4.1. Les fichiers « ts/js » du contrôleur

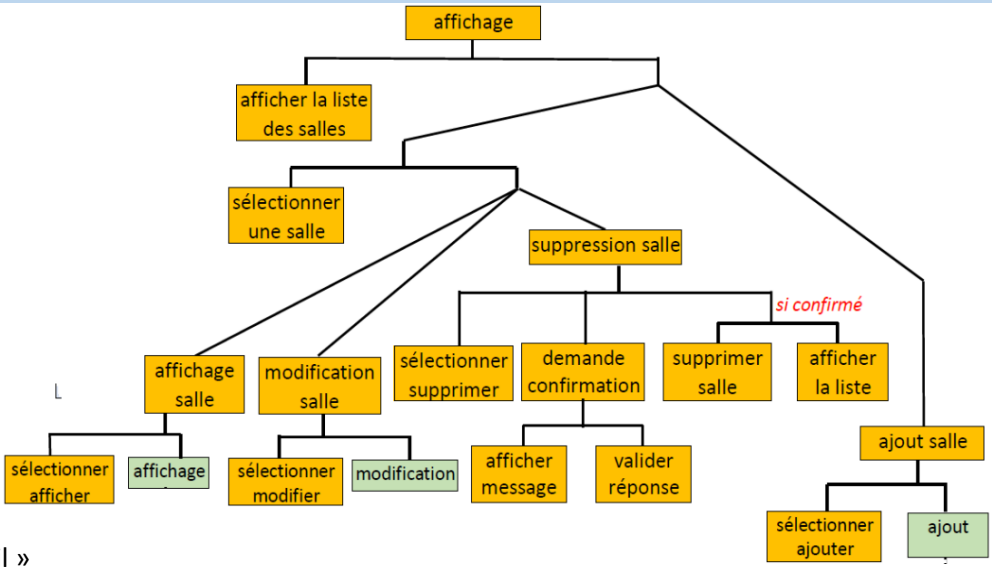
Les derniers fichiers à écrire sont ceux du dossier « src/contrôleur ». Ils vont faire lien entre les interfaces (fichiers « HTML ») et les données extraites ou sauvegardées dans la base de données.

Ce sont ces fichiers qui vont

- contrôler les données saisies,
- afficher les données renvoyées par les fichiers d'accès aux données de la base
- gérer la navigation dans l'application

4.1.1. traitements sur le fichier HTML « salle_liste » : liste des salles avec boutons de traitement

Evénements à implémenter



- « click » sur le bouton « Détail »
  ➔ « afficherClick » ➔ affichage détaillé des informations de la salle sélectionnée
- « click » sur le bouton « Ajouter »
  ➔ « ajouterClick » ➔ ajouter une salle et ses informations
- « click » sur le bouton « Détail »
  ➔ « modifierClick » ➔ modification des informations de la salle sélectionnée
- « click » sur le bouton « Supprimer »
  ➔ « supprimerClick » : suppression de la salle sélectionnée

