Partie théorique:(40 PTs)

Dossier 1 : Créer une application Cloud Native (8pts)

1. Comparez les 2 modèles ***du Cloud Computing : SaaS et PaaS*** de point de vue des utilisateurs du service par rapport au fournisseur. Vous pouvez expliquer avec un schéma **(2pts)**
2. Une application Cloud native se base sur des piliers, parmi lesquels: **Microservices** et **Livraison en continue** *(CD/CI)*. Donner une définition de ces deux piliers **(3 pts)**
3. Qu'est-ce qu'une image ***Docker*** ? **(1.5 pts)**
4. Donner la commande pour instancier une image ***Docker*** dans un conteneur **(1.5 pts)**

- PaaS : Platform as a Service :

Exemples : AWS Elastic Beanstalk, Google App Engine, Azure App Service, Azure function App.

- SaaS : Software as a Service :

Exemples : Zoom, Google Apps, Microsoft Office 365 ...

Définition de l'architecture micro-services :

L'architecture micro-services consiste à décomposer l'application en un ensemble de petites services indépendantes appelés micro-services. Chaque micro-service possède son propre logique et sa propre base de données.

2)

Livraison en continu (CD/CI):

- **CI (Intégration Continue)**: Pratique de fusionner régulièrement le code des développeurs dans un dépôt central, suivi d'automatisation des tests et de la construction de l'application.
- **CD (Livraison Continue)**: Prolongement de la CI, où les modifications de code sont automatiquement déployées en production après avoir passé les tests, réduisant le temps de mise en production.

3) **Qu'est-ce qu'une image Docker ? (1.5 pts)**

- Une image Docker est un modèle léger, autonome et exécutable qui inclut tout ce qui est nécessaire pour faire fonctionner un logiciel : le code, les runtimes, les bibliothèques, les variables d'environnement et les configurations par défaut. Les images Docker sont utilisées pour créer des conteneurs, qui sont des environnements isolés pour l'exécution des applications.

4) **Donner la commande pour instancier une image Docker dans un conteneur (1.5 pts)**

- La commande pour instancier une image Docker dans un conteneur est : `docker run <image_name>`

Dossier 2 : Adopter l'approche Agile(14pts)

Après avoir défini le backlog product du projet, l'équipe Scrum*(Product Owner, Scrum Master et les Développeurs)* a organisé un sprint planning lors duquel ils ont défini les tâches du premier sprint :

