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 d'année
Session de Juin 2023
Examen de Fin de Formation (Epreuve 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.**

Partie Théorique (40 pts)

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

1- Définir un cloud public, citez deux exemples de fournisseurs **(2 pts)**

Avec un Cloud public, votre entreprise achète des services de calcul, de stockage et de réseau virtualisés sur l'Internet public d'un fournisseur de services Cloud Cela peut vous aider à avancer les délais de mise sur le marché, à évoluer rapidement et à profiter de l'agilité permettant d'essayer rapidement de nouvelles applications et services

Exemple de fournisseurs : Alibaba Cloud, AWS , google cloud …..

2- Citer les avantages du Cloud **(2 pts)**

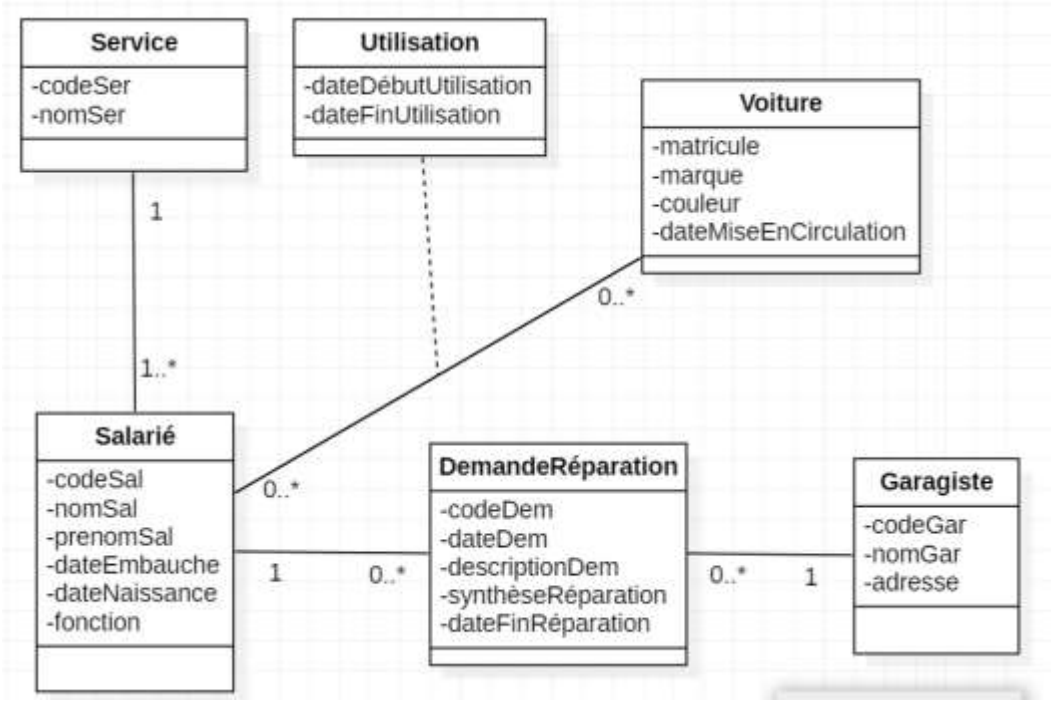
- ✓ Économies de coûts: Les services cloud permettent aux entreprises de réduire les coûts liés à l'achat et à la maintenance d'infrastructures physiques.
- ✓ Évolutivité: Les services cloud offrent une grande flexibilité pour s'adapter aux besoins changeants des entreprises.
- ✓ Accès à distance: Les services cloud peuvent être accessibles à partir de n'importe quel endroit
- ✓ Sécurité: Les fournisseurs de services cloud sont responsables de la sécurité de leurs services.

3- C'est quoi un Microservice ? quelle est la différence entre le microservice et une API ? **(4 pts)**

- ✓ Les microservices désignent à la fois une architecture et une approche de développement logiciel qui consiste à décomposer les applications en éléments les plus simples, indépendants les uns des autres.
- ✓ La principale différence entre un microservice et une API est que le microservice est une méthode de développement logiciel, alors que l'API est une interface qui permet à différents systèmes informatiques de communiquer entre eux. Les microservices peuvent utiliser des API pour communiquer entre eux, mais tous les services qui utilisent des API ne sont pas nécessairement des microservices.

