **Le corrigé est élaboré à titre indicatif,**
- **Eviter de sanctionner doublement le stagiaire sur les questions liées,**
- **Pour toutes les questions de synthèse et de compréhension le correcteur s'attachera à évaluer la crédibilité et la pertinence de la réponse du stagiaire. Et à apprécier toute réponse cohérente du stagiaire,**
- **Le stagiaire n'est pas tenu de fournir des réponses aussi détaillées que celles mentionnées dans le corrigé,**
- **Pour les exercices de calcul :**
  **- Prendre en considération la méthode de calcul correcte (formule et relation de calcul correcte) même si le résultat final de calcul est faux**
  **- Le résultat final correct non justifié ne doit pas avoir la totalité de la note.**
- **En cas de suspicion d'erreur au niveau du corrigé, prière de contacter la Division de Conception des Examens.**

**Détail du Barème :**

| **Théorique** | **40pts** | | **Pratique** | **60pts** |
|---|---|---|---|---|
| **Dossier 1** | **8pts** | | **Dossier 1** | **12pts** |
| Q1 | 2pts | | Q1 | 4pts |
| Q2 | 2pts | | Q2 | 4pts |
| Q3.a | 2pts | | Q3 | 4pts |
| Q3.b | 2pts | | **Dossier 2** | **24pts** |
| **Dossier 2** | **6pts** | | Q1 | 4pts |
| **Dossier 3** | **15pts** | | Q2 | 4pts |
| Q1 | 2pts | | Q3 | 4pts |
| Q2 | 2pts | | Q4 | 4pts |
| Q3.a | 2pts | | Q5 | 4pts |
| Q3.b | 2pts | | Q6 | 4pts |
| Q4 | 2pts | | **Dossier 3** | **24pts** |
| Q5 | 2pts | | Q1 | 1.5pt |
| Q6 | 1.5pt | | Q2 | 2pts |
| Q7 | 1.5pt | | Q3 | 1.5pt |
| **Dossier 4** | **11pts** | | Q4 | 4pts |
| Q1 | 3pts | | Q5 | 6pts |
| Q2 | 2pts | | Q6 | 4pts |
| Q3 | 2pts | | Q7 | 4pts |
| Q4 | 2pts | | Q8 | 1pt |
| Q5 | 2pts | | Total général | 100 pts |

## Partie Théorique (40 pts)

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

1- Définir Azure Cloud ? (2 pts)

Azure Cloud est la plateforme de cloud computing de Microsoft, elle offre une solution complète de services pour le développement, le déploiement et la gestion d'applications et d'infrastructures sur le cloud.

Azure propose une variété de services, tel que le stockage, les bases de données, l'analyse, l'intelligence artificielle, l'IoT (Internet des objets), la sécurité ….

2- Qu'est-ce qu'un conteneur dans le cloud native? (2 pts)

Un conteneur est une unité de logiciel abstraite qui est une unité d'exécutable autonome qui a tout ce qui est nécessaire pour exécuter une application : le code, l'exécution, les outils de système et les bibliothèques de système

3- On suppose qu'on veut travailler avec Docker :

a- Donner la commande qui permet de créer et démarrer un conteneur sur la base d'une image… (2 pts)

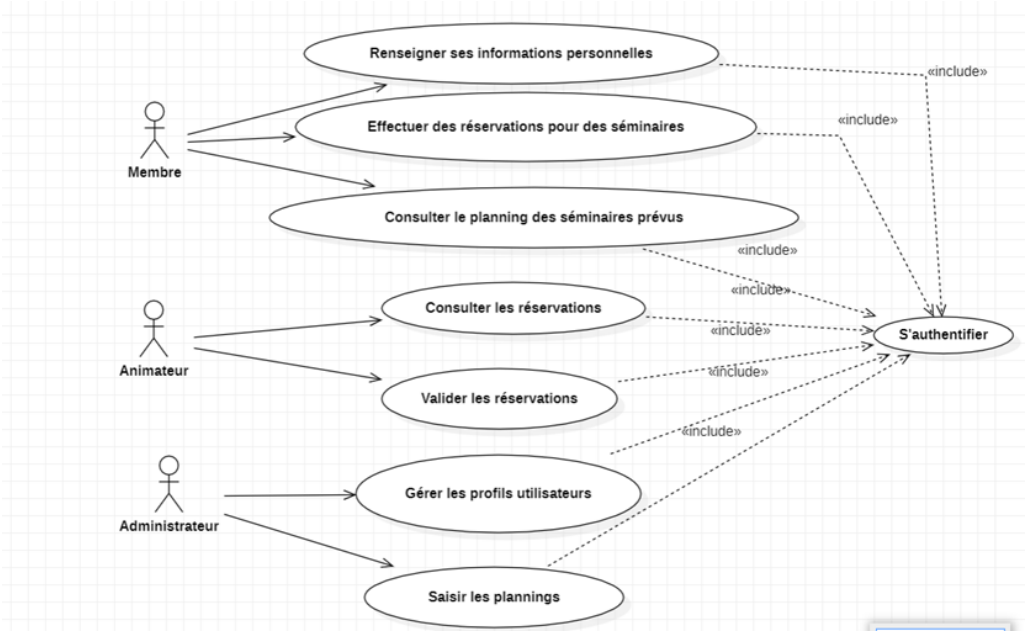
docker run --name <nom_conteneur> <nom_image>