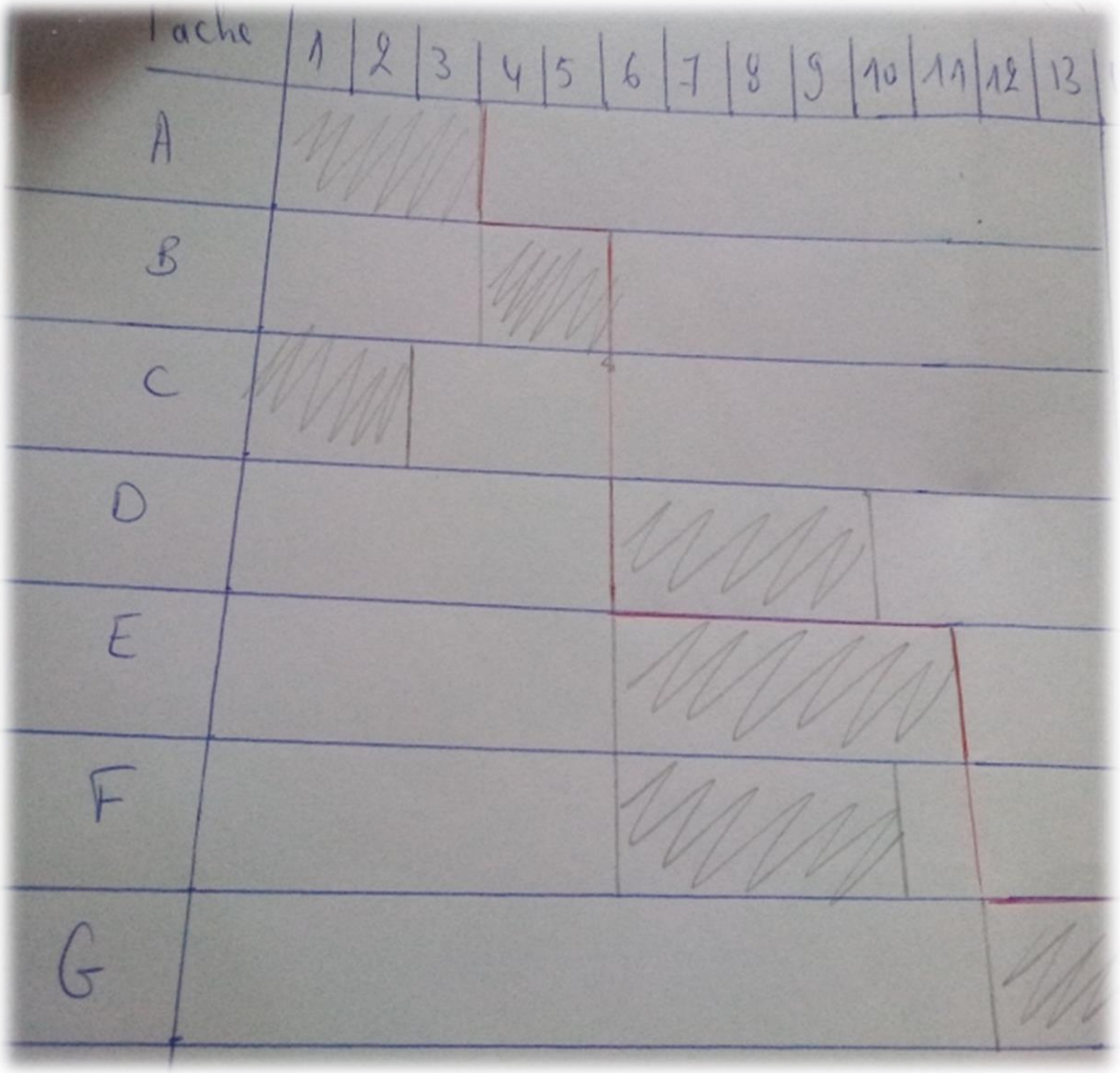
TÂCHE	DURÉE (en jours)	PRÉDÉCESSEURS
A	3	-
B	2	A
C	2	-
D	4	B,C
E	5	B,C
F	4	B,C
G	3	D ,E ,F

Avec :

A : Créer les maquettes	E : Développement du composant 2
B : Création des interfaces html	F : Développement du composant 3
C : Création des tables	G : Test d'intégration
D : Développement du composant 1	

Questions :

1. Définir le backlog product.**(2,5pts)**
2. C'est quoi le rôle du product owner **?(2,5pts)**
3. Réaliser le diagramme de Gantt **(5pts)**
4. Identifier le chemin critique **(4pts)**



Questions :

1. **Définir le backlog product. (2.5 pts)**
    - **Backlog Product**:
        - C'est une liste ordonnée de tout ce qui est connu nécessaire dans le produit. Il est la seule source de travail requis pour l'équipe Scrum. Les éléments de backlog sont souvent des fonctionnalités, des exigences, des améliorations et des corrections de bugs.
        - Il est dynamique, évoluant constamment pour identifier ce dont le produit a besoin pour être adapté, concurrentiel et utile. Le Product Owner est responsable de la gestion et de la visibilité du backlog product.

C’est quoi le rôle du Product Owner ? (2.5 pts)

- **Rôle du Product Owner**:
    - Le Product Owner est responsable de maximiser la valeur du produit résultant du travail de l'équipe Scrum.
    - Gérer le backlog produit en assurant qu'il est visible, transparent et clair pour tous, montrant clairement sur quoi l'équipe Scrum travaillera ensuite.
    - Prioriser les éléments du backlog en fonction des besoins du marché, des clients et des parties prenantes, et être disponible pour clarifier les questions de l'équipe de développement.
    - Travailler avec les parties prenantes pour comprendre leurs besoins et aligner les exigences produit avec les objectifs stratégiques de l'entreprise.

Identification du Chemin Critique

Le chemin avec la durée la plus longue détermine le chemin critique. D'après les calculs ci-dessus, le chemin **A → B → E → G** a la durée la plus longue avec **13 jours**.

