Filière :	**Développement des Systèmes d'Information (DSI)**	**Durée :**	4 heures
Épreuve :	**Développement des Applications Informatiques – DAI -**	**Coefficient :**	45

CONSIGNES

- ✓ **Le sujet comporte 4 dossiers ;**
- ✓ **chaque dossier doit être traité dans une feuille séparée.**

Barème de notation

Dossier 1 : Gestion des branchements électriques.	14 points
Dossier 2 :Consultation des demandes de branchement.	08 points
Dossier 3 : Gestion des plannings.	10 points
Dossier 4 : Suivi à distance des demandes.	08 points
Total	**40 points**

- ✓ **Il sera pris en considération la qualité de la rédaction lors de la correction.**
- ✓ **Aucun document n'est autorisé.**

ÉTUDE DE CAS : BRANCHEMENT DE L'EAU POTABLE ET D'ÉLECTRICITÉ

La société **MSEPE** (*Multi-Service pour le branchement de l'Eau Potable et d'Électricité*) offre des divers services parmi eux les branchements d'électricité et de l'eau potable au Maroc. La société sous-traite une partie de ses activités en délégant à des entreprises tierces la réalisation des branchements chez les clients. Dans cette étude on s'intéresse seulement aux branchements d'électricité.

Le traitement d'un branchement se déroule en plusieurs étapes :

- ✓ L'enregistrement de la demande de branchement d'un client et la validation de ses informations,
- ✓ L'élaboration du devis correspondant à la demande,
- ✓ La gestion des plannings, la communication des dates et lieux des rendez-vous aux sous-traitants,
- ✓ La réalisation des branchements par les sous-traitants,
- ✓ L'enquête de qualité afin de mesurer le degré de satisfaction des clients ainsi que la qualité du travail réalisé par les sous-traitants et par la société **MSEPE**.

Pour organiser les branchements, les villes sont découpées en **ZEI** (*Zones Élémentaires d'Intervention*). Une **ZEI** correspond à un secteur autour d'une **commune**.

DOSSIER 1 : GESTION DES BRANCHEMENTS ÉLECTRIQUES *(14 pts)*

Le diagramme de classes suivant présente une partie du système de gestion des branchements électriques (*figure 1*) :

