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)

Eléments de correction

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

Variante	**V1**	**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)

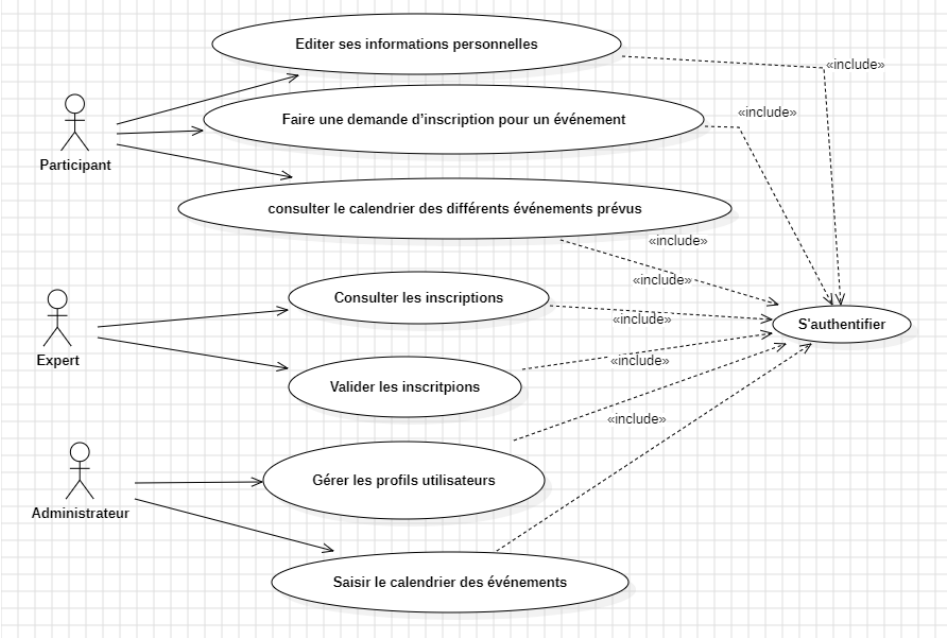
1- Que signifie Node.js ? (2 pts)
Node.js est un environnement d'exécution JavaScript côté serveur, construit sur le moteur JavaScript V8 de Chrome. Il permet d'exécuter du code JavaScript côté serveur

2- Qu'est-ce qu'un micro-service dans le cloud native? (2 pts)
Un microservice est un service unique conçu pour accueillir une fonctionnalité d'application et gérer des tâches discrètes. Chaque microservice communique avec d'autres services via des interfaces simples afin de répondre à des problématiques métier.

3- On suppose qu'on veut travailler avec Docker
   a- Donner la commande pour extraire (télécharger) **l'image Docker** depuis un registre (Docker hub par défaut) (2 pts)
   docker pull <image>
   b- Donner la commande pour supprimer une **image** depuis l'hôte local (2 pts)
   docker rmi <image>

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

Dresser le digramme de cas d'utilisation



Dossier 3 : (Approche Agile) (15 pts)

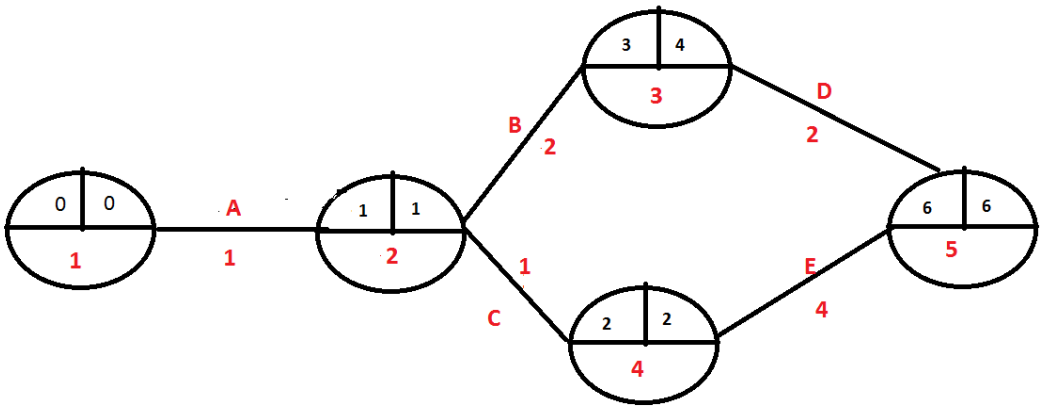
La société **AdvancedEventSolutions** vous a chargé de la création du site web de la gestion des formations.