Dossier 3: Gérer les données (partie :MongoDB) (12pts)

Nous utiliserons une base de données contenant des informations sur les voitures dédouanées par les marocains residants à étranger(MRE). Chaque voiture dédouanée est caractérisée par:

- idvoiture : identifiant de la voiture.
- marque : marque de la voiture.
- Date_circulation : date de la mise en circulation
- Montant_neuf :sa valeur à neuf
- Date_dedouanement :date de dedouanement
- Valeur_actuelle :la valeur actuelle de la voiture
- proprietaire : un document décrivant le propriétaire de la voiture contenant les attributs suivants
  - Num_pass : identifiant du passeport.
  - Pays_accueil : payé d'accueil.
  - situation : situation familiale.
  - Adresse_acceuil :adresse de l'mre dans le pays d'accueil

- Adresse_local :adresse de l'mre dans son pays d'origine
- lieu_naissance :ville de naissance de l'habitant

Écrivez les requêtes suivantes pour :

1. Afficher le nombre total des propriétaires par pays d'accueil. **(2.5pts)**
2. Afficher les propriétaires qui sont nés à Paris.**(2.5pts)**
3. Afficher les propriétaires qui ont une voiture mise en circulation avant le '2023-02-11'**(2pts)**
4. Diminuer de 2% la valeur actuelle de toutes les voitures.**(2.5)**
5. Supprimer toutes les voitures du propriétaire num_pass=P1**(2.5pts)**

1)
```
db.voitures.aggregate([
  {
    $group: {
      _id: "$proprietaire.Pays_accueil",
      total_proprietaires: { $sum: 1 }
    }
  }
])
```

2.
```
db.voitures.find({
    "proprietaire.lieu_naissance": "Paris"
  })
```

3.
```
db.voitures.find({
    "Date_circulation": { $lt: ISODate("2023-02-11") }
```

})

4. db.voitures.updateMany(

{},

{ $mul: { "Valeur_actuelle": 0.98 } }

)

5. db.voitures.deleteMany({

"proprietaire.Num_pass": "P1"

})

3. Code des modèles associés à la table demande_dedouanement et définir les relations

```php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class DemandeDedouanement extends Model
{
    use HasFactory;

    protected $table = 'demande_dedouanement';

    public function voiture()
    {
        return $this->belongsTo(Voiture::class);
    }

    public function beneficiaire()
    {
        return $this->belongsTo(Beneficiaire::class);
    }
}

// app/Models/Voiture.php

namespace App\Models;
```

```php
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Voiture extends Model
{
    use HasFactory;

    public function demandes()
    {
        return $this->hasMany(DemandeDedouanement::class);
    }
}

// app/Models/Beneficiaire.php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Beneficiaire extends Model
{
    use HasFactory;

    public function demandes()
    {
        return $this->hasMany(DemandeDedouanement::class);
    }
}
```

Dossier4: préparer un projet web(6pts)

On souhaite faire la conception et l'analyse d'une application de gestion des demandes de dédouanement des voitures importées avec la méthode UML. Les travaux de l'ancienne équipe chargée de l'analyse et de la conception de l'application ont abouti au modèle relationnel suivant :

- **voiture (numvoiure**, #cin,date_mise_circulation, marque, modele, #id_pays_importation, valeur_neuf,valeur_actuelle, date_entree_territoire)
- **pays**(idpays, nompays)
- **benificiaire (cin**, nom, prenom,ddn,adresse,tel,type_benificiaire)
- **demande_dedouanement(iddemande**,#numvoiture,date_demande,etat)
- **paiement_dedouanement (idpaiement**,#**iddemande**, date_paiement, montantpaye, mode_paiement)