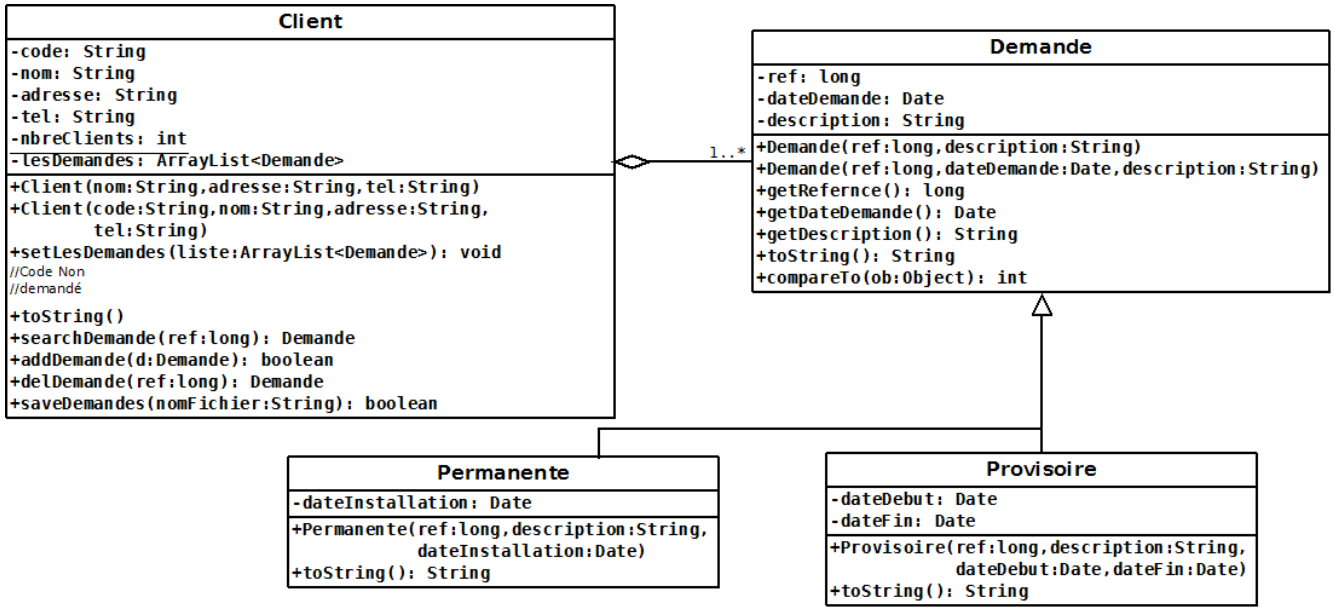

Figure 1 : Diagramme de classes **"Gestion des demandes de branchement"**

1. Créer la classe **« Demande »** qui implémente les deux interfaces « **Serializable** », « **Comparable** » ***et*** qui contient : *(0,5 pt)*
   - Deux constructeurs, le premier avec deux paramètres (*référence et description*), la date de la demande sera celle du système et le deuxième avec trois paramètres qui permet d'initialiser tous les attributs. *(1 pt)*
   - La méthode « **toString** » retourne une chaîne porteuse d'informations sur une demande. La chaîne aura la forme suivante : *(0,5 pt)*
     Référence : xxxx, Description : xxxx, Date de la demande : jj/mm/aaaa.
   - Les accesseurs pour tous les attributs de la classe. *(0,5 pt)*
   - La méthode « **compareTo »** compare deux demandes par leurs **références**. *(1 pt)*
2. Écrire le code de la classe d'exception « **ErreurDate »** qui permet de récupérer un message d'erreur. *(1 pt)*
3. Implémenter la classe **« Permanente»** qui contient : *(0,5 pt)*
   - Un constructeur, avec trois paramètres, qui génère l'exception « **ErreurDate »** si la date de la demande (*date du système*) est supérieure à la date d'installation. Le message d'erreur est : "*Erreur de date* ". *(1 pt)*
   - La méthode « **toString** » retourne une chaîne sous la forme suivante : *(1 pt)*
     Référence : xxxx, Description : xxxx, Date de la demande : jj/mm/aaaa, Date d'installation : jj/mm/aaaa.

   NB : Le code de la classe Provisoire est déjà implémenté.

4. Implémenter la classe **« Client »** qui contient : *(0,5 pt)*

- Un constructeur avec trois paramètres permettant d'initialiser : *(1 pt)*
  - Le nom, l'adresse et le téléphone.
  - L'attribut de classe « **nbreClients »** compte le nombre de clients instanciés.
  - Le « **Code »** par le nom du client suivi du numéro de l'ordre de création de ce client.
  - La collection « **lesDemandes »** de type « **ArrayList ».**

  NB : on suppose que le constructeur de quatre paramètres est déjà implémenté.

- La méthode « **toString** » permet de retourner une chaîne sous la forme suivante : *(1 pt)*

```
Code:xxxx, Nom : xxxx, Adresse : xxxx, Tel : xxxx
Les demandes :
Référence : xxxx, Date de la demande : jj/mm/aaaa, Date d'installation : jj/mm/aaaa
        .....        .....  .....  .....        .....
Référence : xxxx, Date de la demande : jj/mm/aaaa, Date de début : jj/mm/aaaa, Date de fin : jj/mm/aaaa
        .....        .....  .....  .....        .....
```

- La méthode « **addDemande(Demande d) »** qui ajoute une nouvelle demande, donnée en paramètre, à la collection et retourne l'état de l'opération. *(1 pt)*
- La méthode « **searchDemande(long ref) »** qui recherche et retourne une demande en se basant sur sa référence. Si cette demande n'existe pas la méthode retourne « **null ».** *(1 pt)*
- La méthode « **delDemande(long ref) »** qui supprime une demande de la collection en se basant sur sa référence et retourne la demande supprimée. *(1 pt)*
- La méthode « **saveDemandes(String nomFichier) »** qui sauvegarde, dans un fichier d'objets, les demandes permanentes de ce client. *(1,5 pt)*

DOSSIER2 : CONSULTATION DES DEMANDES DE BRANCHEMENT *(8 pts)*

La mise en œuvre d'une nouvelle application est envisagée pour permettre aux sous-traitants de consulter les demandes de branchement des clients et d'enregistrer leurs indisponibilités. Cette application sera hébergée dans un serveur *d'application* qui exploite les objets distribués via l'utilisation de la **RMI** (***RemoteMethod Invocation***).

