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	**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.**

Partie Théorique (40 pts)

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

1- Définir un cloud privé, citez deux exemples de fournisseurs. (2 pts)

Un Cloud privé est un environnement Cloud réservé à un client final, la plupart du temps hébergé dans votre centre de données et mis à jour par votre équipe informatique Même si le Cloud privé était jusqu'à présent habituellement exécuté sur site, les entreprises construisent désormais des Cloud privés dans des datacenters hors site qu'elles louent auprès de fournisseurs.

Exemple de fournisseurs : Maroc Télécom, Inwi Business….

2- Citer les types de services Cloud (2 pts)

Il existe trois principaux types de cloud computing « as a Service », chacun offrant un certain degré de gestion

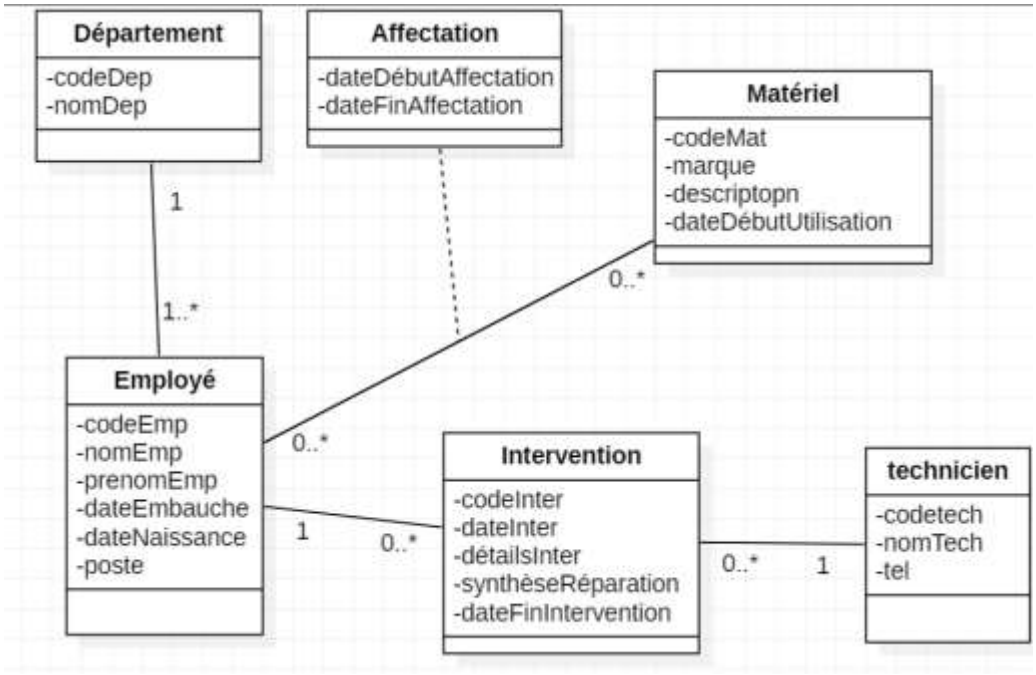
- ✓ IaaS (Infrastructure as a Service)
- ✓ PaaS (Platform as a Service)
- ✓ SaaS (Software as a Service) …

3- C'est quoi un conteneur ? c'est quoi la différence entre une machine virtuelle et un conteneur (4 pts)

Un package logiciel unique, appelé « conteneur », regroupe le code d'une application avec les fichiers de configuration, les bibliothèques et les dépendances requises pour que l'application puisse s'exécuter.
La principale différence entre les conteneurs et les machines virtuelles ? Les machines virtuelles virtualisent toute une machine jusqu'aux couches matérielles, tandis que les conteneurs ne virtualisent que les couches logicielles au-dessus du niveau du système d'exploitation.

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

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

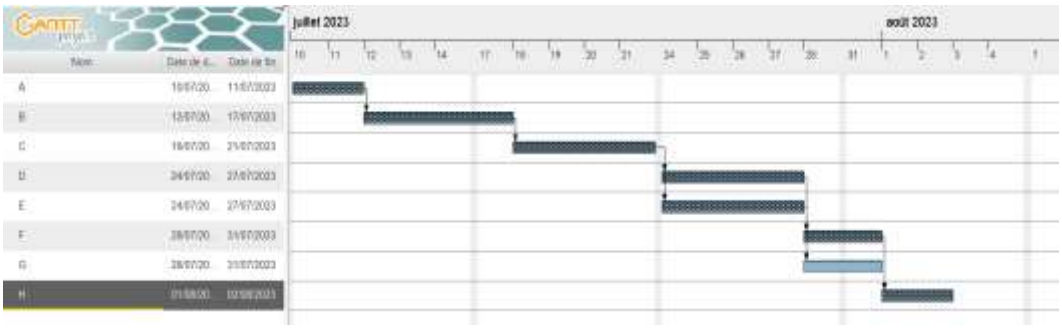


Dossier 3 : (Approche Agile) (15 pts)