1- Citez les contraintes d'un projet informatique **(2 pts)**
   - ✓ Contraintes de délai
   - ✓ Contraintes de coût
   - ✓ Contraintes de qualité
   - ✓ Contraintes dues au client

2- Il est demandé dans cette question de les mettre en ordre **(2 pts)**
   a. Rédaction du cahier de charges.
   b. L'Analyse.
   c. La conception
   d. La réalisation

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)**
Tâches : A-C-E

4- Quelles sont les grandes étapes de la méthode Scrum **(2 pts)**
- ✓ Le product backlog
- ✓ Sprint
- ✓ Sprint review
- ✓ Sprint Retrospective

5- Votre équipe souhaite travailler avec la chaine DEVOPS, citez quelques outils à utiliser **(2 pts)**
- ✓ Les outils de gestion de code source (Git, Github , Gitlab…)
- ✓ Les tests d'intégration continue / déploiement continu (jenkins)
- ✓ Les conteneurs (Docker…)
- ✓ Outils de gestion de projet (Jira , Trello…)

6- Quels sont les principaux rôles de Scrum ? **(1,5 pt)**
- ✓ Product Owner
- ✓ Scrum Master
- ✓ Equipe de développement

7- Quelle est la commande Git permettant de fusionner une branche spécifiée dans la branche actuelle **(1,5 pt)**
git merge [branche]

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

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

```
//Création de la base de données
use DBE
//Création de la collection
db.createCollection("evenements")
//Création des documents
db.evenements.insert({
 "_id" : "e1",
"theme" : "Développement Web",
"Description" : "Introduction au développement web moderne",
"cout_journalier" : 500,
 "Durée" :13,
"Ateliers" :[
 {"_id" : "1",
  "nomAtelier" : "Atelier HTML",
 " descriptionAtelier " : "Introduction à HTML"
 }]})
```

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

```
db.evenements.find({},{"theme":1,"description":1,"_id":0})
```

3- Afficher le nombre des ateliers de l'événement ayant le thème "Développement Web" **(2 pts)**

```
db.evenements.aggregate([
{$match:{"theme":"Developpement Web"}},
{$group:{_id:null,totaleAtelier:{$sum:1}}
]
```

4- Affecter la valeur 600 au coût journalier de l'événement ayant l'id e2 **(2 pts)**

```
db.evenements.update({"_id" : "e2"},{$set :{ "cout_journalier" : 600}})
```

5- Supprimer tous les événements qui ont une durée supérieure à 15 **(2pts)**

```
db.evenements.deleteMany({"Durée" :{$gt : 15}})
```

Partie Pratique : (60pts)

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

1. Créer une procédure qui permet d'insérer un nouvel événement en vérifiant l'existence de l'expert qui assure l'événement. **(4pts)**

```
DELIMITER //
CREATE PROCEDURE InsererNouvelEvenement(
   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_expert_id INT
)
BEGIN
   -- Vérifier l'existence de l'expert
   IF EXISTS (SELECT 1 FROM Experts WHERE id = p_expert_id) THEN
      -- L'expert existe, insérer le nouveau événement
      INSERT INTO Evenements(thème, date_debut, date_fin, description, cout_journalier,
                                    expert_id)
      VALUES (p_theme, p_date_debut, p_date_fin, p_description, p_cout_journalier,
                                 p_expert_id)
      SELECT 'Nouvel événement insérée avec succès.' AS result;
   ELSE
      -- L'expert n'existe pas, renvoyer un message d'erreur
      SELECT 'Erreur : Expert inexistant. Veuillez fournir un ID de l'expert valide.' AS result;
   END IF;
END //
DELIMITER ;
```

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

```
DELIMITER //
CREATE FUNCTION CoutTotalEvenements(
   p_nomExpert VARCHAR(255),
   p_prenomExpert 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 événement de l'expert
    SELECT SUM(FT.cout_journalier * DATEDIFF(FT.date_fin, FT.date_debut))
        INTO total_cout     FROM Evenements FT
    JOIN Expert F ON FT.expert_id = F.id
    WHERE F.nomExp = p_nomExpert AND F.prenomExp = p_ prenomExpert;
    RETURN total_cout;
END //
DELIMITER ;
```

3. Créer un trigger qui interdit l'inscription effectuée au cours du mois de juillet de l'année actuelle. **(4pts)**

```
DELIMITER //
CREATE TRIGGER AnnulerInscriptionsJuillet
BEFORE INSERT ON Inscriptions
FOR EACH ROW
BEGIN
    -- Vérifier si la date d'inscription est en juillet de l'année actuelle
    IF MONTH(NEW.dateInscription) = MONTH(CURDATE()) AND
                    YEAR(NEW.dateInscription) = YEAR(CURDATE()) THEN

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

Dossier 2 : Développement front-end (24pts)

1. Créer le composant **Evenements** qui reçoit le liste d'événements du fichier `expertsData` data.js d'un expert donné en tant que **props** et les affiche dans une table HTML, en incluant le coût total pour chaque événement dans chaque ligne, et en bas de la table, en affichant le coût total de l'ensemble des événements sous la forme suivante : **[voir figure1] (4pts)**