b- Donner la commande permettant de lister les conteneurs actifs (2 pts)

docker ps

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

Diagramme de cas d'utilisation




### Dossier 3 : (Approche Agile) (15 pts)

1- Citez les 4 phases principales du cycle de vie d'un projet (2 pts)

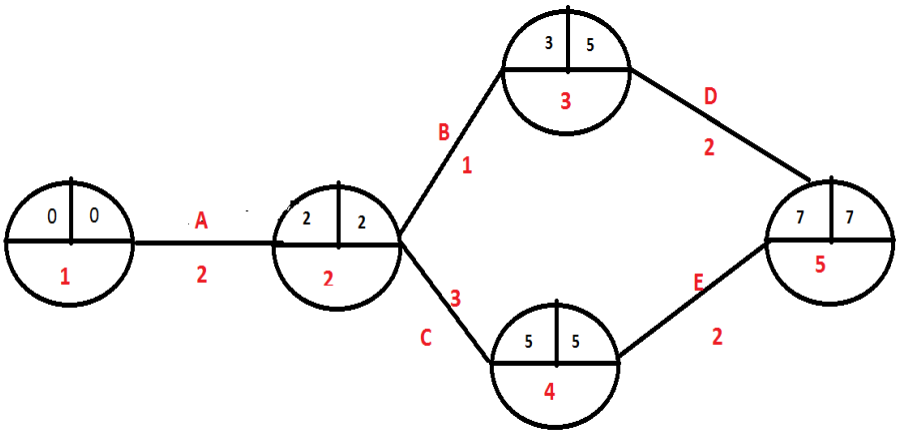
Lancement, planification, exécution et clôture.

2- Mettre en ordre les étapes de gestion du projet suivantes (2 pts)

La conception.
Le développement.
Les tests fonctionnels et techniques
La mise en production.

3-

a- Dresser le diagramme Pert des tâches ci-dessous en précisant les dates au plus tôt et au plus tard (2 pts)




b- Déterminer le chemin critique (2 pts)
Les tâches : A-C-E

4- C'est quoi une ***user story*** ? rédigez un exemple selon l'énoncé du dossier précédent (Préparation d'un projet web) (2 pts)

Une user story est la carte d'identité des fonctionnalités à développer.
Exemple : En tant que qu'Animateur je veux consulter les réservations des séminaires afin de préparer les activités

5- A quoi sert le logiciel ***Jira*** ? (2 pts)

Jira Software fournit des outils de planification et de suivi pour permettre aux équipes de gérer les dépendances, les exigences des fonctionnalités et les parties prenantes dès le démarrage du projet.

6- Quels sont les avantages de l'utilisation de ***SonarQube*** dans le développement logiciel ? (1,5 pt)

Il permet l'analyse de nombreux langages de développement sur plusieurs projets. Il propose de base plusieurs jeux de règles de qualité à appliquer et permet d'en ajouter d'autre.