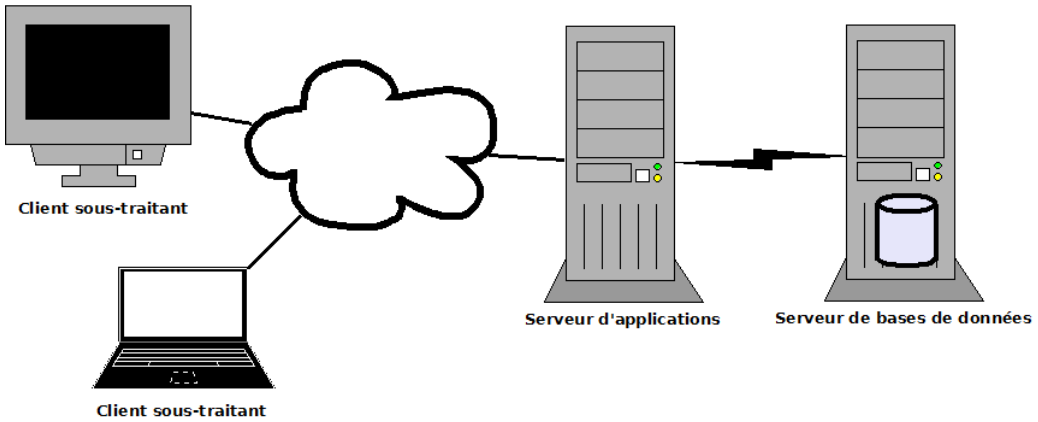

Figure 2 : Architecture client/serveur

Le serveur d'application utilise l'API RMI intégrant une classe représentant des méthodes à partager. Ces méthodes permettent d'accéder à la base de données **« BD_Branchement »** implantée dans un autre serveur de Gestion de Base de Données. Les postes des sous-traitants (clients) exploitent les méthodes offertes par le serveur d'application (*figure 2*).

1. Préciser le type d'architecture client/serveur adopté par ce système et donner le modèle **GartnerGroup** correspondant. *(1 pt)*

Voici un extrait du modèle relationnel de la base de données **"BD_Branchement"** (*figure 3*):

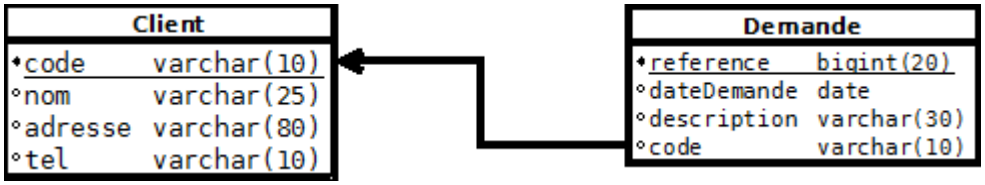

Figure 3 : MLD **"BD_Branchement"**

Soit la classe suivante qui représente l'objet distant :

```
public class ServeurDistant extends Unicast RemoteObject implements IntServiceBranchement{
      private ArrayList<Client> liste;
      private Connection cnx;

      public ServeurDistant() throws RemoteException {
            . . . . . // Code à compléter.
      }

      public ArrayList<Demande> lireDemandes(String code) throws RemoteException {
            . . . . . // Code à compléter.
      }

      public void lireClients() throws RemoteException {
            . . . . . // Code à compléter.
      }
}
```

2. Donner le code de l'interface « **IntServiceBranchement**» implémentée par la classe « **ServeurDistant »** afin d'assurer le partage des méthodes de cette classe. *(1 pt)*

