Application dirigée avec gestion d'une base de données.

Le but de ce mode opératoire consiste à développer, étape par étape, une application Web avec une gestion d'une base de données contenant un nombre limité de tables. Chaque table aura une structure réduite et l'aspect graphique de l'interface sera minimaliste, afin de simplifier le développement et de gagner en rapidité. L'application finale sera sans doute perfectible et chacun pourra y apporter des améliorations. Mais le but premier de ce mode opératoire est de développer cette application en suivant une méthodologie utilisable pour les futurs développements Web.

Sujet : gestion simplifiée d'un inventaire

L'application doit gérer l'inventaire des équipements des salles dans un établissement universitaire.

La page d'accueil de l'application doit

- afficher la liste des salles (n° salle, désignation salle, étage, nombre d'équipements avec un lien sur l'affichage des équipements ou lien « inventaire » si aucun équipement)
- à partir de la liste des salles, modifier, supprimer la salle sélectionnée ou encore ajouter une nouvelle salle

L'édition (ajout, modification) d'une salle affiche les informations : numéro, désignation, étage

La liste des équipements d'une salle doit

- afficher la liste des équipements (n° équipement, désignation équipement, quantité dans la salle)
- afficher des liens pour
  - modifier la quantité d'un équipement,
  - supprimer l'équipement,
  - ajouter un nouveau type d'équipement dans la salle
  - retourner à l'accueil

L'édition (ajout, modification) d'un équipement affiche les informations : désignation, quantité

Maquettage- proposition

Entête présent sur chaque page :

Gestion de l'inventaire des salles

Liste des salles :

Liste des salles

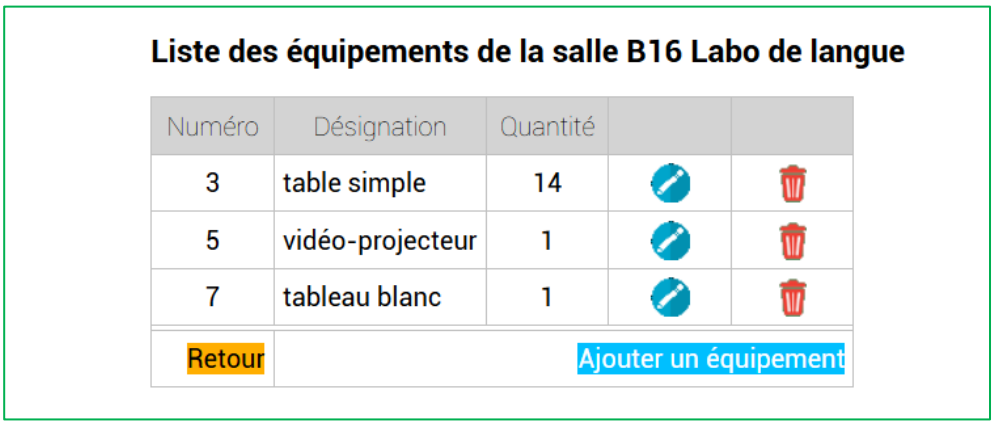
Numéro	Désignation	Etage	Equipements		
B16	Labo de langue	1	16		
F13	machine PC	1	16		
E23		2	inventaire		
C14		1	33		
E36		3	2		
F39	machine PC	3	62		

Ajouter une salle

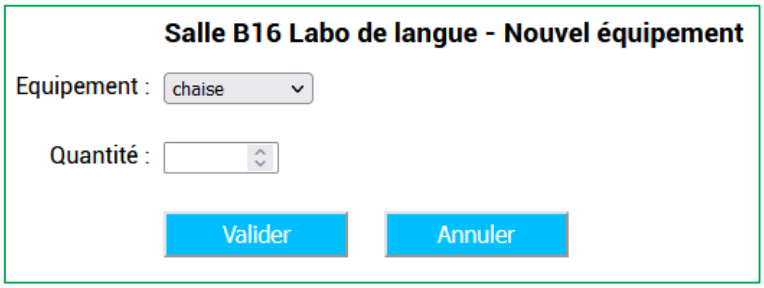
Ajout d'une salle :

Edition d'une salle :

Liste des équipements d'une salle



Ajout d'un équipement dans une salle :



Edition d'une ligne d'inventaire d'une salle :

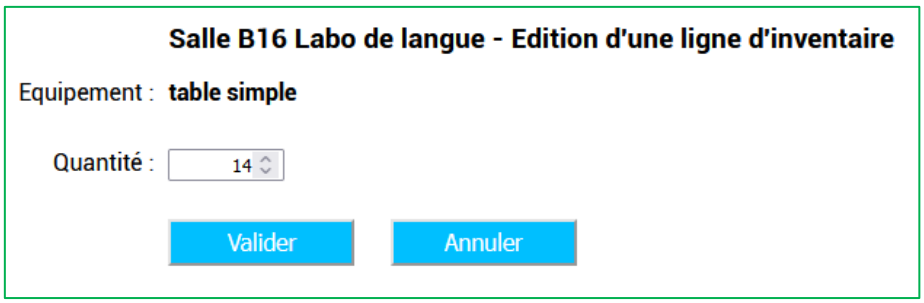


Schéma relationnel de la base

SALLE	(num_salle,	lib_salle,	etage)
TYPE_EQUIPT	(id_equipt,	lib_equipt,	commentaire)
CONTIENT	(num_salle,	id_equipt,	qte)

La table **SALLE** est la table dite « **maître** » de l'application. C'est à partir de cette table que l'inventaire, la liste des équipement, d'une salle est accessible. La table **TYPE_EQUIPT** est la table dite « **détail** ».
La **relation** « **maître – détail** » n'est pas directe. Il faut passer par la table **CONTIENT** qui fait le lien entre les tables SALLE et TYPE_EQUIPT.