```jsx
// Evenements.js
import React from 'react';
const Evenements  = ({ événements  }) => {
 // Calcul du total des coûts
 const totalCout = événements.reduce((total, evenement) =>
                  total + evenement.cout_journalier * evenement.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 Evénement</th>
          </tr>
        </thead>
        <tbody>
          {événements.map((evenement, index) => {

            return (
              <tr key={index}>
                <td>{evenement.thème}</td>
                <td>{evenement.date_debut}</td>
```

```
                <td>{evenement.date_fin}</td>
                <td>{evenement.description}</td>
                <td>{evenement.cout_journalier} DH</td>
                <td>{evenement.durée}</td>
                <td>{evenement.cout_journalier * evenement.durée} DH</td>
              </tr>
            );
          })}
        </tbody>
       <tfoot>
        <td colSpan={7}> Total des couts des événements assurés est : {totalCout}
         DH</td>
       </tfoot>
      </table>
  );
};
export default Evenements;
```

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

```
import React from 'react';
import Evenements from './Evenements';
const Expert  = ({ expert  }) => {
  return (
    <li>
      <h2>{expert.nom_complet}</h2>
      <Evenements événements={expert.événements} />
    </li>
  );
};
export default Expert ;
```

3. Créer le composant **Experts1** qui importe la constante « expertsData» du fichier data.js et affiche la liste des experts sous forme d'une liste à puce (<ul>) en utilisant les composants précédents.**[voir figure 3] (4pts)**

```
import React from 'react';
import Expert from './Expert';
import expertsData from './data';
const Experts1 = () => {
  return (
      <ul>
        {expertsData.map((expert, index) => (
          <Expert key={index} expert={expert} />
        ))}
      </ul>
  );
};
export default Experts1;
```

4. Créer un composant nommé **Formulaire** permettant de saisir le thème de l'événement, sa date de début, sa date de fin, son coût et de choisir l'expert concerné assurant l'événement. Le clic sur le bouton confirmer affiche les informations saisies **(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 [expert, setExpert] = useState('');
  const [messageRecap, setMessageRecap] = useState('');
  const handleSubmit = (e) => {
    e.preventDefault();
    // Calcul de la durée de l'événement
    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'expert:${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 de l'événement</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>Expert :</td>
              <td>
                <input type="text" value={expert} onChange={(e) =>
                      setExpert(e.target.value)} />
              </td>
            </tr>
          </tbody>
        </table>
        <button type="submit">Confirmer</button>
      </form>
      {messageRecap}
    </div>
  );
};
```

```
export default Formulaire;
```

5. Créer le composant **Experts2** qui permet de récupérer la liste des experts 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 Expert from './Expert';
const Experts2 = () => {
  const [expertsData, setExpertsData] = useState([]);
  useEffect(() => {
    // Effectue une requête HTTP pour récupérer les données des experts depuis l'API
    axios.get('http://localhost:8000/experts2')
      .then(response => {
        // Met à jour la variable d'état avec les données de la réponse
        setExpertsData(response.data);
      })
      .catch(error => {
          console.error('Erreur lors de la récupération des experts depuis l\'API :',
error);
      });
  }, []);
  return (
      <ul>
        {expertsData.map((expert, index) => (
          <Expert key={index} expert={expert} />
        ))}
      </ul>
  );
};
export default Experts2;
```

6. Créer le composant App.js incluant les chemins et les routes vers les composants : Formulaire.js (/formulaire), Experts1.js (/Experts1) et Experts2.js (/Experts2): **(4pts)**

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-dom';
import Formulaire from './Formulaire';
import Experts1 from './Experts1';
import Experts2 from './Experts2';
const App = () => {
  return (
    <Router>
      <div>
        <h1>Gestion des événements</h1>
        <nav>
            <ul>
              <li><Link to="/formulaire">Formulaire de l'événement</Link></li>
              <li><Link to="/experts1">liste des experts 1</Link></li>
              <li><Link to="/experts2">liste des experts 2</Link></li>
            </ul>
        </nav>
        <Routes>
          <Route path="/formulaire" element={<Formulaire />} />
          <Route path="/experts1" element={<Experts1 />} />
          <Route path="/experts2" element={<Experts2 />} />
        </Routes>
```

```
      </div>
    </Router>
  );
};
export default App;
```