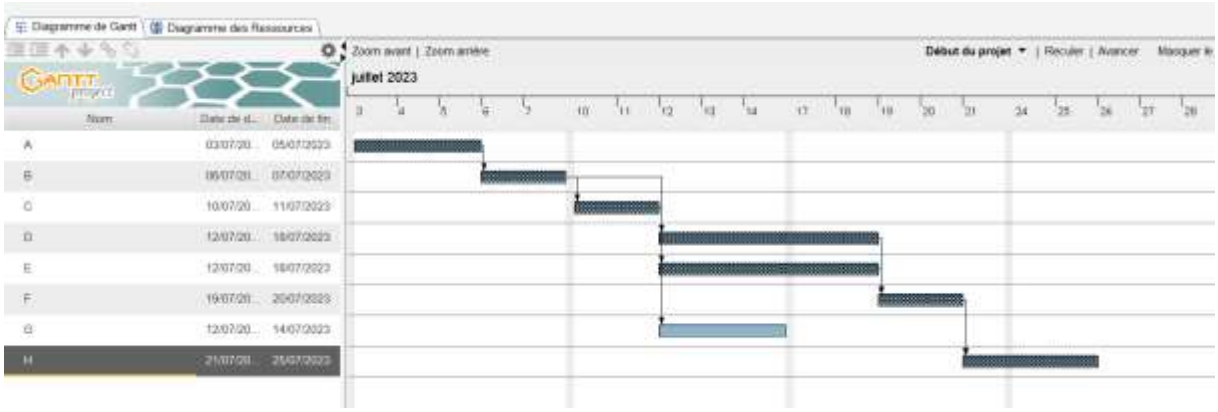
Dossier 2 : (Préparation d'un projet web) (6 pts)

NB : Le correcteur doit prendre en considération les différentes propositions des stagiaires



Dossier 3 : (Approche Agile) (15 pts)

1- Dresser le diagramme de GANTT en supposant que le projet démarrera le lundi 03/07/2023. **(3 pts)**



2- Déterminer le chemin critique et indiquer la durée minimale de réalisation du projet. **(2 pts)**

Chemin critique : A,B,C,D,E,F,H

Durée du projet : 17 jours

1. On souhaite de travailler avec la méthode Scrum pour réaliser le projet précédent :

a- Quels sont les principaux rôles de Scrum **(2 pts)**
Product Owner, Scrum Master et les membres de l'équipe de développement
b- Rédiger trois user stories à intégrer dans le backlog produit **(3 pts)**

2. On suppose maintenant que vous utiliser un outil de gestion des versions avec votre équipe de développement
   a- Ecrire la commande qui transfère votre code et vos modifications dans GitLab **(1 pt)**
   git push
   b- Ecrire la commande permettant d'initialiser le dépôt local avec Git **(1 pt)**
   git init
   c- Ecrire la commande pour enregistrer l'état actuel de votre code dans git **(1 pt)**
   git commit
3. On veut mesurer la qualité de notre code avec SonarQube, quels défauts permet-il d'identifier ? **(2 pts)**
   - ✓ Les bugs : anomalies évidentes du code. Ils impactent la fiabilité (reliability) de l'application.
   - ✓ Les vulnérabilités : faiblesses du code pouvant nuire au système. Elles impactent la sécurité de l'application.
   - ✓ Les « code smells » : anti-patrons (ou anti-patterns).