Etape 1 : créer la structure de votre projet

Créer un **dossier** pour votre application, « **inventaire** » pour notre application
Dans ce dossier, créer 3 sous-dossiers :

- **modele** : ce dossier contiendra les fichiers de gestion des données de la base de données
- **vue** : ce dossier contiendra les fichiers PHP avec la définition des composants HTML dont les valeurs seront généralement alimentées par du code PHP
- **controleur** : ce dossier contiendra les fichiers servant à faire le lien entre les fichiers de « modele » et les fichiers de « vue »

Dans le dossier « vue », créer un sous-dossier « **style** » qui contiendra la feuille de style et les images.

A la racine du dossier « inventaire », créer un fichier « **index.php** »

```php
<?php
header("location: ./controleur/salles.php");
?>
```

L'accueil de l'application est le fichier « salles » affichage de la liste des salles.

Etape 2 : définition du modèle de données dans le dossier « **modele** »

Créer le fichier « **connexion.php** » : définition de la **classe Connexion**

1. définir l'**attribut** privé $db
2. écrire le **constructeur** de connexion à la base (créer un tableau $db_config associatif contenant les informations de connexion (SGBD, HOST, DB_NAME, USER, PASSWORD)

   Mots clés :
   - new PDO
   - PDO ::MYSQL_ATTR_INIT_COMMAND=> 'SET NAMES utf8'
   - unset
   - try … catch
   - die

3. écrire la fonction **execSQL(string $req, array $valeurs=[])** qui retourne le résultat d'une requête sous forme d'un tableau.

   Paramètres de la fonction
   - $req contient la requête à préparer
   - $valeurs contient les paramètres de la requête

   Mots clés :
   - prepare
   - execute
   - fetchall
   - PDO ::FETCH_ASSOC
   - try … catch
   - die

Fichiers PHP de gestion des tables

Pour chaque table, on va créer
- une classe pour la gestion des attributs
- une classe d'accès aux données (**DAO** : Data Access Object)

Créer le fichier « **salle.class.php** » : définition de la **classe Salle** (table « maître » SALLE)

1. définir les **attributs** privés : un attribut par champ de la table
2. écrire le **constructeur** qui initialise les attributs avec les paramètres en entrée initialisés à chaîne vide par défaut :

```
function __construct(string $num='', string $libelle='', string $etage='')
```

3. écrire les **getters** et **setters** pour chaque attribut

Créer le fichier « **salleDAO.class.php** » : définition de la **classe SalleDAO**

1. importer les fichiers « connexion.php » et « salle.class.php »
2. définir les **attributs** privés : **$bd** et **$select**
3. écrire le **constructeur** :
   - instancier l'attribut *$bd*
   - affecter dans *$select*, la requête de sélection des champs qui correspondent aux attributs de la classe *Salle*

4. recopier la fonction **insert (Salle $salle)** : requête d'insertion d'une salle *$salle* dans la table

```
function insert (Salle $salle) : void {
   $this->bd->execSQL("INSERT INTO SALLE (num_salle, lib_salle, etage)
                       VALUES (:num, :libelle, :etage)"
                       ,[':num'=>$salle->getNum(), ':libelle'=>$salle->getLibelle()
                       ,':etage'=>$salle->getEtage() ] );
}
```

5. écrire la fonction **delete (string $num)** : requête de suppression de la salle *$num* dans la table
6. écrire la fonction **update (Salle $salle)** : requête de mise à jour de la salle *$salle* dans la table
7. recopier la fonction **loadQuery (array $result)** : affectation résultat d'une requête de sélection dans un tableau d'objets de classe *Salle*

```
private function loadQuery (array $result) : array  {
   $salles = [];
   foreach($result as $row) {
      $salle = new Salle();
      $salle->setNum($row['num_salle']);
      $salle->setLibelle($row['lib_salle']);
      $salle->setEtage($row['etage']);
      $salles[] = $salle;
   }
   return $salles;
}
```

8. recopier la fonction **getAll ( )** : retourne le tableau de toutes les salles (tableau d'objets instances de la classe *Salle*)

```
function getAll () : array  {
   return  ($this->loadQuery($this->bd->execSQL($this->select)));
}
```

9. recopier la fonction **getByNum (string $num)** : retourne l'objet de classe *Salle* de la salle *$num*

```
function getByNum (string $num) : Salle {
   $uneSalle = new Salle();
   $lesSalles = $this->loadQuery($this->bd->execSQL($this->select ." WHERE
num_salle=:num", [':num'=>$num]) );
   if (count($lesSalles)>0) { $uneSalle = $lesSalles[0]; }
   // il y a un seul élément dans le tableau de salles ➔ indice 0
   return $uneSalle;
}
```

10. recopier la fonction **existe (string $num)** : retourne vrai si la salle *$num* existe dans la table

```
function existe (string $num) : bool {
   $req    = "SELECT *  FROM  SALLE WHERE num_salle = :num";
   $res    = ($this->loadQuery($this->bd->execSQL($req,[':num'=>$num])));
   return ($res != []);   // si tableau de salles est vide alors la salle n'existe pas
}
```