La société vous a contacté pour créer le site permettant de gérer les affectations du matériel et les interventions effectuées par les techniciens, après L'évaluation des besoins du projet, le chef de projet a tracé le tableau des tâches suivantes :

Tâche	intitulé	Durée en jours	Tâches antérieures
A	Cadrage du projet	2	-
B	Analyse	4	A
C	Conception	4	B
D	Réalisation des IHM	4	C
E	Création de la BDD	4	C
F	Tests et recette	2	D,E
G	Rédaction des manuels d'utilisation	2	D
H	Hébergement et déploiement	2	F

1. Dresser le diagramme de GANTT en supposant que le projet démarrera le lundi 10/07/2023. **(3 pts)**



2. Déterminer le chemin critique et indiquer la date fin du projet **(2 pts)**
   Chemin critique : A,B,C,D,E,F,H
   Date fin de projet : 02/08/2023
3. On souhaite de travailler avec la méthode Scrum pour réaliser le projet précédent :
   a- Quels sont les 3 artefacts Scrum ? **(2 pts)**
      - ✓ Sprint backlog
      - ✓ Product backlog
      - ✓ Increment product
   b- Rédiger trois user stories à intégrer dans le backlog produit (3 pts)
4. On suppose maintenant que vous utilisez un outil de gestion des versions avec votre équipe de développement (commandes git)
   a- Ecrire la commande qui ajoute des fichiers à un commit? **(1 pt)**
      git add
   b- Ecrire la commande qui vérifie l'état de votre référentiel local depuis votre dernier commit? **(1 pt)**
      git status
   a- Ecrire la commande qui télécharge votre référentiel de GitHub/Gitlab sur votre ordinateur? **(1 pt)**
      git clone
5. On veut mesurer la qualité de notre code avec SonarQube, Quelles sont ses fonctionnalités? **(2 pts)**
   Il est utilisé pour inspecter le code source des logiciels et applications en développement et détecter des bugs, vulnérabilités de sécurités, instances de code dupliqué et autres anomalies pouvant nuire à la qualité du code source, et ainsi au fonctionnement de l'application qui en résulte.

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

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

```
{
"_id" :"e1","nomEmp" :"Lamrabet","prenomEmp":"Oussama",
"poste" : "Directeur",
"departement" :{ "codeDep" : "1","nomDep" : "RH"}
}
```

```
//Création de la base de données
use DBEmployes
//Création de la collection
db.createCollection("employes")
//Création des documents
db.employes.insert({ "_id" : "e1",
"nomEmp" : "Lamrabet",
"prenomEmp" : "Oussama",
```

```
"poste" : "Directeur",
"departement" :{ "codeDep" : "1","nomDep" : "RH"}})
```

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

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

```
db.employes.find().sort({"nomEmp":1})
```

3- Afficher le nombre des employés ayant le poste "Directeur" **(2 pts)**

```
db.employes.find({"poste" : "Directeur"}).count()
```

4- Supprimer l'employé ayant l'_id "e5" **(2 pts)**

```
db.employes.deleteOne({"_id","e5"})
```

5- Afficher le nombre d'employés par poste **(2 pts)**

```
db.employes.agregate({[{"$group":{"_id": "departement.nomdep","nombre":{"$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 d'employés de chaque département. **(4 pts)**

```
DELIMITER $$

CREATE PROCEDURE P1()

begin

select count(*) as nb, nomDep from employé,département

where département.codeEmp= département.codeEmp group by nomDep ;

END$$

DELIMITER ;
```

2- Créer une fonction permettant de retourner le code de l'employé utilisant un matériel dont le code et la date sont passés en paramètre. **(4 pts)**

```
delimiter $$
create function F1( mat varchar(20) , dt  date) returns varchar(20)
READS SQL DATA
BEGIN
declare codeE varchar(20);
select codeEmp into codeE from affectation where codeMat=mat and
dt between datedébutAffectation and dateFinAffectation;
return codeE ;
end$$
DELIMITER
```

3- Créer un déclencheur (Trigger) qui empêche, lors de l'insertion d'une nouvelle affectation, d'affecter un matériel pour un employé attaché au département 'RH'. **(4pts)**