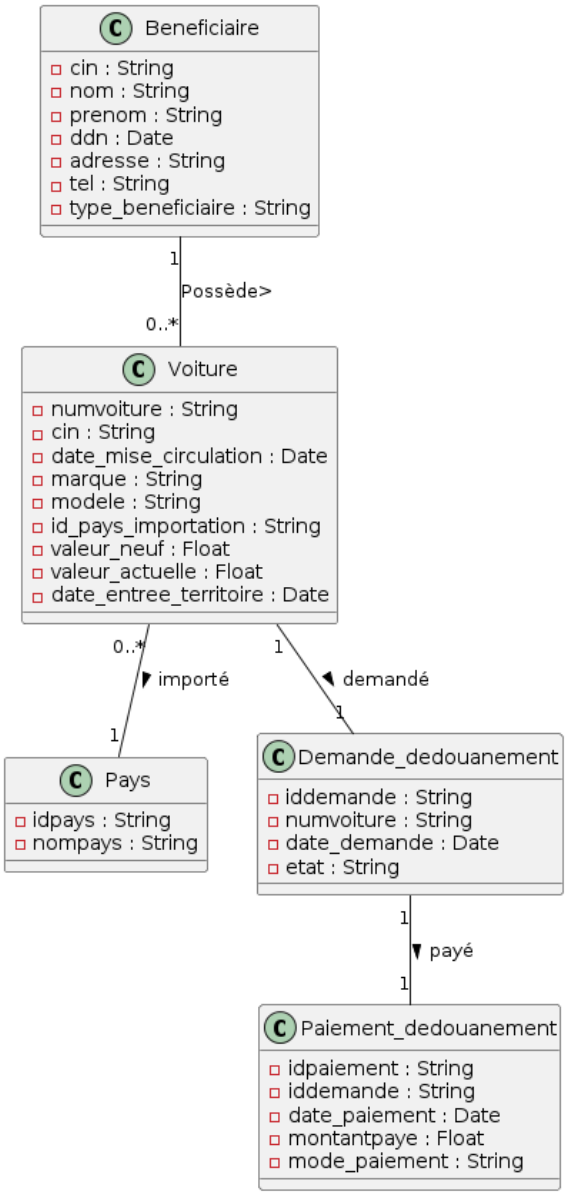
NB :

- le champ état prend l'une des valeurs : « encours », « rejetée »ou « validée »
- les champs marqués en gras et soulignés représentent les clés primaires des table, les champs marqués par # représentent les clés étrangères
- toutes les clés primaires se sont des chaines de caracteres

On précise que :

- Chaque voiture est importée d'un seul pays.
- Un bénéficiaire peut avoir plusieurs voitures.
- Le paiement effectué concerne une voiture.
- Une voiture figure dans une seule demande de dédouanement.
- Une demande de dédouanement concerne une seule voiture.
- Le paiement d'une demande de dédouanement s'effectue une seule fois et ceci si la demande est acceptée

1. En se basant sur le MLD déjà établi et points précisés, réaliser le digramme de classe équivalent.**(6pts)**


Beneficiaire
cin : String
nom : String
prenom : String
ddn : Date
adresse : String
tel : String
type_beneficiaire : String
1
Possède>
0..*
Voiture
numvoiture : String
cin : String
date_mise_circulation : Date
marque : String
modele : String
id_pays_importation : String
valeur_neuf : Float
valeur_actuelle : Float
date_entree_territoire : Date
0..*
importé
1
1
demandé
1
Pays
idpays : String
nompays : String
Demande_dedouanement
iddemande : String
numvoiture : String
date_demande : Date
etat : String
1
payé
1
Paiement_dedouanement
idpaiement : String
iddemande : String
date_paiement : Date
montantpaye : Float
mode_paiement : String

Partie Pratique :(60 Pts)

Dans cette partie, vous devez utiliser le modèle relationnel de la partie théorique. (Voir Dossier 4 – Préparer un projet web)

Dossier 5 : Gérer les données (partie : MySql) (12pts)

1. Écrire une requête SQL qui affiche les voitures de marque « peugeot » importées de la France et qu'ont été mise en circulation cette année. **(3pts).**
2. Écrire une fonction qui retourne, pour un bénéficiaire passé en paramètre, le nombre de ses demandes de dédouanement rejetées et le nombre de ses demandes validées. **(3pts).**
3. Écrire une procédure qui affiche les années où on a rejeté le plus de demandes. **(3pts).**
4. Écrire un déclencheur qui empêche le paiement des droits de dédouanement d'une voiture si sa demande de dédouanement est rejetée.**(3pts)**

1.
```
SELECT *
FROM voitures
WHERE marque = 'peugeot'
  AND pays_importation = 'France'
  AND YEAR(date_mise_circulation) = YEAR(CURDATE());
```

2.
```
CREATE FUNCTION nombre_demandes(beneficiaire_id INT)
RETURNS TABLE(
  nombre_rejetes INT,
  nombre_valides INT
)
BEGIN
  RETURN (
    SELECT
      SUM(CASE WHEN statut = 'rejetée' THEN 1 ELSE 0 END) AS nombre_rejetes,
      SUM(CASE WHEN statut = 'validée' THEN 1 ELSE 0 END) AS nombre_valides
    FROM demandes_dedouanement
```

```
   WHERE beneficiaire_id = beneficiaire_id
  );
END;
```