11. recopier la fonction **getTotalNbEquipt (string $numSalle)** : retourne la quantité totale d'équipements d'une salle ; cette fonction sera appelée dans la liste des salles, colonne « nb équipements »

```
function getTotalNbEquipt(string $numSalle) : int {
// renvoie la quantité totale d'équipements d'une salle si salle présente dans CONTIENT
   $res = $this->bd->execSQL("SELECT SUM(qte) as total FROM CONTIENT
                              WHERE num_salle=:numSalle", [':numSalle'=>$numSalle]);
   return (isset($res[0]['total']))?$res[0]['total']:0;
}
```

Créer le fichier « **equipement.class.php** » : définition de la **classe Equipement (**table « détail » TYPE_EQUIPT)

1. définir les **attributs** privés : un attribut par champ de la table
2. écrire le **constructeur** qui initialise les attributs avec les paramètres en entrée initialisés à chaîne vide par défaut
3. écrire les **getters** et **setters** pour chaque attribut

Créer le fichier « **equipementDAO.class.php** » : définition de la **classe EquipementDAO**

Même si dans l'application, la gestion de la table TYPE_EQUIPT n'est pas demandée, elle peut être envisagée par la suite. Il est donc conseillé de gérer cette table en implémentant les requêtes d'ajout, de modification et de suppression.

1. importer les fichiers « connexion.php » et « equipement.class.php »
2. définir les **attributs** privés : **$bd** et **$select**
3. écrire le **constructeur** :
   - instancier l'attribut $bd
   - affecter dans *$select*, la requête de sélection des champs qui correspondent aux attributs de la classe *Equipement*
4. écrire la fonction **insert (Equipement $equipement)** : requête d'insertion d'un type d'équipement *$equipement* dans la table
5. écrire la fonction **delete (string $idEquipt)** : requête de suppression du type d'équipement *$idEquipt* dans la table
6. écrire la fonction **update (Equipement $equipement)** : requête de mise à jour du type d'équipement *$equipement* dans la table
7. écrire la fonction **loadQuery (array $result)** : affectation résultat d'une requête de sélection
8. écrire la fonction **getAll ( )** : retourne le tableau de tous les types d'équipement (tableau d'objets instances de la classe *Equipement*)
9. écrire la fonction **getById (string $id)** : retourne l'objet de classe Equipement du type d'équipement *$id*
10. écrire la fonction **existe (string $id)** : retourne vrai si le type d'équipement *$id* existe dans la table
11. écrire la fonction **getNonInventaire (string $numSalle)** : retourne le tableau des types d'équipement non présents dans la salle *$numSalle*. Fonction utile pour le choix d'un type d'équipement à ajouter dans une salle.

Créer le fichier « **equiptBySalle.class.php** » : définition de la **classe EquiptBySalle (**table de liaison CONTIENT entre SALLE et TYPE_EQUIPT)

1. importer le fichier « equipement.class.php »
   qui contient la définition des attributs de la table « détail » TYPE_EQUIPT
2. définir les **attributs** privés : la règle « un attribut par champ » reste valable sauf pour le champ *id_equipt* qui sera représenté par l'attribut privé *$equipement* (instance de la classe Equipement)
3. écrire le **constructeur** qui initialise les attributs avec les paramètres en entrée
   ```
   function __construct(string $numSalle='', Equipement $equipement=null, int $qte=0)
   ```
4. écrire les **getters** et **setters** pour chaque attribut

Créer le fichier « **equiptBySalleDAO.class.php** » : définition de la **classe EquiptBySalleDAO**

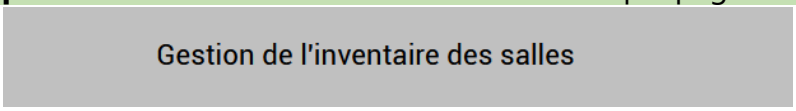
1. importer les fichiers « connexion.php », « equiptBySalle.class.php » et « equipementDAO.class.php »
2. définir les **attributs** privés : **$bd** et **$select**
3. écrire le **constructeur** :
   - instancier l'attribut $bd
   - affecter dans *$select*, la requête de sélection des champs qui correspondent aux attributs de la classe *EquiptBySalle* (attention : la classe contient l'attribut *equipement* ,instance de la classe *Equipement*)
4. écrire la fonction **insert (EquiptBySalle $equiptBySalle)** : requête d'insertion d'un équipement *$equiptBySalle* dans la table CONTIENT
5. écrire la fonction **deleteByNumSalleByIdEquipt (string $numSalle, string $idEquipt)** : requête de suppression de l'équipement *$idEquipt* de la salle *$numSalle* dans la table CONTIENT
6. écrire la fonction **deleteByNumSalle (string $numSalle)** : requête de suppression des équipements de la salle *$numSalle* dans la table CONTIENT
7. écrire la fonction **update (EquiptBySalle $equiptBySalle)** : requête de mise à jour de l'équipement *$equiptBySalle* dans la table CONTIENT
8. écrire la fonction **loadQuery (array $result)** : affectation résultat d'une requête de sélection telle que

```php
$equipementDAO = new EquipementDAO();
$lesEquiptBySalle = [];
foreach($result as $row)  {
   $equipement = $equipementDAO->getById($row['id_equipt']);
   $lesEquiptBySalle[]= new EquiptBySalle($row['num_salle'], $equipement, $row['qte']);
}
return $lesEquiptBySalle;
```