7- Quelle est la commande ***Git*** permettant de créer une nouvelle branche nommée B1 et l'utiliser ? (1,5 pt)

git checkout -b B1

## Dossier 4 :(Gestion de données NOSQL) (11 pts)

1- Créez une base de données " DBS" et une collection "seminaires" contenant les informations suivantes : **(3 pts)**

```
//Création de la base de données
use DBS
//Création de la collection
db.createCollection("seminaires")
//Création des documents
db.seminaires.insert({ "_id" : "s1",
"theme": "Animation 3D",
"Description" : ""Introduction à l'animation 3D",
"cout_journalier" : 500,
 "Durée" :13,

"Activités" :[
{"_id" : "1",
 "nomActivité" : "Personnages en 3D",
" descriptionActivité " : " Atelier 3D"
}, {"_id" : "1",
 "nomActivité" : "Personnages en 3D",
```

```
 " descriptionActivité " : " Atelier 3D"
 }]
})
```

2- Afficher les thèmes et les descriptions des séminaires **(2 pts)**

```
db.seminaires.find({},{"thème":1,"description":1,"_id":0})
```

3- Afficher le nombre des activités du séminaire ayant le thème "Animation 3D" **(2 pts)**

```
4- db. seminaires.aggregate([
5- {$match:{“theme”:” Animation 3D”}},
6- {$group:{_id:null,totaleActivites:{$sum:1}}
7- ]
```

8- Affecter la valeur 700 au coût journalier du séminaire ayant l’id s2 **(2 pts)**

```
db.seminaires.update({"_id" : "s2"},{$set :{ "cout_journalier" : 700}})
```

9- Supprimer tous les séminaires qui ont une durée supérieure à 20 **(2pts)**

```
db.seminaires.deleteMany({"Durée" :{$gt : 20}})
```

**Partie Pratique : (60pts)**

**Dossier 1 : Gestion de données (MYSQL) (12pts)**

1. Créer une procédure qui permet d’insérer un nouveau séminaire en vérifiant l’existence de l’animateur qui assure le séminaire. **(4pts)**

```
DELIMITER //
CREATE PROCEDURE InsererNouveauSeminaire(
   IN p_theme VARCHAR(255),
   IN p_date_debut DATE,
   IN p_date_fin DATE,
   IN p_description TEXT,
   IN p_cout_journalier DECIMAL(10,2),
   IN p_animateur_id INT
)
BEGIN
   -- Vérifier l'existence de l'animateur
   IF EXISTS (SELECT 1 FROM Animateurs WHERE id = p_animateur_id) THEN
      -- L'animateur existe, insérer le nouveau séminaire
      INSERT INTO Séminaires(thème, date_debut, date_fin, description,
                                    cout_journalier,   animateur_id)
      VALUES (p_theme, p_date_debut, p_date_fin, p_description, p_cout_journalier,
                                  p_animateur_id);
      SELECT 'Nouveau séminaire insérée avec succès.' AS result;
   ELSE
      -- L'animateur n'existe pas, renvoyer un message d'erreur
      SELECT 'Erreur : Animateur inexistant. Veuillez fournir un ID d"animateur valide.' AS result;
   END IF;
END //
DELIMITER ;
```

2. Créer une fonction qui retourne le coût total des toutes les séminaires dispensées par un animateur dont le nom et prénom sont spécifiés en paramètres. **(4pts)**

```
DELIMITER //
CREATE FUNCTION CoutTotalSéminaires(
   p_nomAnimateur VARCHAR(255),
   p_prenomAnimateur VARCHAR(255)
)
RETURNS DECIMAL(10,2)
NOT deterministic
READS SQL DATA
BEGIN
   DECLARE total_cout DECIMAL(10,2) default 0.0 ;
   -- Calculer le coût total des séminaires de l'animateur
   SELECT SUM(S.cout_journalier * DATEDIFF(S.date_fin, S.date_debut))
       INTO total_cout     FROM Séminaires S
   JOIN Animateurs A ON S.animateur_id = A.id
   WHERE A.nomAnim= p_nomAnimateur AND A.prénomAnim = p_prenomAnimateur;
   RETURN total_cout;
END //
DELIMITER ;
```

3. Créer un trigger qui annule automatiquement les réservations effectués au cours du mois de juillet de l'année actuelle. **(4pts)**

```
DELIMITER //
CREATE TRIGGER AnnulerReservationJuillet
BEFORE INSERT ON Reservations
FOR EACH ROW
BEGIN
   -- Vérifier si dateReservation est en juillet de l'année actuelle
   IF MONTH(NEW.dateReservation) = MONTH(CURDATE()) AND
                    YEAR(NEW.dateReservation) = YEAR(CURDATE()) THEN

        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Annulation : Les réservations ne sont pas autorisées en
        juillet  de l\'année actuelle.';
   END IF;
END //
DELIMITER ;
```