créer le fichier « src/controleur/class_salle_liste.ts » classe contenant les traitements sur la vue HTML

```
// importation des classes de gestion des données d'une salle et de ses équipements
import {UneSalle, LesSalles, TSalles}   from "../modele/data_salle.js"
import {LesDepts}                       from "../modele/data_departement.js"
import {LesTypEquiptsBySalle}           from "../modele/data_equipement.js"

// déclaration de l'ensemble des zones de saisie et d'affichage nécessaires à la gestion du formulaire type
type TSalleListeForm = {
   divTitre :HTMLElement, btnAjouter :HTMLInputElement, tableSalle : HTMLTableElement

}

class VueSalleListe {
   private _form    : TSalleListeForm;

   get form()   :TSalleListeForm  { return this._form     }
```

```
init(form : TSalleListeForm ):void {
   this._form      = form;

   const lesSalles             = new LesSalles;
   const lesDepts              = new LesDepts();
   const lesTypEquiptsBySalle = new LesTypEquiptsBySalle();
   const data = lesSalles.all();

   this.form.divTitre.textContent = 'Liste des salles'; // construction du titre

   for (let num in data) {
      const uneSalle : UneSalle = data[num];
      const tr = this.form.tableSalle.insertRow(); // création nlle ligne dans tableau

      let balisea : HTMLAnchorElement; // déclaration balise <a>
      // création balise <a> pour appel page visualisation du détail de la salle
      balisea = document.createElement("a")
      balisea.classList.add('img_visu') // définition class contenant l'image (voir css)
      balisea.onclick = function():void { vueSalleListe.detailSalleClick(uneSalle.numSalle); }
      tr.insertCell().appendChild(balisea) // création nlle cellule dans ligne

      tr.insertCell().textContent = uneSalle.numSalle;
      tr.insertCell().textContent = uneSalle.libSalle;
      tr.insertCell().textContent = uneSalle.etage;
      tr.insertCell().textContent = uneSalle.codeDept;
      tr.insertCell().textContent = lesDepts.byCodeDept(uneSalle.codeDept).nomDept;
      tr.insertCell().textContent =
       lesTypEquiptsBySalle.getTotalNbEquipt(lesTypEquiptsBySalle.byNumSalle(num)).toFixed(0);

      // création balise <a> pour appel page modification du détail de la salle
      balisea = document.createElement("a")
      balisea.classList.add('img_modification') // définition class contenant l'image (voir css)
      balisea.onclick = function():void { vueSalleListe.modifierSalleClick(uneSalle.numSalle); }
      tr.insertCell().appendChild(balisea)
      // création balise <a> pour appel page suppression d'une salle
      balisea = document.createElement("a")
      balisea.classList.add('img_corbeille') // définition class contenant l'image (voir css)
      balisea.onclick = function():void { vueSalleListe.supprimerSalleClick(uneSalle.numSalle); }
      tr.insertCell().appendChild(balisea)
   }

   // définition événement onclick sur bouton "ajouter"
   this.form.btnAjouter.onclick = function():void { vueSalleListe.ajouterSalleClick(); }
}
```

```
   detailSalleClick(num : string):void {
      // redirection vers « salle_edit.html »avec indication du statut « affi » et du numéro de salle
      location.href = "salle_edit.html?affi&" +encodeURIComponent(num);
   }
   modifierSalleClick(num : string):void {
      // redirection vers « salle_edit.html »avec indication du statut « modif » et du numéro de salle
      location.href = "salle_edit.html?modif&" +encodeURIComponent(num)
   }
   supprimerSalleClick(num : string):void {
      // redirection vers « salle_edit.html »avec indication du statut »suppr »  et du numéro de salle
      location.href = "salle_edit.html?suppr&" +encodeURIComponent(num)
   }
   ajouterSalleClick():void {
      // redirection vers « salle_edit.html »avec indication du statut « ajout »
      location.href = "salle_edit.html?ajout"
   }
}

let vueSalleListe = new VueSalleListe;
export {vueSalleListe}
```

transpiler et corriger les éventuelles erreurs

créer le fichier « src/controleur/salle_liste.ts » :
initialisation avec passage des composants identifiés « id » du fichier HTML, définition des événements

```
import {vueSalleListe} from "../controleur/class_salle_liste.js"

vueSalleListe.init( { divTitre    :document.querySelector('[id=div_salle_liste_titre]')
                     ,btnAjouter  :document.querySelector('[id=btn_salle_ajouter]')
                     ,tableSalle  :document.querySelector('[id=table_salle]')
                    });
```

transpiler et corriger les éventuelles erreurs

Tester l'application

https://devweb.iutmetz.univ-lorraine.fr/~toto3u/ihm/tp_inventaire/vue/inventaire.html

Résultat à obtenir :