9. écrire la fonction **getAll ( )** : retourne le tableau de tous les équipements (tableau d'objets instances de la classe *EquiptBySalle*) de toutes les salles
10. écrire la fonction **getByNumSalle (string $numSalle)** : retourne le tableau des équipements (tableau d'objets instances de la classe *EquiptBySalle*) de la salle *$*numSalle
11. écrire la fonction **getByNumSalleByIdEquipt (string $numSalle, string $idequipt)** : retourne l'objet de classe EquiptBySalle de l'équipement *$idEquipt* de la salle *$numSalle*
12. écrire la fonction **existe (string $numSalle, string $idEquipt)** : retourne vrai si l'équipement *$idEquipt* est présent dans la salle *$numSalle*

Etape 3 : définition de l'entête de chaque page et du fichier style

Créer le fichier « **vue/header.php** » : définition de l'entête à afficher sur chaque page



1. dans une balise **<header>**, insérer une balise **<section>** avec deux colonnes respectivement une balise **** et une balise **<h1>**
2. dans la balise **<h1>**, écrire le titre

Recopier le fichier « **vue/css/style.css** » dans le **sous-dossier** « **style** »

```
body{
   font-family : Roboto,Arial, sans-serif;
   width       : auto;
}
h1{
   font-size   : 1.2em;
}
header {
   background-color : silver;
}
header h1 {
   font-weight : normal;
   line-height : 2em;
   font-size   : 1.4em;
}
a {
   text-decoration : none; // suppression du soulignement pour la balise a
   cursor      : pointer;
}
section {
   min-height  : 3em;
   text-align  : left;
   display     : grid;      // intérieur de la section divisée en colonnes
   grid-template-columns : 1fr 8fr;  // la seconde colonne est 8 fois plus large que la 1ère
   grid-gap    : 10px;      // espace entre les colonnes
}
.erreur {
   font-size   : small;
   color       : red;
}
label {
   text-align  : right;
}
input[type="submit"], .ajout {
   min-height  : 2em;
   min-width   : 8em;
   background-color : rgba(0, 191, 255);
   font-size   : medium;
   cursor      : pointer;
   color       : white;
}
```

```
input[type="number"] {
   text-align  : right;
   width       : 6em;
}
input[name="Valider"], .info {
   background-color : rgba(0, 255, 34);
   color       : black;
}
input[name="Annuler"], .retour {
   background-color : rgba(255, 174, 0);
   color       : black;
}
.supprimer {
   background-color : red;
   color       : white;
}
.modifier {
   background-color : rgba(0, 85, 255);
   color       : white;
}

table {
   margin      : auto;
   margin-left : 0px;
}

table, td, th{
   border      : 1px solid silver;
   border-collapse : collapse;
   line-height : 2em;
}
td, th{
   padding-left : 5px;
   padding-right: 5px;
}
th{
   background-color : lightgrey;
   font-weight : lighter;
}
td  {
   min-width : 4em;
  text-align : center;
}

.table_salle td:nth-child(2) {
   text-align : left;
}

.table_salle_equipt td:nth-child(2) {
   text-align : left;
}
```

Etape 4 : définition des pages du site

Chaque page du site sera construite avec deux fichiers :

- un contenant la structure de la page dans le dossier « vue »
- un autre contenant les instructions pour alimenter la page dans le dossier « controleur »

Créer le fichier « **vue/salle.view.php** » : liste des salles

1. écrire le contenu de la balise **<head>** contenant
   - la mise en encodage utf-8, balise **<meta>**
   - le titre de la page « Liste des salles »
   - l'importation du fichier des styles
2. écrire le contenu de la balise **<body>** contenant
   - l'importation de l'entête
   - une première section contenant
     - ✓ un label vide
     - ✓ le titre de la page
   - une seconde section contenant
     - ✓ un label vide
     - ✓ le tableau « table_salle » avec une ligne de titre,
       , des lignes de données (séquence PHP) :

```
<?php
    foreach($lignes as $ligne) {
       echo $ligne;  // tableau de lignes à créer dans /controleur/salles.php
    }
?>
```

       , une ligne vide
       , une ligne avec le lien qui appelle le formulaire d'édition d'une salle en mode ajout

```
<a href="editSalle.php?op=a" class="ajout">Ajouter une salle</a>
```

Liste des salles

Numéro	Désignation	Etage	Equipements		
B16	Labo de langue	1	16		
F13	machine PC	1	16		
E23		2	inventaire		
C14		1	33		
E36		3	2		
F39	machine PC	3	62		

Ajouter une salle

Rappel :
dans la feuille de style, le contenu de la balise <section> est affiché sur 2 colonnes :

```
display               : grid;
grid-template-columns : 1fr 8fr;
```

Créer le fichier « **controleur/salle.php** »