3. Compléter la classe « **ServeurDistant** » par l'implémentation des méthodes suivantes :
   - Un constructeur permettant d'instancier l'attribut « **liste »** des clients et d'établir une connexion à la base de données MYSQL. *(1 pt)*
     - ✓ **URL:**`jdbc:mysql://Administrateur:3306/bd_branchement`
     - ✓ **User** :*root*
     - ✓ ***Password*** *:*`DSI2018.`

     En cas d'échec de connexion, afficher un message d'erreur.
   - La méthode « **lireDemandes** » permet de retourner une collection des demandes dont le code client est donné en paramètre. *(1 pt)*
   - La méthode « **lireClients** » permet de stocker tous les clients de la table « **Client** » de la base de données dans l'attribut **« liste »**. Pour chaque client de cette collection, on doit récupérer ses demandes en les sauvegardant dans l'attribut « **lesDemandes** » de l'objet client. *(1 pt)*

4. Écrire le code du ***serveur RMI*** permettant de démarrer l'annuaire ***rmiregistry***, de créer l'objet partagé de la classe « **ServeurDistant** » et de publier la référence de cet objet dans l'annuaire sous le nom «**ob**». *(1,5 pt)*

5. Écrire le code du ***client RMI*** permettant de : *(1,5 pt)*

- Récupérer, de l'annuaire, la référence de l'objet distant,
- Lire et afficher la liste des demandes d'un client dont le code est : « aziz2018 ».

DOSSIER 3 : GESTION DES PLANNINGS *(10 pts)*

Pour effectuer les installations électriques, la société **MSEPE** organise le planning des interventions des sous-traitants. Chaque contrat de sous-traitance couvre un certain nombre de **ZEI** (*Zones Élémentaires d'Intervention*) et indique les jours d'intervention possibles.

Son service informatique est en train de développer, *sous VB.Net*, une application de gestion du planning utilisant une base de données centralisée nommée **« DB_Planification »**. Cette dernière est implémentée sous un serveur MS SQL Server nommée « **Srv-MSEPE** », dont voici un extrait de son modèle relationnel.

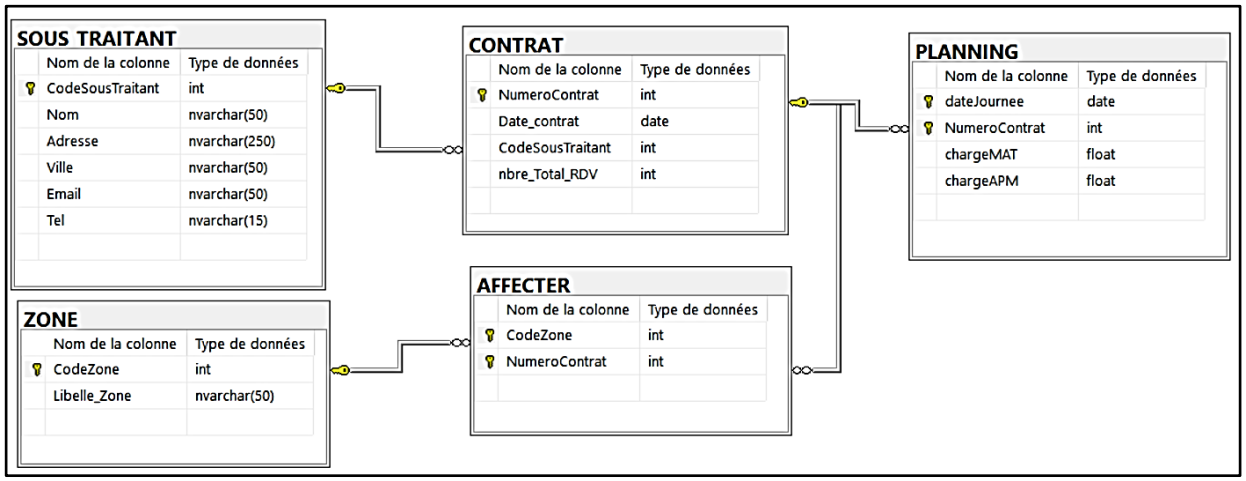

Figure 4 : Extrait du modèle relationnel de la base de données **« DB_Planification »**

- **Nbre_Total_RDV :** nombre total de rendez-vous pour l'année en cours.
- **chargeMAT :** correspond à la charge de travail affectée le matin ; elle est initialisée à **zéro** au moment de la création et ne peut pas dépasser 240 minutes (*4 heures*).
- **chargeAPM :** correspond à la charge de travail affectée l'après-midi ; elle est initialisée à zéro au moment de la création et ne peut pas dépasser 240 minutes (*4 heures*).
- La table **« AFFECTER »** possède une clé primaire composée des clés étrangères « **CodeZone** » et « **NumeroContrat** ».