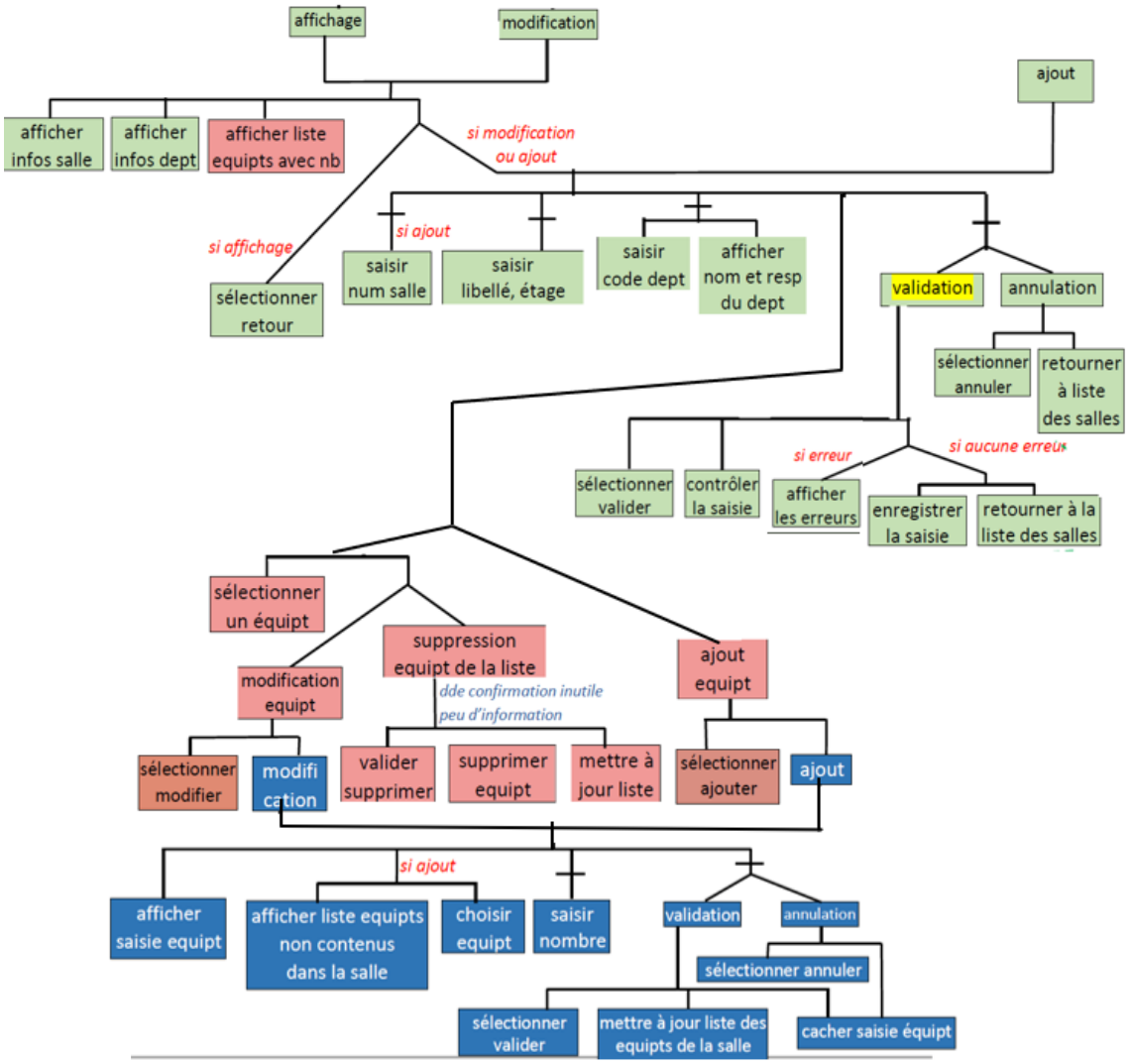
Liste des salles

	Numéro	Libellé	Etage	Département		Equipements		
	B16	Labo de langue	1	GEA	Gestion des Entreprises et Administrations	16		
	F13	machine PC	1	INFO	Informatique	16		
	E23		2	INFO	Informatique	1		
	C14		1	GEA	Gestion des Entreprises et Administrations	33		
	E36		3	INFO	Informatique	2		
	F39	machine PC	3	INFO	Informatique	62		