3.
```
CREATE PROCEDURE annees_max_rejets()
BEGIN
  SELECT YEAR(date_demande) AS annee, COUNT(*) AS nombre_rejets
  FROM demandes_dedouanement
  WHERE statut = 'rejetée'
  GROUP BY annee
  ORDER BY nombre_rejets DESC;
END;
```

4.
```
CREATE TRIGGER empecher_paiement_rejet
BEFORE INSERT ON paiements
FOR EACH ROW
BEGIN
  DECLARE statut_dedouanement VARCHAR(20);

  SELECT statut INTO statut_dedouanement
  FROM demandes_dedouanement
  WHERE demande_id = NEW.demande_id;

  IF statut_dedouanement = 'rejetée' THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Le paiement des droits de dédouanement est interdit car la demande a été rejetée';
  END IF;
END;
```

Dossier 6: développer en Back end (23 pts)

On veut développer une application web de gestion des demandes de dédouanement. Pour la partie backend, on utilisera les tables **demande_dedouanement** et **voiture** du modèle relationnel de la partie théorique **(Voir Dossier 4 – Préparer un projet web)**

1. Écrire le code de la méthode **up()** et la méthode **down** du fichier migration de la table **demande_dedouanement (2pts).**
2. Écrire le code qui permet de peupler la table **demande_dedouanement** par un jeux d'enregistrement **en utilisant seeder. (2pts).**
3. Écrire le code des deux modèles associés à la table **demande_dedouanement** et **voiture**. Et définir les relations entre eux. **(3pts).**

   N.B : On suppose que les autres modèles sont déjà crées
4. Écrire le code du contrôleur **DedouanementControleur** qui contient les méthodes **:(8pts).**
   a) **rechercher** : qui permet de rediriger vers la vue **Recherche.blade.php** qui affiche la liste des demandes « encours » d'un bénéficiaire passé comme paramètre. Et devant chaque demande, on affiche un lien « valider » pour valider la demande associe **(2pts)**
   b) **valider** : qui permet de valider la demande passée en paramètre et ceci en rendant son champ état= 'validée'. **(2pts)**
   c) **Ajouter :** qui insère une nouvelle voiture dans la table voiture **en respectant les contraintes suivantes :(2pts)**

1. Code de la méthode `up()` et la méthode `down()` du fichier de migration de la table `demande_dedouanement`

```php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateDemandeDedouanementTable extends Migration
{
    public function up()
    {
        Schema::create('demande_dedouanement', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('voiture_id');
            $table->unsignedBigInteger('beneficiaire_id');
            $table->string('statut');
            $table->timestamps();

            $table->foreign('voiture_id')->references('id')->on('voitures')->onDelete('cascade');
            $table->foreign('beneficiaire_id')->references('id')->on('beneficiaires')->onDelete('cascade');
        });
    }

    public function down()
    {
        Schema::dropIfExists('demande_dedouanement');
    }
}
```

2. Code pour peupler la table demande_dedouanement en utilisant un seeder

```php
use Illuminate\Database\Seeder;
use App\Models\DemandeDedouanement;

class DemandeDedouanementSeeder extends Seeder
{
    public function run()
    {
        DemandeDedouanement::factory()->count(50)->create();
    }
}

// database/factories/DemandeDedouanementFactory.php

use App\Models\DemandeDedouanement;
use Illuminate\Database\Eloquent\Factories\Factory;

class DemandeDedouanementFactory extends Factory
{
    protected $model = DemandeDedouanement::class;

    public function definition()
    {
        return [
            'voiture_id' => \App\Models\Voiture::inRandomOrder()->first()->id,
            'beneficiaire_id' => \App\Models\Beneficiaire::inRandomOrder()->first()->id,
            'statut' => $this->faker->randomElement(['en cours', 'validée', 'rejetée']),
        ];
    }
}
```