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

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

```
//Création de la base de données
use DBSalaries
//Création de la collection
db.createCollection("salaries")
//Création des documents
db.salaries.insert({ "_id" : "s1",
"nomsal" : "Alami",
"prenomsal" : "Sara",
"fonction" : "Technicien",
"service" :{ "codeser" : "1","nomser" : "informatique"}})
db.salaries.insert(
{ "_id" : "s2",
"nomsal" : "Hilali",
"prenomsal" : "Hamza",
"fonction" : "Gestionnaire",
"service" :{ "codeser" : "2","nomser" : "logistique"}})
```

On suppose que la collection salaries contient un ensemble de documents, écrire les codes des requêtes mongoDB permettant de :

2- Afficher les salariés triés par ordre croissant des noms (2 pts)

```
db.salaries.find().sort({"nomsal":1})
```

3- Afficher le nombre des salariés ayant la fonction "Technicien" (2 pts)

```
db.salaries.find({"fonction" : "Technicien"}).count()
```

4- Supprimer le salarié ayant l' _id "s3" (2 pts)

```
db.salaries.deleteOne({"_id","s3"})
```

5- Afficher le nombre de salariés par fonction (2pts)

```
db.salaries.agregate({[{"$group":{"_id": "service.nomser","nombre":{"$count":"$]})
```

Partie Pratique (60 pts)

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

A partir du schéma relationnel (voir Préliminaire)

1- Créer une procédure stockée qui affiche le nombre de salariés de chaque service. **(4 pts)**

```
DELIMITER $$
CREATE PROCEDURE P1()
begin
select count(*) as nb, nomSer from salarié,service
where salarié.codeSer=service.codeSer group by nomSer ;
END$$
DELIMITER ;
```

2- Créer une fonction permettant de retourner le code du salarié utilisant une voiture dont le matricule et la date sont passés en paramètre. **(4 pts)**

```
delimiter $$
create function Fonction1( mat varchar(20) , dt  date) returns varchar(20)
READS SQL DATA
BEGIN
declare codeS varchar(20);
select codeSal into codeS from utilisation where matricule=mat and
dt between datedébutUtilisation and dateFinUtilisation;
return codeS;
end$$
DELIMITER ;
```

3- Créer un déclencheur (Trigger) qui empêche, lors de l'insertion d'une nouvelle utilisation, d'affecter une voiture pour un salarié attaché au service *'informatique'*. **(4pts)**

```
CREATE TRIGGER `tg1` BEFORE INSERT ON `utilisation`
 FOR EACH ROW BEGIN
DECLARE X varchar(255) ;
select nomSer into X from service, salarié where salarié.codeSer=service.codeSer
and NEW.codeSal=salarié.codeSal;
IF X='informatique' THEN
 SIGNAL SQLSTATE '45000'
   SET MESSAGE_TEXT = 'non autorisé';
END IF;
END
```

Dossier 2 : Développement front-end (24 pts)

1- Réaliser un composant **composant1.js** permettant à l'utilisateur de saisir les informations d'une voiture, lors du clic sur le bouton confirmer on doit afficher le récapitulatif des informations sur la même page comme suit : **(4 pts)**

```
import React from 'react';
export default class composant1 extends React.Component{
constructor()
{super();
   this.state= {matricule:'',marque:'', datem:'',couleur:''}
  }
Afficher(e)
```

```
{ e.preventDefault();
  this.setState({
   matricule:e.target[0].value,
   marque:e.target[1].value,
   datem:e.target[2].value,
   couleur:e.target[3].value
   })
}
render()
{
   return ( <div className="container">
              <h1> Gestion Voitures</h1>
               <form onSubmit={(e)=>{this.Afficher(e)}}>
               <div className="form-group mb-5">
                <label className="form-label">Matricule:</label>
                 <input type="text"  className="form-control" />
                 <label className="form-label">Marque:</label>
                 <select className='form-control'>
                   <option value='Toyota'>Toyota</option>
                   <option value='Hyundai'>Hyundai</option>
                   <option value='Honda'>Honda</option>
                 </select>
              <label className="form-label">Date de mise en circulation:</label>
              <input type="date"  className="form-control"/>
              <label className="form-label"> Couleur:</label>
               <input type="text"  className="form-control" />

                 <button  class="btn btn-primary"> Confirmer </button>
                 </div>
               </form>
               {this.state.matricule==""?"":<div>
                <h1> Récapitulatif des informations :</h1>
                <ul>
                  <li>Matricule :{this.state.matricule}  </li>
                  <li>Marque :  {this.state.marque}</li>
              <li>Date Mise en circulation : {this.state.datem} </li>
                  <li>Couleur :   {this.state.couleur} </li>
                </ul>
                </div> }
   </div>)
}
}
```