1. Déclarer les objets de connexion à la base de données. *(0,5 pt)*

2. Écrire le code de la fonction « **F_Connexion()** » qui retourne « **True** » si la connexion au serveur de base de données est établie avec succès ou **« False »** dans le cas échéant. Gérer les exceptions. *(1 pt)*

Signature de la fonction

`Public Function F_Connexion() As boolean`
`. . . . . .`
`End Function`

3. Écrire le code de la procédure « **Ps_Affecter()** » qui prend deux arguments (*Code de la zone* et *le numéro de contrat*) puis enregistre les données dans la table « **AFFECTER** ». La procédure doit : *(3 pts)*

- vérifier l'existence du numéro de contrat dans la table « **CONTRAT** » ;
- vérifier l'existence du code zone dans la table « **ZONE** »;
- vérifier l'unicité de la clé composée (**Codezone** et **NumeroContrat**) dans la table « **AFFECTER** ».
- insérer le nouveau enregistrement dans la table « **AFFECTER** ».

Signature de la procédure

```
Public Sub Ps_Affecter(ByVal codeZone As Integer, ByVal NumeroContrat As Integer)
            . . . . .. . .. . . . . . . . .
End Sub
```

4. Écrire le code de la procédure « **Supp_Contrat()** » qui prend en argument le numéro du contrat et qui permet de supprimer : *(1,5 pt)*

- les affectations de ce contrat ;
- les plannings de ce contrat ;
- le contrat lui-même.

Signature de la procédure

```
Public Sub Supp_Contrat(ByVal NumeroContrat As Integer)
            . . . . . . . . . . . . . . .
End Sub
```

5. Pour lister les interventions prévues ou planifiées pour un contrat donné, le concepteur a mis en place le formulaire suivant :

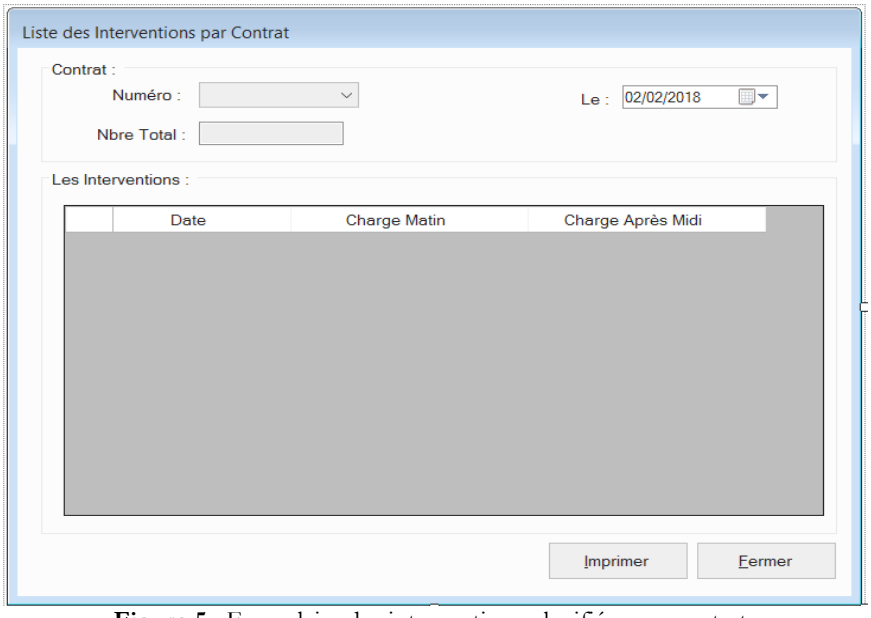

Figure 5 : Formulaire des interventions planifiées par contrat

a. Écrire le code permettant de remplir le **Combobox** nommé « **CmbContrat** » par les numéros des contrats. *(1 pt)*

b. Écrire le code de la procédure « **Lister_Detail()** » qui liste, dans l'objet **DataGridView** nommé « **DGListe** », les plannings d'un contrat dont les champs sont illustrés dans la figure 5. *(2 pts)*