4. Code du contrôleur DedouanementControleur avec les méthodes demandées

```php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\DemandeDedouanement;
```

```php
use App\Models\Voiture;

class DedouanementControleur extends Controller
{
    // a) rechercher
    public function rechercher($beneficiaireId)
    {
        $demandes = DemandeDedouanement::where('beneficiaire_id', $beneficiaireId)->where('statut', 'en cours')->get();
        return view('Recherche', ['demandes' => $demandes]);
    }

    // b) valider
    public function valider($demandeId)
    {
        $demande = DemandeDedouanement::find($demandeId);
        $demande->statut = 'validée';
        $demande->save();

        return redirect()->back()->with('success', 'Demande validée avec succès.');
    }

    // c) ajouter
    public function ajouter(Request $request)
    {
        $request->validate([
            'marque' => 'required|string|max:255',
            'modele' => 'required|string|max:255',
            'annee' => 'required|integer',
            'pays_importation' => 'required|string|max:255',
        ]);

        Voiture::create([
            'marque' => $request->marque,
            'modele' => $request->modele,
            'annee' => $request->annee,
            'pays_importation' => $request->pays_importation,
        ]);

        return redirect()->back()->with('success', 'Voiture ajoutée avec succès.');
```

```
    }
}
4.d public function suivre_demande($idDemande)
{
    $demande = DemandeDedouanement::find($idDemande);

    if (!$demande) {
        return response()->json(['error' => 'Demande non trouvée.'],
404);
    }

    return response()->json($demande);
}
```

5. Écrire les routes à ajouter dans web.php pour accéder aux 2 méthodes Rechercher, Ajouter et Modifier du contrôleur DedouanementControleur et la route à ajouter dans api.php pour accéder à la méthode Suivre_demande.

```
// web.php

use App\Http\Controllers\DedouanementControleur;

Route::get('/rechercher/{beneficiaireId}',
[DedouanementControleur::class, 'rechercher']);
Route::post('/ajouter', [DedouanementControleur::class, 'ajouter']);
Route::post('/modifier/{demandeId}', [DedouanementControleur::class,
'valider']);

// api.php

use App\Http\Controllers\DedouanementControleur;

Route::get('/suivre_demande/{idDemande}',
[DedouanementControleur::class, 'suivre_demande']);
```

6. Écrire le code de la vue Recherche.blade.php.

```
<!-- resources/views/Recherche.blade.php -->


<!DOCTYPE html>
```

```
<html>
<head>
    <title>Recherche des demandes</title>
</head>
<body>
    <h1>Liste des demandes en cours</h1>

    @if(session('success'))
        <div>
            {{ session('success') }}
        </div>
    @endif

    @if($demandes->isEmpty())
        <p>Aucune demande trouvée.</p>
    @else
        <table border="1">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Marque</th>
                    <th>Modèle</th>
                    <th>Année</th>
                    <th>Pays d'importation</th>
                    <th>Statut</th>
                    <th>Actions</th>
                </tr>
            </thead>
            <tbody>
```

```
                @foreach($demandes as $demande)
                    <tr>
                        <td>{{ $demande->id }}</td>
                        <td>{{ $demande->voiture->marque }}</td>
                        <td>{{ $demande->voiture->modele }}</td>
                        <td>{{ $demande->voiture->annee }}</td>
                        <td>{{ $demande->voiture->pays_importation
}}</td>
                        <td>{{ $demande->statut }}</td>
                        <td>
                            <form method="POST" action="/modifier/{{
$demande->id }}">
                                @csrf
                                <button
type="submit">Valider</button>
                            </form>
                        </td>
                    </tr>
                @endforeach
            </tbody>
        </table>
    @endif
</body>
</html>
```

Dossier 7: développer en front end: (25 PTS)

Dans ce dossier, on veut développer avec React la partie front end d'une application de gestion des taxes de dédouanement. L'équipe chargée de cette partie a décomposé la page souhaitée en composants React suivants.