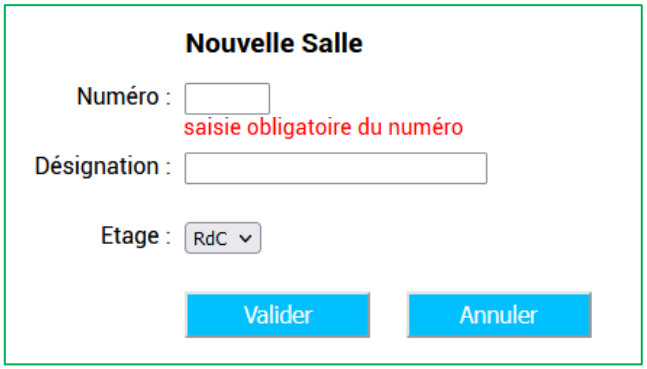
- importer le fichier contenant la classe « SalleDAO »
- créer une instance de la classe « SalleDAO »
- affecter la méthode contenant la requête qui liste toutes les salles dans la variable *$lesSalles*
- déclarer le tableau de lignes *$lignes*
- parcourir *$lesSalles* et construire chaque ligne du tableau de lignes en complétant (*partie en italique*) le code ci-après :

```
foreach($lesSalles as $uneSalle)
{
   $ch= '';
   $ch .='<td>' . à compléter . '</td>';
   $ch .='<td>' . à compléter . '</td>';
   $ch .='<td>' . à compléter. '</td>';
   $nb = à compléter nombre total d'équipement de la salle;
   if ($nb==0) $ch .='<td><a href="editSalleEquipt.php?op=a&num='
                  .urlencode(à compléter)  .'" class="info" >inventaire</a></td>';
   else $ch .='<td><a href="salleEquipt.php?num='
                  .urlencode(à compléter)  .'" class="info" >' .$nb .'</a></td>';
   $ch .='<td><a href="editSalle.php?op=m&num=' .urlencode(à compléter)
       .'" class="modifier">Modifier</a></td>';
   $ch .=  '<td><a href="editSalle?op=s&num=' .urlencode(à compléter) .'" >
       <img src="../vue/style/corbeille.png"></a></td>';

   $lignes[] = "<tr>$ch</tr>";
}
```

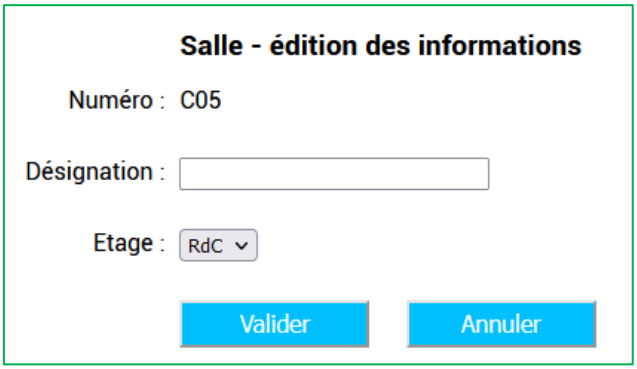
- désallouer *$lesSalles* avec la fonction *unset*
- importer le fichier de « salles.view.php »

Créer le fichier « **vue/editSalle.view.php** » : liste des salles

Ajout d'une salle :



Edition d'une salle (modification) :



1. écrire le contenu de la balise **<head>** contenant
   - la mise en encodage utf-8, balise **<meta>**
   - le titre de la page avec la variable *$titre*
   - l'importation du fichier des styles
2. écrire le contenu de la balise **<body>** contenant
   - l'importation de l'entête
   - une première section contenant
     - ✓ un label vide
     - ✓ le titre de la page : « *Nouvelle salle* » ou « *Salle – édition des informations* »
   - balise <*form*>, méthode « post »
   - une seconde section pour l'affichage/saisie du numéro de la salle avec affichage de l'erreur telle que la saisie du numéro est possible uniquement pour une nouvelle salle ; le booléen *$ediNum* à vrai indique que le numéro de salle est un champ de saisie.

Rappel : dans la feuille de style, le contenu de la balise <section> est affiché sur 2 colonnes :

```
display               : grid;
grid-template-columns : 1fr 8fr;
```

Numéro :

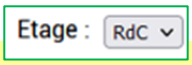
```
<section>
    <label  for="num">Numéro :</label>
    <div>
        <?php
        if ($editNum) {
        ?>
            <!--
            -->
            <input  id="num" name="num" type="text" size="5" maxlength="5"
               value="<?= htmlentities($valeurs['num']) ?>" />
            <br/>
            <span   class="erreur"><?= $erreurs['num'] ?></span>
        <?php
        }
        else echo($valeurs['num']);
        ?>
    </div>
</section>
```

htmlentities nécessaire pour les chaînes de caractères sinon la chaîne est tronquée à l'affichage à la première guillemet ou première quote rencontrée

Si mofification :
- affichage du numéro de salle passé en paramètre

- une troisième section pour la saisie de la désignation, avec affichage de l'erreur

Espace dédié pour l'affichage de l'erreur au plus près de la zone de saisie.
Règle : une zone ou un groupe de zones de saisie
➔ espace dédié pour l'affichage du message d'erreur même si la gestion de l'erreur n'est pas exigée



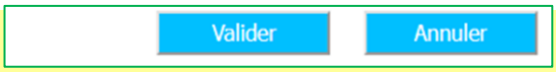

```
<section>
    <label  for="libelle">Désignation :</label>
    <div>
        <input  id="libelle" name="libelle" type="text" size="30" maxlength="30"
                value="<?= htmlentities($valeurs['libelle']) ?>" />
        <br />
        <span   class="erreur"><?= $erreurs['libelle'] ?></span>
    </div>
</section>
```