Signature de la procédure

```
Public Sub Lister_Detail(ByVal NumeroContrat As Integer)
      . . . . . . . . . .. . . . .
End Sub
```

c. Donner le code de la procédure événementielle du comboBox « **CmbContrat** » qui permet, pour le contrat sélectionné, de : *(1 pt)*

- afficher dans la zone texte « **txtNbreTotal »** le nombre total des rendez-vous ;
- appeler la procédure « **Lister_Detail()** » pour afficher les planifications du même contrat.

Signature de la procédure

```
Public Sub CmbContrat_SelectedIndexChanged (ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles CmbContrat.SelectedIndexChanged
      . . . . . .
End Sub
```

DOSSIER 4 : SUIVI À DISTANCE DES DEMANDES *(8 pts)*

Le service d'abonnement de la société **MSEPE** souhaite améliorer la gestion des demandes de branchement électriques des clients. Pour ce faire, il propose de mettre en place un site web dynamique pour le suivi à distance de ces demandes. Dans ce contexte, un extrait de la base de données, sous MySQL, nommée «**BD_Demandes**» contient les tables suivantes :

DEMANDE(**refd**, description, dated, **#codec**, **#refop**)
CLIENT(**codec**, civilite, nom, prenom, adresse, ville, mail, tele)
OPERATION(**refop**, detail, type)

CHAMP	SIGNIFICATION	TYPE
refd	Référence de la demande	varchar(25)
description	Description de la demande	varchar(200)
dated	Date de la demande	Date
codec	Code client	varchar(25)
civilite	Civilité (Mr : 'M', Mme : 'E' , Mlle : 'L')	varchar(25)
tel	Téléphone	varchar(25)
refop	Référence de l'opération	varchar(25)
detail	Détail de l'opération	varchar(250)
type	Type de la demande (nouvelle :'N', renouvellement :'R', modification :'M')	char(1)

Les paramètres du serveur MYSQL sont :

- ✓ **Nom de serveur :** DB_SERVER.
- ✓ **User :***root*
- ✓ ***Password :***DSI2018.

1. Pour accéder aux différentes pages du site web, le client doit saisir son code dans le formulaire de la page **«index.php»** ou de créer une nouvelle inscription. Le champ « **code client** » est validé par la fonction « **valide** ».

Figure 6 : Page « index.php »

Le code de la page «**index.php**» est le suivant:

```
<html>
<head><meta charset="UTF-8">
<title>Gestion des demandes de branchements électriques</title>
<link href="styles.css" rel="stylesheet">
<script language="JavaScript" >
1.a
        function valide(){
. . . . . .
                    }
</script>
</head>
<body><center>
<fieldset style="width:400px">
<legend><b>Saisissez votre code client </b></legend>
<form action= "rech.php" method="post" name="fcon">
<table border="0" align="center" style="width:400px">
<tr><td colspan="2" align="right"><i><u>
<a href="inscription.php">Nouvelle inscription</a></u></i></td></tr>
<tr><td>Code client : </td>
<td>1.b . . . . . </td></tr>
<tr><td colspan="2" align="center">1.c . . . . . </td></tr>
</table></form></fieldset></center>
</body></html>
```