- Un composant '**Calcul_taxe_dédouanement.js**'
- un composant '**Suivre_Demande.js**'
- un composant **Menu.js**

1. Écrire le composant **Menu.js** pour accéder aux autres composants. **(4PTS)**

Le composant '**Calcul_taxe_dédouanement.js**' est un simulateur qui permet aux utilisateurs de calculer les taxes de dédouanement d'un véhicule qui a une valeur inférieur à **400.000DH.**

Montant total des droits et taxes=**Droit d'importation**(17,5% valeur imposable)+**TVA** (**20.0%** valeur imposable) +**Taxe parafiscale** (**0.25%** valeur imposable)

La Valeur imposable (taxable) se calcule comme suit :

Age de la voiture	la valeur imposable (taxable)
moins d'un an	la valeur de la voiture à l'état neuf
1 an <= âge < 2 ans	90%de la valeur de la voiture à l'état neuf
2 ans <= âge < 3 ans	80% la valeur de la voiture à l'état neuf
Trois ans d'âge et plus :	75 % la valeur de la voiture à l'état neuf

Voici un exemple du Composant **Calcul_taxe_dédouanement.js** pour une voiture âgée entre un an et 2 ans et d'une valeur à neuf de 193800DH :

1. Écrire le composant `Menu.js` pour accéder aux autres composants.

```
import React from 'react';
import { BrowserRouter as Router, Route, Routes, Link } from 'react-
router-dom';
import CalculTaxeDedouanement from './Calcul_taxe_dédouanement';
import SuivreDemande from './Suivre_Demande';

function Menu() {
    return (
        <Router>
            <div>
                <nav>
                    <ul>
                        <li>
                            <Link to="/calcul-taxe-
dedouanement">Calcul Taxe Dédouanement</Link>
                        </li>
                        <li>
                            <Link to="/suivre-demande">Suivre
```

```
Demande</Link>
                        </li>
                    </ul>
                </nav>

                <Routes>
                    <Route path="/calcul-taxe-dedouanement"
element={<CalculTaxeDedouanement />} />
                    <Route path="/suivre-demande"
element={<SuivreDemande />} />
                    <Route path="/" element={<h2>Bienvenue au
système de gestion des taxes de dédouanement</h2>} />
                </Routes>
            </div>
        </Router>
    );
}
```

```
export default Menu;
```

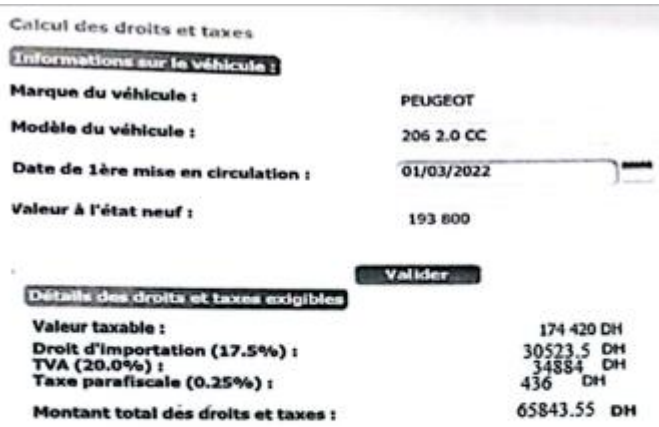


Figure1

On veut sauvegarder les calculs de la simulation dans un store REDUX.

L'état initial de ce store va suivre la structure suivante :

constinitState={ListeSimulation :[

{ marque : la marque de la voiture ,

date_mise_circulation : Date de mise en circulation de la voiture ,

valeur imposable : valeur imposable de la voiture,

montant : Montant total des droits et taxes calculé}] }

2. Créer le store Redux décrit ci-dessus avec son état initial et un reducer pour les deux actions suivantes : **(6PTS)**
   - **a. Ajouter :** ajouter une simulation de calcul dans le tableau « ListeSimulation »**(3pts)**
   - **b. supprimer :** qui supprime les calculs d'une marque passée en paramètre du tableau« ListeSimulation » .**(3pts)**
3. Écrire le code du composant **Calcul_taxe_dédouanement.js. :**
   a. le clic sur le bouton « **valider** » calcule et affiche la valeur imposable, Droit d'importation, TVA ,Taxe parafiscale et Montant total des droits et taxes. **(5PTS)**
   b. Le calcul doit être sauvegardé dans le store **(5PTS)**
4. Écrire le code du composant **'Suivre_Demande.*js*'** qui permet de suivre l'état d'une demande de dédouanement donnée. En utilisant **l'api *créé dans partie backend dont les informations suivantes*** **(5PTS)**