- une quatrième section pour l'affichage de la liste des étages, avec affichage de l'erreur


```
<section>
    <label  for="etage">Etage :</label>
    <div>
        <select name = "etage" />
        <?php
            foreach ($etages as $cle=>$valeur){
                echo "<option value='$cle'";
                if ($cle == $valeurs['etage']) {
                    echo ' selected';
                }
                echo ">", $valeur, "</option>";
            }
        ?>
        </select>
        <br />
        <span   class="erreur"><?= $erreurs['etage'] ?></span>
    </div>
</section>
```

- à l'affichage du formulaire en modification, sélection de l'étage lu dans la table
- sinon sélection de l'étage choisi si le formulaire a été « posté

- une dernière section pour l'affichage des boutons




```
<section>
    <label> </label>
    <div>
        <input type="submit" id="valider" name="valider" value="Valider" />
         
        <input type="submit" id="annuler" name="annuler" value="Annuler" />
    </div>
</section>
```

- fermer la <*form*>

Créer le fichier « **controleur/editSalle.php** »

```php
<?php
// type de demande : affichage, modification, suppression d'une salle
$op     = (isset($_GET['op'])?$_GET['op']:null);
$ajout  = ($op == 'a');
$modif  = ($op == 'm');
$suppr  = ($op == 's');

// récupération du numéro passé dans l'URL
$num    = (isset($_GET['num'])?$_GET['num']:null);
$editNum= $ajout;

// étages valides
$etages = ['RdC'=>'RdC',1=>'1',2=>'2',3=>'3'];

// création objet pour la récupération des données de la base
require_once('../modele/salleDAO.class.php');
$salleDAO = new SalleDAO();

// gestion des zones non modifiables en mode "modif"
$valeurs['num'] = null;
if ($modif) {
    $valeurs['num'] = $num; // numéro passé en paramètre
    $uneSalle = $salleDAO->getByNum($valeurs['num']);
}
if ($editNum) { // récupération numéro saisi si existe sinon valeur du numéro conservée
    $valeurs['num'] = (isset($_POST['num'])?trim($_POST['num']):$valeurs['num']);
}
```

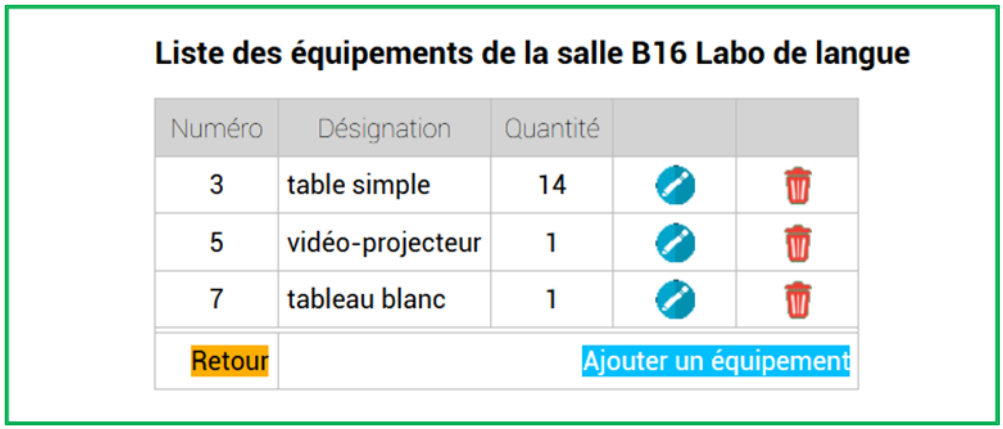
```php
// initialisation du titre
$titre = (($ajout)?'Nouvelle Salle':(($modif)?"Salle - édition des informations":null));
```

```php
// initialisation du tableau des erreurs :
// 1 erreur par champ de saisie ou par groupe de champs de saisie
$erreurs = ['num'=>"", 'libelle'=>'', 'etage'=>""];
// récupération des valeurs, autres que l'identifiant, saisies dans le formulaire
$valeurs['libelle'] = (isset($_POST['libelle'])?trim($_POST['libelle']):null);
$valeurs['etage'] = (isset($_POST['etage'])?trim($_POST['etage']):null);

// booléen qui sera vrai si validation sans erreur ou annulation
$retour = false;

// si validation, test des données du formulaire
if (isset($_POST['valider'])) {
    if (!isset($valeurs['num']) or strlen($valeurs['num'])==0)  {
        $erreurs['num']   = 'saisie obligatoire du numéro';   }
    else if ($editNum and $salleDAO->existe($valeurs['num']))   {
        $erreurs['num']   = 'Numéro de salle déjà existant.'; }
    if (!isset($valeurs['etage']) or strlen($valeurs['etage'])==0
        or !in_array($valeurs['etage'],$etages,true)) {
        $erreurs['etage'] = 'Etage non valide.';
    }
```

```
  // comptage du nombre d'erreurs détectées
    $nbErreurs = 0;
    foreach ($erreurs as $erreur){
        if ($erreur != "") $nbErreurs++;
    }

  // si aucune erreur alors enregistrement dans la base
    if ($nbErreurs == 0){
        $uneSalle = new Salle($valeurs['num'],$valeurs['libelle'], $valeurs['etage']);
        $retour = true;
        if ($ajout) {
            $salleDAO->insert($uneSalle);
        }
        else {
            $salleDAO->update($uneSalle);
        }
    }
}

// sinon si annulation
else if (isset($_POST['annuler']))  {
    $retour = true;
}

// sinon si suppression
else if ($suppr) {
// suppression
    $salleDAO->delete($num);
    $retour = true;
}

// sinon si modification
else if ($modif)    { // initialisation du formulaire
    $uneSalle = $salleDAO->getByNum($num);
    $valeurs['num']     = $uneSalle->getNum();
    $valeurs['libelle'] = $uneSalle->getLibelle();
    $valeurs['etage']   = $uneSalle->getEtage();
}


if ($retour)
{
    header("location: salles.php");
}

require_once("../vue/editSalle.view.php");
?>
```