```
CREATE TRIGGER `tg1` BEFORE INSERT ON `affectation`

FOR EACH ROW BEGIN

DECLARE X varchar(255) ;
```

```
select nomDep into X from département, employé where employé.codeDep=déaprtement.codeDep
and NEW.codeEmp=employé.codeEmp;
IF X=’RH’ THEN
 SIGNAL SQLSTATE '45000'
   SET MESSAGE_TEXT = 'non autorisé';
END IF;
END
```

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

1- Réaliser un composant permettant à l’utilisateur de saisir les informations d’un matériel, 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 Materiel extends React.Component{
constructor()
{super();
   this.state= {CodeMat:'',marque:'', dated:'',categorie:''}

  }
Afficher(e)
{
   e.preventDefault();
  this.setState({
   CodeMat:e.target[0].value,
   marque:e.target[1].value,
   dated:e.target[2].value,
   categorie:e.target[3].value
   })
}
render()
{
   return ( <div className="container">
              <h1> Gestion Matériel</h1>
               <form onSubmit={(e)=>{this.Afficher(e)}}>
               <div className="form-group mb-5">
                <label className="form-label">Code Matériel:</label>
                 <input type="text"  className="form-control" />
                 <label className="form-label">Marque:</label>
                 <select className='form-control'>
                   <option value='HP'>HP</option>
                   <option value='Siemens'>Siemens</option>
                   <option value='Toshiba'>Toshiba</option>
                 </select>
               <label className="form-label">Date début utilisation:</label>
               <input type="date"  className="form-control"/>
               <label className="form-label"> Catégorie:</label>
                <input type="text"  className="form-control" />

                  <button  class="btn btn-primary"> Confirmer </button>
                  </div>
```

```
            </form>
            {this.state.CodeMat==""?"":<div>
             <h1> Récapitulatif des informations :</h1>
             <ul>
               <li>Code Matériel :{this.state.CodeMat}  </li>
               <li>Marque :  {this.state.marque}</li>
               <li>Date début utilisation : {this.state.datd} </li>
               <li>catégorie :   {this.state.categorie} </li>
             </ul>
             </div> }
  </div>)
}
}
```

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

Méthode HTTP	GET
URL de l'api	http://localhost:8000/employes

Résultat	[
	{
	"_id" : "e1",
	"nomEmp" :"Lamrabet",
	"prenomEmp" :"Oussama",
	"poste" : "Directeur",
	"département" :{
	"codeDep" : "1",
	"nomDep" : "RH"
	}
	},
	....
	]

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

3- Afficher les informations des employés stockés dans la variable du state ***employes*** du **composante App.js** dans la composante **composant1 .js** (voir figure2): **(6 pts)**

```
function composant1(props) {
  return (
   <main>

    <div className="container">
    <h1>Liste des Employés</h1>
      <table className="table">
       <tr>
        <th>nom</th>
        <th>prénom</th>
        <th>Poste</th>
        <th>Département</th>
```

```
          </tr>
         {props.employes.map((s) => (
          <tr>
            <td>  {s.nomEmp}</td>
            <td>  {s.prenomEmp} </td>
            <td>  {s.poste} </td>
            <td>{s.departement.nomDep}</td>
          </tr>
       ))}
       </table>
     </div>
   </main>
  )
 }
 export default Employes;
```

4- Créer la composante **composant3.js** permettant d'effectuer une recherche par département sur les données stockées dans la variable du state ***employes*** du **composante App.js** , si le département contient des employés, les afficher sous forme de liste, sinon afficher le message « Aucun employé n'est affecté à ce département » **(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 Département: </h2>
          <div>
            <label>Entrer le nom du Département:</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 Employé affecté à ce Département</p> ) :
        ( <div className="list">
          <ul> {props.resultats.map((item) =>
           { return <li key={item.nomEmp}>Nom : {item.nomEmp} Prénom :
{item.prenomEmp}</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 C3 from './Composant3';
import C2 from './Composant2';

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

    );
}
export default Menu;
```

Dossier 3 : back end (24 pts)

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

php artisan make:migration create_affectation_table