## Dossier 2 : Développement front-end (24pts)

1. Créer le composant **Séminaires** qui reçoit la liste **séminaires du** tableau **animateursData** du fichier data.js d'un animateur donné en tant que **props** et les afficher dans une table HTML, en incluant le coût total pour chaque séminaire dans chaque ligne, et en bas de la table, en affichant le coût total de l'ensemble des séminaires sous la forme suivante : **[voir figure1] (4pts)**

```
// Séminaires.js
import React from 'react';
const Séminaires = ({ séminaires }) => {
 // Calcul du total des coûts
 const totalCout = séminaires.reduce((total, séminaire) =>
                   total + séminaire.cout_journalier * séminaire.durée, 0);
  return (
      <table border="1px">
        <thead>
          <tr>
            <th>Thème</th>
            <th>Date de début</th>
            <th>Date de fin</th>
```

```
            <th>Description</th>
            <th>Coût journalier</th>
            <th>Durée (jours)</th>
            <th>Coût Total Séminaire</th>
          </tr>
        </thead>
        <tbody>
          { séminaires  && séminaires.map((séminaire, index) => {

            return (
              <tr key={index}>
                <td>{séminaire.thème}</td>
                <td>{séminaire.date_debut}</td>
                <td>{séminaire.date_fin}</td>
                <td>{séminaire.description}</td>
                <td>{séminaire.cout_journalier} DH</td>
                <td>{séminaire.durée}</td>
                <td>{séminaire.cout_journalier * séminaire.durée} DH</td>
              </tr>
            );
          })}
        </tbody>
       <tfoot>
        <td colSpan={7}> Total des coûts des séminaires assurée est : {totalCout}
DH</td>
       </tfoot>
      </table>
      );
};
export default Séminaires;
```

2. Créer le composant **Animateur** qui reçoit un objet animateur du tableau **animateursData** en tant que props et qui l'affiche sous forme d'une balise (<li>) (**nom complet de l'animateur et sa liste des séminaires**) sous la forme suivante : **[voir figure 2] (4pts)**

```
// Animateur.js
import React from 'react';
import Séminaires from './Séminaires';
const Animateur = ({ animateur }) => {
  return (
    <li>
      <h2>{animateur.nom_complet}</h2>
      <Séminaires séminaires={animateur.séminaires} />
    </li>
  );
};
export default Animateur;
```

3. Créer le composant **Animateurs1** qui importe la constante « `animateursData`» du fichier data.js et affiche la liste des animateurs sous forme d'une liste à puce (<ul>) en utilisant les composants précédents.**[voir figure 3] (4pts)**

```
import React from 'react';
import Animateur from './Animateur';
```

```
import animateursData from './data';
const Animateurs1 = () => {
  return (
      <ul>
        {animateursData.map((animateur, index) => (
          <Animateur key={index} animateur={animateur} />
        ))}
      </ul>
  );
};
export default Animateurs1;
```

4. Créer un composant nommé **Formulaire** permettant de saisir le thème du séminaire, sa date de début, sa date de fin, son coût et un animateur. Le clic sur le bouton confirmer affiche les informations saisies de la manière suivante : **[voir figure 4] (4pts)**

```
// Formulaire.js
import React, { useState } from 'react';
const Formulaire = () => {
  const [theme, setTheme] = useState('');
  const [dateDebut, setDateDebut] = useState('');
  const [dateFin, setDateFin] = useState('');
  const [cout_journalier, setCout_journalier] = useState(0);
const [animateur, setAnimateur] = useState('');
  const [messageRecap, setMessageRecap] = useState('');
  const handleSubmit = (e) => {
    e.preventDefault();

    // Calcul de la durée du séminaire
    const dateDebutObj = new Date(dateDebut);
    const dateFinObj = new Date(dateFin);
    const differenceEnMillisecondes = dateFinObj - dateDebutObj;
    const differenceEnJours = differenceEnMillisecondes / (1000 * 3600 * 24)+1;
    // Calcul du coût total
    const coutTotal = parseFloat(cout_journalier) * differenceEnJours;

    // Formatage du message de récapitulation en liste à puce
 setMessageRecap(
     `l'animateur:${expert} assurera le thème: ${theme},avec un coût journalier:
${cout_journalier} DH,pour une
     durée de : ${differenceEnJours} jours,soit un coût total de:${coutTotal}DH `
    );
  };
  return (
    <div>
      <h3>Formulaire du séminaire</h3>
      <form onSubmit={handleSubmit}>
        <table>
          <tbody>
            <tr>
              <td>Thème :</td>
              <td><input type="text"  size={40} value={theme}  onChange={(e) =>
```

```
                    setTheme(e.target.value)} /></td>
            </tr>
            <tr>
              <td>Date de début :</td>
              <td><input type="date" value={dateDebut} onChange={(e) =>
                    setDateDebut(e.target.value)} /></td>
            </tr>
            <tr>
              <td>Date de fin :</td>
              <td><input type="date" value={dateFin} onChange={(e) =>
                    setDateFin(e.target.value)} /></td>
            </tr>
            <tr>
              <td>Coût :</td>
              <td><input type="number" value={cout_journalier} onChange={(e) =>
                     setCout_journalier(e.target.value)} /></td>
            </tr>
            <tr>
              <td>Animateur :</td>
              <td>
               <input type="text" value={animateur} onChange={(e) =>
                         setAnimateur(e.target.value)} />
              </td>
            </tr>
          </tbody>
        </table>
        <button type="submit">Confirmer</button>
      </form>
      {messageRecap}
    </div>
  );
};
export default Formulaire;
```