2- Initialiser la variable du state ***salaries*** du **composante principale App.js** avec les résultats de l'api du backend **(4 pts)**

```
 const [salaries,setSalaries]=useState([])
  fetch(' http://localhost:8000/salaries ')
    .then((response)=>{ console.log(response); return response.json()})
    .then((salaries)=>{setSalaries(salaries);})
```

3- Afficher les informations du salariés stockées dans la variable du state ***salaries*** du **composante App.js** dans la composante **composant2 .js** :**(6 pts)**

```
function composant2(props) {
   return (
    <main>
      <div className="container">
      <h1>Liste des Salariés</h1>
        <table className="table">
          <tr>
        <th>nom</th>
           <th>prénom</th>
           <th>Fonction</th>
           <th>Service</th>
          </tr>
        {props.salaries.map((s) => (
          <tr>
           <td>  {s.nomsal}</td>
           <td>  {s.prenomsal} </td>
           <td>  {s.fonction} </td>
           <td>{s.service.nomser}</td>
          </tr>
       ))}
       </table></div></main>
   )}
 export default Salaries;
```

4- Créer la composante **composant3.js** permettant d'effectuer une recherche par service sur les données stockées dans la variable du state ***salaries*** du **composante App.js**, si le service contient des salariés, les afficher sous forme de liste, sinon afficher le message « Aucun salarié n'est affecté à ce service » **(6 pts)**

```
import React, { useState } from "react";
 export default function Rechercher(props)
 {  const [term,setTerm]=useState('')
 function onChercheSubmitBar(event)
   {  event.preventDefault()
      props.onChercheSubmit(term)
   }
   return (
       <div className="container">
       <form onSubmit={(event)=>onChercheSubmitBar(event)}>
       <h2>Recherche par service: </h2>
       <div>
         <label>Entrer le nom du service:</label>
         <input type="text" className="form-control" value={term}
         onChange={(event)=>setTerm(event.target.value)} />
         </div>
          <button className="btn btn-primary"type="submit">chercher</button>
          </form>  </div>   ); }
 import React from "react";
export default function ResultatList(props)
{
    return (
    <div className="container">
    <h1>Résultat</h1>
    {props.resultats.length == 0 ? ( <p>aucun salarié affecté à ce service</p> ) :
    ( <div className="list">
```

```
      <ul> {props.resultats.map((item) =>
       { return <li key={item.nomsal}>Nom : {item.nomsal} Prénom : {item.prenomsal}</li>; })}
       </ul>
        </div> )}
         </div>
         );
}
```

5- Ajouter dans la composante principale **App.js** la partie de routage pour les deux composantes **composant2.js** et **composant3.js (4 pts)**

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-
dom';
import C2 from './Composant2';
import C3 from './Composant3';

function Menu() {
    return (
        <Router>
        <div>
          <ul>
            <li><Link to={'/composant2'} > Composant2 </Link></li>
            <li><Link to={'/composant3'} > Composant3</Link></li>
          </ul>
          <hr />
          <Routes>
                <Route path='/composant2' element={<C2/>} />
                <Route path='/composant3'  element={<C3/>} />
        </Routes>
        </div>
      </Router>

    );
}
export default Menu;
```

Dossier 3 : Développement Back-end (24 pts)

1. Créer la migration de la table ***Utilisation*** **(2 pts)**

php artisan make:migration create_utilisation_table

```
  Schema::create('utilisations', function (Blueprint $table) {
            $table->string('matricule');
            $table->string('codeSal');
            $table->date('dateDebutUtilisation');
            $table->date('datefinUtilisation');
            $table->primary(['matricule', 'codeSal'  ,
'dateDebutUtilisation']);
            $table->foreign('matricule')->references('matricule')-
>on('voiture');
            $table->foreign('codeSal')->references('codeSal')->on('salarie');
});
```

2. Créer des modèles pour chacune des tables suivantes: ***voiture***, ***salarié*** et ***service*****(5pts)**