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

1. Donner les commandes de créations des fichiers de migrations : **(1.5pts)**
   « create_experts_table », « create_evenements_table », « create_ateliers_table »
   - **php artisan make:migration create_experts_table**
   - **php artisan make:migration create_evenements_table**
   - **php artisan make:migration create_ateliers_table**
2. Donner le code de la méthode up () du fichier de migration de la table **evenements (2pts)**

```
public function up()
    {
        Schema::create('evenements', 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('expert_id')->constrained('experts');
            $table->timestamps();
        });
    }
```

3. Donner les commandes de création des modèles : Expert, Evenement, Atelier. **(1.5pts)**
   - php artisan make:model Expert
   - php artisan make:model Evenement
   - php artisan make:model Atelier
4. Donner le code du fichier modèle Evenement : **(4pts)**

```
class Evenement extends Model
{
    use HasFactory;
    protected $fillable = [
        'thème',
        'date_debut',
        'date_fin',
        'description',
        'cout_journalier',
        'expert_id',
    ];
    public function expert()
    {
        return $this->belongsTo(Expert::class);
    }
    public function ateliers()
    {
        return $this->hasMany(Atelier::class);
    }
}
```

5. Créer le contrôleur de ressource **EvenementController** (2pts + 2pts + 2pts)

```
class EvenementController extends Controller
{
```

```php
    public function index()
    {
        $evenements = Evenement::all();
        return view('evenements.index', compact('evenements'));
    }
    public function show(Evenement $evenement)
    {
        return view('evenements.show', compact('evenement'));
    }
    public function destroy(Evenement $evenement)
    {
        // Supprimez l'évenement
        $evenement->delete();

        // Redirigez vers la liste des évenements avec un message de succès
        return redirect()->route('evenements.index')->with('success',
                                'Evénement supprimée avec succès.');
    }
}
```

6. Créer la vue index.blade.php affichant la liste des événements et les actions (Consulter,Modifier,Supprimer): **(4pts)**

```blade
<div >
    <h2>Liste des événements</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>Expert_id</th>
                <th>Actions</th>
            </tr>
        </thead>
        <tbody>
            @foreach($evenements as $evenement)
                <tr>
                    <td>{{ $evenement->theme }}</td>
                    <td>{{ $evenement->date_debut }}</td>
                    <td>{{ $evenement->date_fin }}</td>
                    <td>{{ $evenement->description }}</td>
                    <td>{{ $evenement->cout_journalier }}</td>
                    <td>{{ $evenement->expert_id }} </td>
                    <td>
                        <a href="{{ route('evenements.show',$evenement->id) }}">Consulter</a>
                        <a href="{{ route('evenements.edit', $evenement->id) }}">Modifier</a>
                        <form action="{{ route('evenements.destroy', $evenement->id) }}"
```

```
                            method="POST">
                            @csrf
                            @method('DELETE')
                            <button type="submit" class="btn btn-danger" onclick="return
confirm('Êtes-vous sûr de vouloir supprimer cet événement?')">Supprimer</button>
                        </form>
                    </td>
                </tr>
            @endforeach
        </tbody>
    </table>
</div>
```

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

```
<h2>Détails de l'événement : {{ $evenement->id }}</h2>
<table border='1' cellspacing='0px' cellpadding='5px'>
            <tr>
                <th>Thème</th>
                <td>{{ $evenement->theme }}</td>
            </tr>
            <tr>
                <th>Date début</th>
                <td>{{ $evenement->date_debut }}</td>
            </tr>
            <tr>
                <th>Date fin</th>
                <td>{{ $evenement->date_fin }}</td>
            </tr>
            <tr>
                <th>Description</th>
                <td>{{ $evenement->description }}</td>
            </tr>
            <tr>
                <th>Coût journalier</th>
                <td>{{ $evenement->cout_journalier }}</td>
            </tr>
            <tr>
                <th>Expert</th>
                <td>{{ $evenement->expert->nomExp }} {{ $evenement->expert->prenomExp }}</td>
            </tr>
</table>
<h3>Liste des ateliers assurées : </h3>
<table border='1' cellspacing='0px' cellpadding='5px'>
    <thead>
        <tr>
            <th>Nom de l'atelier</th>
            <th>Description de l'atelier</th>
        </tr>
    </thead>
    <tbody>
        @foreach($evenement->ateliers as $atelier)
            <tr>
                <td>{{ $atelier->nomAtelier }}</td>
                <td>{{ $atelier->descriptionAtelier }}</td>
            </tr>
        @endforeach
    </tbody>
</table>
<a href="{{ route('evenements.index') }}">retour</a>
```

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

```
Route::resource('evenements', EvenementController::class);
```