Ajouter

4.1.2. traitements sur le fichier HTML « salle_edit » : édition d'une salle

Evénements à implémenter



- « click » sur le bouton « Retour »
  ➔ « retourClick » ➔ fermer le fichier « salle_edit »
- « click » sur le bouton « Annuler »
  ➔ « retourClick » ➔ fermer le fichier « salle_edit »
- « click » sur le bouton « Valider »
  ➔ « validerClick » ➔ enregistrer les données du formulaire (information de la salle et liste des équipements de la salle) après vérification des valeurs
- « change » sur la zone d'édition du code de département
  ➔ « detailDepartement » ➔ affichage du détail des informations du département
- « click » sur le bouton « Ajouter un équipement »
  ➔ « ajouterEquiptClickClick » ➔ afficher la partie d'ajout d'un équipement à la salle
- « click » sur le bouton « Modifier la quantité d'un équipement »
  ➔ « modifierEquiptClickClick » ➔ afficher la partie d'ajout d'un équipement à la salle
- « click » sur le bouton « Retirer un équipement de la salle »
  ➔ « supprimerEquiptClickClick » ➔ afficher la partie d'ajout d'un équipement à la salle
- « click » sur le bouton « Valider l'ajout/mise à jour d'un équipement »
  ➔ « validerEquiptClickClick » ➔ mettre à jour la liste des équipements de la salle après vérification des données
- « click » sur le bouton « Annuler l'ajout/mise à jour d'un équipement »
  ➔ « annulerEquiptClickClick » ➔ retour à la liste des équipements de la salle