```
class Voiture extends Model
{
        protected $primaryKey = 'matricule';
         protected $table = 'voiture';
         public $incrementing = false ;
         protected
$fillable=['matricule','marque','couleur','dateMiseEnCirculation'];
        public function salariés() {
        return $this->belongsToMany(Salarié::class);
        }
        public function salariésUs(){
                return $this->belongsToMany(Salarié::class,
 'utilisation', 'matricule', 'codeSal')
 ->withPivot('dateDébutUtilisation','dateFinUtilisation');  }
}
```

```
class Salarié extends Model
{
    use HasFactory;
    protected $primaryKey = 'codeSal';
    protected $table = 'salarié';
    public function service(){
        return $this->belongsTo(Service::class,'codeSer');
        }
        public function voitures(){

            return $this->belongsToMany(Voiture::class,
            'utilisation', 'codeSal', 'matricule')
            ->withPivot('dateDébutUtilisation','dateFinUtilisation')  ;   }
}
```

```
class service extends Model
{
    use HasFactory;
    protected $table = 'service';
    protected $primaryKey = 'codeSer';
    public function salariés(){
        return $this->hasMany(Salarié::class,'codeSer');
        }
       }
```

3. Créer le contrôleur **VoitureController** ayant les méthodes: **(5pts)**

```
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
```

```php
use App\Models\Voiture;
class VoitureController extends Controller
{
   function afficher()
   {
$listeV=Voiture::all();
return view ('voiture.index',['data'=>$listeV]);
   }
   function ajouter(Request $request )
   {
     $a=$request->input('matricule');
     $b=$request->input('marque');
     $c=$request->input('couleur');
     $d=$request->input('dateMiseEnCirculation');

     $voiture = new Voiture([
     'matricule' => $a,
     'marque'=>$b,
     'couleur'=>$c,
     'dateMiseEnCirculation'=>$d
     ]);
    $voiture->save();
     return redirect()->route('accueil1');
}
function créer(){
   return view('voiture.créer');
}
function supprimer($code)
{
   $v=Voiture::find($code);
   $v->delete();
   return redirect()->route('accueil1');
}
}
```

4. Créez la vue **index.blade.php** du dossier voiture pour afficher la liste des voitures dans un tableau, avec des liens ajouter et supprimer**. (3 pts)**

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Models\Salarié;
class SalariéController extends Controller
{
    function afficher()
    {
 $listes=Salarié::with(['service'])->withCount(['voitures'])->get();
 return   view ('salarié.index',['data'=>$listes]);
}
function rechercher($code)
    {
```

```
        $sal = Salarié::find($code);
       $voitures = $sal->voitures()-
>orderby('dateDébutUtilisation','desc')->get();
return view ('salarié.rechercher',['data'=>$voitures,'code'=>$code]);
    }

}
```

5. Créer le contrôleur **SalariéController** avec les méthodes: **(5pts)**

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Models\Salarié;
class SalariéController extends Controller
{
    function afficher()
    {
 $listes=Salarié::with(['service'])->withCount(['voitures'])->get();
 return   view ('salarié.index',['data'=>$listes]);
}
function rechercher($code)
    {
        $sal = Salarié::find($code);
       $voitures = $sal->voitures()-
>orderby('dateDébutUtilisation','desc')->get();
return view ('salarié.rechercher',['data'=>$voitures,'code'=>$code]);
    }
}
```

6. Ecrire le code du fichier web.php contenant les routes des méthodes déjà crées des contrôleurs : **VoitureController** et **SalariéController. (3pts)**

```
use App\Http\Controllers\SalariéController;
use App\Http\Controllers\VoitureController;
Route::get('/voiture',[VoitureController::class,'afficher'])-
>name('accueil1');
Route::get('/voiture/ajouter',[VoitureController::class,'créer']);
Route::post('/voiture/ajouter',[VoitureController::class,'ajouter']);
Route::get('/voiture/supprimer/{code}',[VoitureController::class,'supprimer
']);
Route::get('/salarié',[SalariéController::class,'afficher']);
Route::get('/salarié/rechercherVoitures/{code}',[SalariéController::class,'
rechercher']);
```