Compléter le code des parties : **1.a**, **1.b** et **1.c**. *(2 pts)*

2. Le formulaire d'inscription d'un nouveau est illustré par la figure suivante :

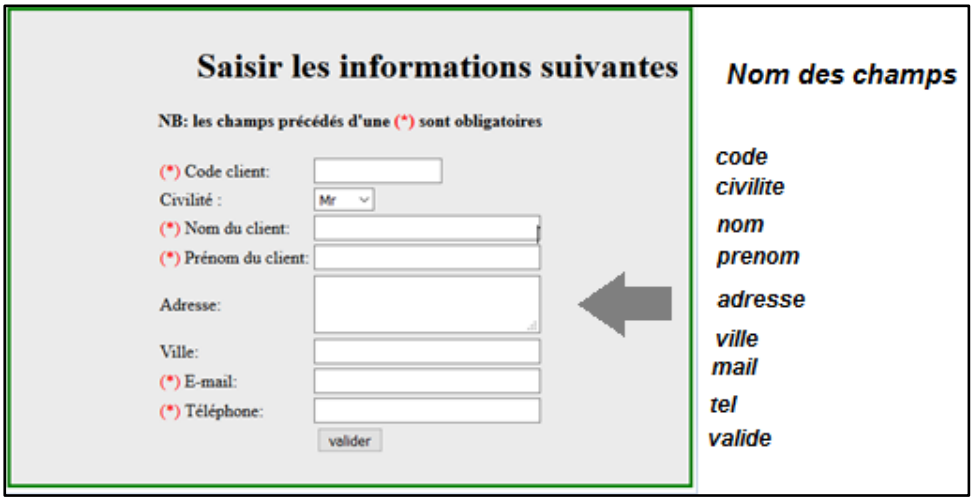

Figure 7 : Page « inscription.php»

Le canevas du formulaire ci-dessus est le suivant :

```
<form action='inscriptionPost.php' method='POST' name="fvald">
. .. . . . . . . . . . . . .
</form></div></center></body></html>
```

Écrire le script PHP de la page « **inscriptionPost.php** » permettant de : *(3 pts)*

- Vérifier l'unicité de code client ;
- Enregistrer toutes les informations du formulaire dans la table « **Client** » si ce code est unique. Dans le cas contraire, afficher un message d'erreur dans une boite de dialogue.

NB : on suppose que les champs obligatoires du formulaire d'inscription sont bien remplis.

3. Le clic sur le bouton « **soumettre** » de la page « **index.php** » ouvre la page «**rech.php**» (*figure 8*). Cette page démarre une session puis enregistre les variables de la session (**code_client, nom, prenom, civilité**) dans le cas où le code existe dans la table « **Client** » et affiche le contenu de la page (*figure 8*). Dans le cas contraire, elle redirige vers la page « **index.php** ».

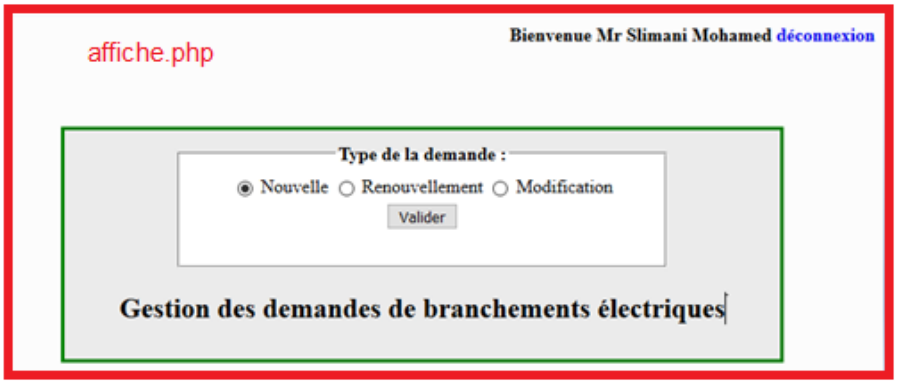

Figure 8 : Page « rech.php »

Le canevas de la page « **rech.php** » est le suivant :

```
<html>
      <head><meta charset="UTF-8">
         <title>Gestion des demandes de branchement
           électriques</title>
         <link href="styles.css" rel="stylesheet">
      </head>
      <body>
         <?php
          . . . . . .
          . . . . . .
          ?>
      </body>
</html>
```

Écrire le code PHP qui permet de : *(3 pts)*

- Récupérer le code client de la page «**index.php**».
- Vérifier l'existence d'un client ayant le code récupéré précédemment.
- Mémoriser, si le code existe, dans une session les champs **: code client**, **nom**, **prénom** et **civilité,** puis inclure le fichier « **affiche.php** » déjà existant.
- Rediriger, dans le cas contraire, vers la page «**index.php**».