5. Créer le composant **Animateurs2** qui permet de récupérer la liste des animateurs à partir de l'URL, l'affichage se présente sous forme d'une liste à puces (<ul>). **(4pts)**

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import Animateur from './Animateur';
const Animateurs2 = () => {
  const [animateursData, setAnimateursData] = useState([]);
  useEffect(() => {
    axios.get('http://localhost:8000/animateurs')
      .then(response => {
            setAnimateursData(response.data);
      })
      .catch(error => {
        console.error('Erreur lors de la récupération des animateurs depuis l\'API :',
error);
      });
  }, []);
```

```
  return (
      <ul>
        {animateursData.map((animateur, index) => (
          <Animateur key={index} animateur={animateur} />
        ))}
      </ul>
  );
};
export default Animateurs2;
```

6. Créer le composant App.js incluant les chemins et les routes vers les composants : Formulaire.js (/formulaire), Animateurs1.js (/Animateurs1) et Animateurs2.js (/Animateurs2) sous la forme suivante : **(4pts)**

```
// App.js
import React from 'react';
import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-dom';
import Formulaire from './Formulaire';
import Animateurs1 from './Animateurs1';
import Animateurs2 from './Animateurs2';
const App = () => {
  return (
    <Router>
      <div>
        <h1>Gestion des séminaires</h1>
        <nav>
            <ul>
              <li><Link to="/formulaire">Formulaire du séminaire</Link></li>
              <li><Link to="/animateurs1">liste des animateurs 1</Link></li>
              <li><Link to="/animateurs2">liste des animateurs 2</Link></li>
            </ul>
        </nav>
        <Routes>
          <Route path="/formulaire" element={<Formulaire />} />
          <Route path="/animateurs1" element={<Animateurs1 />} />
          <Route path="/animateurs2" element={<Animateurs2 />} />
        </Routes>
      </div>
    </Router>
  );
};
export default App;
```

## Dossier 3 : Développement back-end (Laravel) (24pts)

1. Donner les commandes de créations des fichiers de migrations : **(1.5pt)**
   « create_animateurs_table », « create_séminaires_table », « create_activités_table »
   - **php artisan make:migration create_animateurs_table**
   - **php artisan make:migration create_séminaires_table**
   - **php artisan make:migration create_activités_table**
2. Donner le code de la méthode up () du fichier de migration de la table **séminaires (2pts)**

```php
public function up()
    {
        Schema::create('séminaires', function (Blueprint $table) {
            $table->id();
            $table->string('thème');
            $table->date('date_debut');
            $table->date('date_fin');
            $table->text('description');
            $table->decimal('cout_journalier', 8, 2);
            $table->foreignId('animateur_id')->constrained('animateurs');
            $table->timestamps();
        });
    }
```

3. Donner les commandes de création des modèles : Animateur, Séminaire, Activité. **(1.5pts)**
    - php artisan make:model Animateur
    - php artisan make:model Séminaire
    - php artisan make:model Activité
4. Donner le code des fichiers du modèle Formation : **(4pts)**