URL de l'api	http://127.0.0.1:8000/api/suivre/{iddemande}
Méthode HTTP	GET
Résultat	Résultat de la fonction **Suivre_demande** du backend


```
import { createStore } from 'redux';

const initialState = {
  ListeSimulation: []
};

function reducer(state = initialState, action) {
  switch(action.type) {
    case 'AJOUTER_SIMULATION':
      return {
        ...state,
        ListeSimulation: [...state.ListeSimulation, action.payload]
      };
```

```
    case 'SUPPRIMER_SIMULATION':
      return {
        ...state,
        ListeSimulation: state.ListeSimulation.filter(simulation =>
simulation.marque !== action.payload)
      };
    default:
      return state;
  }
}

const store = createStore(reducer);

export default store;
//action.js :

export const ajouterSimulation = (simulation) => ({
  type: 'AJOUTER_SIMULATION',
  payload: simulation
});

export const supprimerSimulation = (marque) => ({
  type: 'SUPPRIMER_SIMULATION',
  payload: marque
});
Calcul_taxe_dédouanement.js ;

import React, { useState } from 'react';
import { useDispatch } from 'react-redux';
import { ajouterSimulation } from './redux/actions';

function CalculTaxeDédouanement() {
  const [marque, setMarque] = useState('');
  const [modele, setModele] = useState('');
  const [dateMiseEnCirculation, setDateMiseEnCirculation] =
useState('');
  const [valeurEtatNeuf, setValeurEtatNeuf] = useState('');
  const [valeurImposable, setValeurImposable] = useState('');
  const [montantTotal, setMontantTotal] = useState('');

  const dispatch = useDispatch();

  const calculerTaxes = () => {
```

```
    const age = new Date().getFullYear() - new
Date(dateMiseEnCirculation).getFullYear();
    let taxableValue;

    if (age < 1) taxableValue = valeurEtatNeuf;
    else if (age >= 1 && age < 2) taxableValue = 0.9 *
valeurEtatNeuf;
    else if (age >= 2 && age < 3) taxableValue = 0.8 *
valeurEtatNeuf;
    else taxableValue = 0.75 * valeurEtatNeuf;

    const droitImportation = 0.175 * taxableValue;
    const tva = 0.2 * taxableValue;
    const taxeParafiscale = 0.0025 * taxableValue;
    const totalTaxes = droitImportation + tva + taxeParafiscale;

    setValeurImposable(taxableValue);
    setMontantTotal(totalTaxes);

    dispatch(ajouterSimulation({
      marque,
      dateMiseEnCirculation,
      valeurImposable: taxableValue,
      montant: totalTaxes
    }));
  };

  return (
    <div>
      <h2>Calcul des droits et taxes</h2>
      <div>
        <label>Marque du véhicule</label>
        <input type="text" value={marque} onChange={(e) =>
setMarque(e.target.value)} />
      </div>
      <div>
        <label>Modèle du véhicule</label>
        <input type="text" value={modele} onChange={(e) =>
setModele(e.target.value)} />
      </div>
      <div>
        <label>Date de mise en circulation</label>
```

```
        <input type="date" value={dateMiseEnCirculation}
onChange={(e) => setDateMiseEnCirculation(e.target.value)} />
      </div>
      <div>
        <label>Valeur à l'état neuf</label>
        <input type="number" value={valeurEtatNeuf} onChange={(e) =>
setValeurEtatNeuf(e.target.value)} />
      </div>
      <button onClick={calculerTaxes}>Calculer</button>
      <div>
        <p>Valeur taxable: {valeurImposable} DH</p>
        <p>Droit d'importation: {0.175 * valeurImposable} DH</p>
        <p>TVA: {0.2 * valeurImposable} DH</p>
        <p>Taxe parafiscale: {0.0025 * valeurImposable} DH</p>
        <p>Montant total des droits et taxes: {montantTotal} DH</p>
      </div>
    </div>
  );
}

export default CalculTaxeDédouanement;
```

Suivre_Demande.js:

```
import React, { useEffect, useState } from 'react';
import axios from 'axios';

function SuivreDemande() {
  const [demandes, setDemandes] = useState([]);

  useEffect(() => {
    axios.get('http://127.0.0.1:8000/api/suivre/liddemande')
      .then(response => setDemandes(response.data))
      .catch(error => console.error('Erreur lors du chargement des
données', error));
  }, []);

  return (
    <div>
      <h2>Suivre Demande</h2>
      <ul>
        {demandes.map((demande, index) => (
          <li key={index}>{demande}</li>
```

```
        ))}
      </ul>
    </div>
  );
}

export default SuivreDemande;
```