Créer le fichier « **vue/salleEquipt.view.php** »

Liste des équipements d'une salle :



1. écrire le contenu de la balise **<head>** contenant
   - la mise en encodage utf-8, balise **<meta>**
   - le titre de la page avec la variable *$titre*
   - l'importation du fichier des styles
2. écrire le contenu de la balise **<body>** contenant
   - l'importation de l'entête
   - une première section contenant
     ✓ un label vide
     ✓ le titre de la page : `<h1>Liste des équipements de la salle <?=$titre ?></h1>`

     Raccourci séquence PHP : **<?** … **?>**
   - une seconde section contenant
     ✓ un label vide
     ✓ le tableau « table_salle_equipt » avec une ligne de titre,
     , des lignes de données (séquence PHP) :

```php
<?php
    foreach($lignes as $ligne) {
       echo $ligne;  // tableau de lignes à créer dans /controleur/salles.php
    }
?>
```

     , une ligne vide
     , une ligne avec les liens

```
<a href="salle.php?op=a" class="retour">Retour</a>
<a href="editSalleEquipt.php?op=a&num=<?=urlencode($num) ?>"
   class="ajout">Ajouter un équipement</a>
```

     retour à l'affichage de la liste des salles

     ajouter un nouvel équipement (*op=a*) dans la salle *$num*

Créer le fichier « **controleur/salleEquipt.php** »

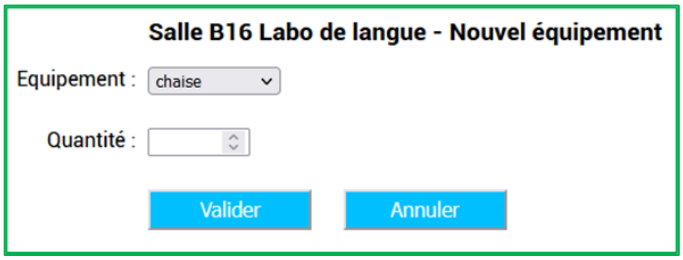
Ecrire les parties « ***à compléter*** »

```php
// récupération du numéro de salle passé en paramètre
$num = (isset($_GET['num'])?$_GET['num']:null);

// accès à la page uniquement si un numéro de salle est passé en paramètre
if ($num==null) {
    header("location: salles.php");
}

// importer la classe « salleDAO » et créer une instance de la classe
à compléter

// données de la salle $num
$uneSalle = à compléter

// construire le titre de la page
$titre  = à compléter

// liste des équipements de la salle
// importer la classe « EquiptBySalleDAO » et créer une instance de la classe
à compléter

// requête des équipements de la salle $num
$lesEquiptsBySalle = à compléter

$lignes = [];
foreach($lesEquiptsBySalle as $unEquiptBySalle)
{
    $unEquipt = $unEquiptBySalle->getEquipement();
    $ch = '';
    $ch .= '<td>' . à compléter .'</td>';
    $ch .= '<td>' . à compléter .'</td>';
    $ch .= '<td>' . à compléter .'</td>';

    // lien vers « editSalleEquipt.php » modification de l'équipement de l'inventaire de la salle
    $ch .='<td><a href="editSalleEquipt.php?op=m&num=' .urlencode($num)
                .'&id=' .urlencode($unEquipt->getId()) .'" >
                <img src="../vue/style/modification.png"></a></td>';
    // lien vers « editSalleEquipt.php » suppression de l'équipement de l'inventaire
    à compléter

    $lignes[] = "<tr>$ch</tr>";
}

// importer le fichier « view » correspondant
à compléter
```

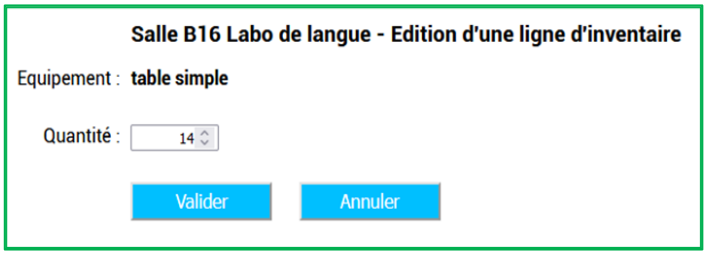
Liste des équipements de la salle **B16 Labo de langue**

Créer le fichier « **vue/editSalleEquipt.view.php** » : liste des salles

Ajout des équipements dans une salle :

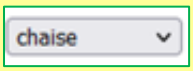


Edition d'une ligne d'inventaire d'une salle :