créer le fichier « src/controleur/class_salle_edit.ts » classe contenant les traitements sur la vue HTML

```
import {UneSalle, LesSalles}from "../modele/data_salle.js"
import {UnDept, LesDepts}    from "../modele/data_departement.js"
import {UnTypEquiptBySalle, LesTypEquiptsBySalle, TTypEquiptsBySalle, UnTypEquipt,
LesTypEquipts}   from "../modele/data_equipement.js"

type TStatutValeur = 'correct' | 'vide' | 'inconnu' | 'doublon'
type TErreur    =  { statut : TStatutValeur, msg:{ [key in TStatutValeur] : string } }
type TSalleEditForm = {
   divDetail:HTMLElement, divTitre:HTMLElement
   , edtNum:HTMLInputElement, edtLib:HTMLInputElement
   , edtEtage:HTMLInputElement, edtCodeDept:HTMLInputElement
   , btnRetour:HTMLInputElement, btnValider:HTMLInputElement, btnAnnuler:HTMLInputElement
   , lblDetailDept:HTMLLabelElement
   , lblNumErreur :HTMLLabelElement, lblEtageErreur:HTMLLabelElement
   , lblDeptErreur:HTMLLabelElement, lblEquiptErreur:HTMLLabelElement
   , divSalleEquipt : HTMLDivElement, divSalleEquiptEdit : HTMLDivElement
   , btnAjouterEquipt:HTMLInputElement
   , lblTotal : HTMLLabelElement, tableEquipement : HTMLTableElement
   , listeEquipt:HTMLSelectElement, edtQte:HTMLInputElement
   , btnValiderEquipt:HTMLInputElement, btnAnnulerEquipt:HTMLInputElement
   , lblSelectEquiptErreur:HTMLLabelElement, lblQteErreur:HTMLLabelElement
}

class VueSalleEdit {
   private _form      : TSalleEditForm
   private _params  : string[];  // paramètres reçus par le fichier HTML
                          // tel que  params[0] :  mode affi, modif, suppr, ajout
                          //      params[1] : id en mode affi, modif, suppr
   private _grille  : TTypEquiptsBySalle;   // tableau des équipements de la salle

   private _erreur  : {
   // tableau contenant les messages d'erreur pour chaque type d'erreur pour chaque zone de saisie à vérifier
      [key:string] : TErreur
   }

   get form()   :TSalleEditForm         { return this._form     }
   get params() :string[]               { return this._params   }
   get grille() :TTypEquiptsBySalle     { return this._grille   }
   get erreur() :{[key:string]:TErreur}{ return this._erreur   }
```

```
initMsgErreur():void {
   // les erreurs "champ vide", "valeur inconnue", "doublon"
   //sont les trois principales erreurs dans un formulaire
   // pour chaque champ à contrôler (événement onChange),
   //création des 3 messages d'erreur + message pour correct
   // avec chaîne vide si pas d'erreur générée pour un type d'erreur potentielle
   this._erreur = {
      edtNum     :  {statut :'vide'
                     , msg:{correct:""
                          , vide:"Le numéro de salle doit être renseigné."
                          ,inconnu:"Le numéro ne peut contenir que des lettres et des chiffres."
                          ,doublon:"Le numéro de salle est déjà attribué."} }
      ,edtEtage  :  {statut :'vide'
                     , msg:{correct:""
                          , vide:"L'étage doit être renseigné."
                          ,inconnu:""
                          ,doublon:""} }
      ,edtCodeDept:{statut :'vide'
                     , msg:{correct:""
                          , vide:"Le département doit être renseigné."
                          ,inconnu:"Département inconnu."
                          ,doublon:""} }
      ,equipt    :  {statut :'vide'
                     , msg:{correct:""
                          , vide:"La salle doit contenir au moins un équipement."
                          ,inconnu:""
                          ,doublon:""} }
      ,listeEquipt:{statut :'vide'
                     , msg:{correct:""
                          , vide:"Aucun équipement choisi"
                          ,inconnu:""
                          ,doublon:""} }
      ,edtQte    :  {statut :'vide'
                     , msg:{correct:""
                          , vide:"La quantité doit être un nombre entier supérieur à 0"
                          ,inconnu:""
                          ,doublon:""} }
   }
}
```

```
init(form:TSalleEditForm):void {
   this._form   = form;
   this._params = location.search.substring(1).split('&');
   // params[0] :  mode affi, modif, suppr, ajout
   // params[1] : id en mode affi, modif, suppr
   this.form.divSalleEquiptEdit.hidden = true;
   this.initMsgErreur();
   let titre : string;
   switch (this.params[0]) {
      case 'suppr' : titre = "Suppression d'une salle";   break;
      case 'ajout' : titre = "Nouvelle salle";      break;
      case 'modif' : titre = "Modification d'une salle";  break;
      default   : titre = "Détail d'une salle";
   }
   this.form.divTitre.textContent = titre;