```php
class Séminaire extends Model
{
    use HasFactory;
    protected $fillable = [
        'thème',
        'date_debut',
        'date_fin',
        'description',
        'cout_journalier',
        'animateur_id',
    ];
    public function animateur()
    {
        return $this->belongsTo(Animateur::class);
    }
    public function activités()
    {
        return $this->hasMany(Activité::class);
    }
}
```

5. Créer le contrôleur de resource **SéminaireController** (2pts + 2pts + 2pts)

```php
class SéminaireController extends Controller
{
    public function index()
    {
        $séminaires = Séminaire::all();
        return view('séminaires.index', compact('séminaires'));
    }
    public function show(Séminaire $séminaire)
    {
        return view('séminaires.show', compact('séminaire'));
    }
    public function destroy(Séminaire $séminaire)
    {
```

```
        $séminaire->delete();

        // Redirigez vers la liste des formations avec un message de succès
        return redirect()->route('séminaires.index')->with('success',
                                'Sémiaire supprimé avec succès.');
    }
}
```

6. Créer la vue index.blade.php affichant la liste des séminaires et les actions comme suit : **(4pts)**

```
<h2>Liste des séminaires</h2>
@if(session('success'))
    <div class="alert alert-success">
        {{ session('success') }}
    </div>
@endif
<table border='1px' cellspacing='0' cellpadding='5'>
    <thead>
        <tr>
            <th>Thème</th>
            <th>Date début</th>
            <th>Date fin</th>
            <th>Description</th>
            <th>Coût journalier</th>
            <th>Séminaire_id</th>
            <th>Actions</th>
        </tr>
    </thead>
    <tbody>
        @foreach($séminaires as $séminaire)
            <tr>
                <td>{{ $séminaire ->thème }}</td>
                <td>{{ $séminaire ->date_debut }}</td>
                <td>{{ $séminaire ->date_fin }}</td>
                <td>{{ $séminaire ->description }}</td>
                <td>{{ $séminaire ->cout_journalier }}</td>
                <td>{{ $séminaire->animateur_id }} </td>
                <td>
                    <a href="{{ route('seminaires.show', $séminaire->id)
                        }}">Consulter</a>
                    <a href="{{ route('seminaires.edit', $séminaire->id)
                        }}">Modifier</a>
                    <form action="{{ route('seminaires.destroy', $seminaire-
                        >id) }}" method="POST">
                        @csrf
                        @method('DELETE')
                        <button type="submit" class="btn btn-danger"
                          onclick="return confirm('Êtes-vous sûr de vouloir
                         supprimer ce séminaire?')">Supprimer</button>
                    </form>
                </td>
```

```
                </tr>
            @endforeach
        </tbody>
    </table>
```

7. Créer la vue **show.blade.php** affichant le détail d'un séminaire **[utiliser les relations définies dans la question 4)] (4pts)**

```
<h2>Détails du séminaire: {{ $séminaire->id }}</h2>
<table border='1' cellspacing='0px' cellpadding='5px'>
            <tr>
                <th>Thème</th>
                <td>{{ $séminaire->thème }}</td>
            </tr>
            <tr>
                <th>Date début</th>
                <td>{{ $séminaire->date_debut }}</td>
            </tr>
            <tr>
                <th>Date fin</th>
                <td>{{ $séminaire->date_fin }}</td>
            </tr>
            <tr>
                <th>Description</th>
                <td>{{ $séminaire->description }}</td>
            </tr>
            <tr>
                <th>Coût journalier</th>
                <td>{{ $séminaire->cout_journalier }}</td>
            </tr>
            <tr>
                <th>Animateur</th>
                <td>{{ $séminaire->animateur->nomAnim }} {{ $séminaire->animateur-
                                                        >prénomAnim }}</td>
            </tr>
</table>
<h3>Liste des activités assurés : </h3>
<table border='1' cellspacing='0px' cellpadding='5px'>
    <thead>
        <tr>
            <th>Nom de l'activité</th>
            <th>Description de l'activité</th>
        </tr>
    </thead>
    <tbody>
        @foreach($séminaire->activités as $activite)
            <tr>
                <td>{{ $activite->nomActivité }}</td>
                <td>{{ $activite->descriptionActivité }}</td>
            </tr>
        @endforeach
    </tbody>
</table>
<a href="{{ route('seminaires.index') }}">retour</a>
```

8. Écrire le code de la route menant aux actions du contrôleur **SéminaireController (1pt)**

```
Route::resource('semainaires', SéminairesController::class);
```