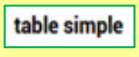


1. écrire le contenu de la balise **<head>** contenant
   - la mise en encodage utf-8, balise **<meta>**
   - le titre de la page avec la variable *$titre*
   - l'importation du fichier des styles
2. écrire le contenu de la balise **<body>** contenant
   - l'importation de l'entête
   - une première section contenant
     - ✓ un label vide
     - ✓ le titre de la page
   - balise <*form*>, méthode « post »
   - une seconde section pour l'affichage du libellé de l'équipement si modification ou pour la liste des équipements si ajout

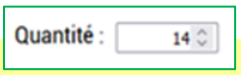
```
<section>
  <label  for="id">Equipement :</label>
  <div>
    <?php
    if ($editId) {
      echo '<select name = "id" />';
      foreach ($libelles as $cle=>$libelle){
        echo "<option value='$cle'";
        if ($cle == $valeurs['id']) {
          echo ' selected ';
        }
        echo ">", $libelle, "</option>";
      }
      echo '</select>';
      <br/>
      <span   class="erreur"><?= $erreurs['id'] ?></span>
    }
    else echo '<b>', $valeurs['libelle'], '</b>';
    ?>
  </div>
</section>
```



Sélection de l'équipement choisi
- soit saisi dans le formulaire « posté"
- soit passé en paramètre dans l'URL



- une troisième section pour la saisie de la quantité, avec affichage de l'erreur



Nombre entier : type « number » avec « step » à 1

```
<section>
    <label  for="qte">Quantité :</label>
    <div>
        <input  id="qte" name="qte" type="number" min="0" step="1"
               value="<?= $valeurs['qte'] ?>" />
        <br/>
        <span   class="erreur"><?= $erreurs['qte'] ?></span>
    </div>
</section>
```

- une dernière section pour l'affichage des boutons


```
<section>
    <label> </label>
    <div>
        à compléter
         
        à compléter
    </div>
</section>
```

- fermer la *<form>*

Créer le fichier « **controleur/editSalleEquipt.php** »

Écrire les parties ***« à compléter »*** dans le code suivant (s'aider du contenu du fichier « editSalle.php »)

```php
<?php
// type de demande : affichage, modification, suppression d'une salle
$op     = à compléter
$ajout  = à compléter
$modif  = à compléter
$suppr  = à compléter

// récupération du numéro de salle passé dans l'URL
$num    = à compléter
// récupération de l'id équipement passé dans l'URL
$id     = à compléter

$editId= $ajout;

// accès à la page uniquement si un numéro de salle est passé en paramètre
if ($num==null) {
    header("location: salles.php");
}
```

```php
// équipements sélectionnables qui ne sont pas encore dans l'inventaire de la salle
$libelles = []; // création tableau associatif id équipement ➔ libellé équipement
// importation classe « equipementDAO » et création d'une instance
require_once(à compléter );
$equipementDAO = à compléter
// récupération des équipements qui ne sont pas dans l'inventaire de la salle $num
$lesEquipements = à compléter
// remplir tableau associatif $libelles
foreach ($lesEquipements as $unEquipt) {
    à compléter
}
```

```php
// importation classe « equiptBySalleDAO » et création d'une instance
require_once(à compléter );
$equiptBySalleDAO = à compléter
```

```php
// gestion des zones non modifiables en mode "modif"
$valeurs['id'] = null;
if ($modif) {
    $valeurs['id']      = $id;
    // récupération de l'équipement $id à modifier dans la salle $num
    $unEquiptBySalle    = à compléter
    $valeurs['libelle'] = à compléter
}
```

```php
if ($editId) { // récupération de l'id équipement saisi si existe sinon valeur de l'id conservée
    à compléter
}
```

```php
// initialisation du titre « salle … nouvel équipement » ou « salle … édition d'une ligne d'ivnentaire »
require_once('../modele/salleDAO.class.php');
à compléter

// initialisation du tableau des erreurs :
// 1 erreur par champ de saisie ou par groupe de champs de saisie
$erreurs = à compléter
/ récupération des valeurs, autres que l'identifiant, saisies dans le formulaire
à compléter

// booléen qui sera vrai si validation sans erreur ou annulation
$retour = false;
```

Salle B16 Labo de langue - Nouvel équipement

Salle B16 Labo de langue - Edition d'une ligne d'inventaire

```php
// si validation, test des données du formulaire
if (isset($_POST['valider'])) {
    if à compléter

    // comptage du nombre d'erreurs détectées
    à compléter

    // si aucune erreur alors enregistrement dans la base
    if ($nbErreurs == 0){
        // récupération des données de l'équipement $valeurs['id']
        $unEquipt       = à compléter
        $unEquiptBySalle= new EquiptBySalle(à compléter);
        $retour = true;
        à compléter
    }
}
// sinon si annulation
else if (isset($_POST['annuler']))  {
    à compléter
}
// sinon si suppression
else if ($suppr) {
    à compléter
}
// sinon si modification
else if ($modif)    {
    $unEquiptBySalle = $equiptBySalleDAO->getByNumSalleByIdEquipt ($num,$id);
// affectation de la quantité, les autres valeurs ont déjà été renseignées
// voir en début de fichier la partie gestion des zones non modifiables en mode "modif"
    $valeurs['qte']     = $unEquiptBySalle->getQte();
}
```

```php
if ($retour) {
// redirection vers l'inventaire de la salle $num
    à compléter
}

require_once("../vue/editSalleEquipt.view.php");
?>
```