```
  Schema::create('affectations', function (Blueprint $table) {
            $table->string('codeMat');
            $table->string('codeEmp');
            $table->date('dateDebutAffectation');
            $table->date('datefinAffectation');
            $table->primary(['codeMa', 'codeEmp'   , 'dateDebutAffectation']);
            $table->foreign('codeMa')->references('codeMa')->on('materiel');
            $table->foreign('codeEmp')->references('codeEmp')->on('employer');
        });
```

2. Créer des modèles pour chacune des tables suivantes: matériel, employé et département **(5 pts)**

```
class Matériel extends Model
{
        protected $primaryKey = codeMat;
         protected $table = matériel;
         public $incrementing = false ;
 protected $fillable=[codeMat, 'marque', 'description',
'dateDébutUtilisation'];
        public function employés() {
        return $this->belongsToMany(Employé::class);
        }
        public function employésUs(){
                return $this->belongsToMany(Employé::class,
 'affectation', 'codeMat' , 'codeEmp')
 ->withPivot('dateDébutAffectation','dateFinAffectation');  }
}
```

```
class Employé extends Model
{
    use HasFactory;
    protected $primaryKey = 'codeEmp';
    protected $table = employé;
    public function service(){
        return $this->belongsTo(Département::class,'codeDep');
        }
        public function matériels(){

            return $this->belongsToMany(Matériel::class,
            'affectation', 'codeEmp', 'codeMat')
            ->withPivot('dateDébutAffectation','dateFinAffectation')  ;   }
}
```

```
class Département extends Model
{
```

```
    use HasFactory;
    protected $table = département;
    protected $primaryKey = 'codeDep';
    public function employés(){
        return $this->hasMany(Employé::class,'codeEmp');
        }
       }
```

3. Créer le contrôleur **MatérielController** ayant les méthodes:

```
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Models\Matériel;
class MatérielController extends Controller
{
   function afficher()
   {
$listeM=Matériel::all();
return view ('matériel.index',['data'=>$listeM]);
   }
   function ajouter(Request $request )
   {
     $a=$request->input('codeMat');
     $b=$request->input('marque');
     $c=$request->input(description);
     $d=$request->input('datedébutUtilisation');

     $matériel = new Matériel([
     'codeMat' => $a,
     'marque'=>$b,
     'description'=>$c,
     'dateDébutUtilisation'=>$d
     ]);
    $ matériel->save();
     return redirect()->route('accueil1');
}
function créer(){
   return view('matériel.créer');
}
function supprimer($code)
{
   $v=Matériel::find($code);
   $v->delete();
   return redirect()->route('accueil1');
}
}
```

4. Créez la vue **index** du dossier matériel pour afficher la liste des matériels dans un tableau, avec les liens ajouter et supprimer**. (3 pts)**

```
@extends('template')
```

```
@section( 'contenu')
<br>
<div class="container">
<div class="row card text-black bg-white">
<h4 class="card-header">liste des matériels</h4>
<a  class="btn btn-success"  href='/matériel/ajouter'> ajouter une nouveau
matériel</a>
<br/>
<table class="table table-hover">
    <tr> <th> code </th>
    <th> marque  </th>
    <th> description </th>
    <th> date début d'utilisation</th>
    <th> supprimer </th>

</tr>
    @foreach($data as $c)
    <tr> <td>{{$c->codeMat}}</td>
   <td> {{$c->marque}}</td>
   <td> {{$c->description}}</td>
   <td>{{$c->dateDébutUtilisation}}</td>
   <td><a class="btn btn-danger"
    href='/matériel/supprimer/{{$c->codeMat}}'  >
     supprimer </a></td>

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

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

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Models\Employé;
class EmployéController extends Controller
{
    function afficher()
    {
 $listes=Employé::with(['département'])->withCount(['matériels'])->get();
 return   view ('employé.index',['data'=>$listes]);
}
function rechercher($code)
    {
        $emp = Employé::find($code);
       $matériels = $emp->matériels()-
>orderby('dateDébutAffectation','desc')->get();
return view('employé.rechercher',['data'=>$matériels,'code'=>$code]);
    }
}
```

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

```
use App\Http\Controllers\EmployéController;
use App\Http\Controllers\MatérielController;
Route::get('/matériel,[ MatérielController::class,'afficher'])-
>name('accueil1');
Route::get('/matériel/ajouter',[ MatérielController::class,'créer']);
Route::post('/ matériel/ajouter',[ MatérielController::class,'ajouter']);
Route::get('/ matériel/supprimer/{code}',[
MatérielController::class,'supprimer']);
Route::get('/ employé,[ EmployéController::class,'afficher']);
Route::get('/employé/rechercherMatériels/{code}',[
EmployéController::class,'rechercher']);
```