   const lesSalles = new LesSalles;
   const affi = this.params[0] === 'affi';
   if (this.params[0] !== 'ajout')
   {   // affi ou modif ou suppr
      const salle = lesSalles.byNumSalle(this._params[1]);
      this.form.edtNum.value         = salle.numSalle;
      this.form.edtLib.value         = salle.libSalle;
      this.form.edtEtage.value       = salle.etage;
      this.form.edtCodeDept.value    = salle.codeDept;
      this.form.edtNum.readOnly      = true;
      this.form.edtLib.readOnly      = affi;
      this.form.edtEtage.readOnly    = affi;
      this.form.edtCodeDept.readOnly = affi;
      this.erreur.edtNum.statut      = "correct";
      this.detailDepartement(salle.codeDept);
   }
   this.affiEquipement();
   if (this.params[0] === 'suppr') {
   // temporisation 1 seconde pour afficher les données de la salle avant demande de confirmation de la supression
      setTimeout(() => {this.supprimer(this.params[1])}, 1000);
   }
   this.form.btnRetour.hidden  = !affi;
   this.form.btnValider.hidden = affi;
   this.form.btnAnnuler.hidden = affi;
   this.form.btnAjouterEquipt.hidden = affi;
   // définition des événements
   this.form.edtCodeDept.onchange  = function():void {
                        vueSalleEdit.detailDepartement(vueSalleEdit.form.edtCodeDept.value); }
   this.form.btnRetour.onclick    = function():void { vueSalleEdit.retourClick(); }
   this.form.btnAnnuler.onclick   = function():void { vueSalleEdit.retourClick(); }
   this.form.btnValider.onclick   = function():void { vueSalleEdit.validerClick(); }
   this.form.btnAjouterEquipt.onclick  = function():void { vueSalleEdit.ajouterEquiptClick(); }
   this.form.btnValiderEquipt.onclick  = function():void { vueSalleEdit.validerEquiptClick(); }
   this.form.btnAnnulerEquipt.onclick  = function():void { vueSalleEdit.annulerEquiptClick(); }
}
```

```
detailDepartement(valeur : string):void {
   const err       = this.erreur.edtCodeDept
   const lesDepts  = new LesDepts;
   const detail    = this.form.lblDetailDept;
   detail.textContent = "";
   err.statut         = "correct";
   const chaine : string = valeur.trim();
   if (chaine.length > 0) {
      const dept : UnDept = lesDepts.byCodeDept(chaine);
      if (dept.codeDept !== "") { // département trouvé
         detail.textContent
         = dept.nomDept +"\r\n" +"Responsable : " +dept.respDept;
      }
      else {
         err.statut = 'inconnu';
         detail.textContent = err.msg.inconnu;
      }
   }
   else err.statut = 'vide';
}
affiEquipement():void {
   const lesTypEquiptsBySalle = new LesTypEquiptsBySalle();
   this._grille = lesTypEquiptsBySalle.byNumSalle(this.params[1]);
   this.affiGrilleEquipement();
}
affiGrilleEquipement():void {
   while (this.form.tableEquipement.rows.length > 1) { this.form.tableEquipement.rows[1].remove();   }
   let   total = 0;
   for (let id in this._grille) {
      const unTypEquiptBySalle : UnTypEquiptBySalle = this.grille[id];
      const tr = this.form.tableEquipement.insertRow();
      tr.insertCell().textContent = unTypEquiptBySalle.unTypEquipt.libEquipt;
      tr.insertCell().textContent = unTypEquiptBySalle.qte.toFixed(0);

      const affi = this.params[0] === 'affi';
      if (!affi) {
         let balisea : HTMLAnchorElement; // déclaration balise <a>
         // création balise <a> pour appel modification équipement dans salle
         balisea = document.createElement("a")
         balisea.classList.add('img_modification')
         balisea.onclick = function():void { vueSalleEdit.modifierEquiptClick(id); }
         tr.insertCell().appendChild(balisea)
         // création balise <a> pour appel suppression équipement dans salle
         balisea = document.createElement("a")
         balisea.classList.add('img_corbeille')
         balisea.onclick = function():void { vueSalleEdit.supprimerEquiptClick(id); }
         tr.insertCell().appendChild(balisea)
      }
      total += unTypEquiptBySalle.qte
   }
   this.form.lblTotal.textContent = total.toString();
```

```
}

supprimer(numSalle : string):void   {
   if (confirm("Confirmez-vous la suppression de la salle "+numSalle)) {
      let lesTypEquiptsBySalle : LesTypEquiptsBySalle = new LesTypEquiptsBySalle();
      lesTypEquiptsBySalle.delete(numSalle);  // suppression dans la base des equipements de la salle

      const lesSalles = new LesSalles;
      lesSalles.delete(numSalle);             // suppression dans la base de la salle
   }
   this.retourClick();
}

verifNum(valeur : string):void {
   const lesSalles = new LesSalles;
   const err = this.erreur.edtNum
   err.statut = "correct";
   const chaine : string = valeur.trim();
   if (chaine.length > 0) {
      if (! chaine.match(/^([a-zA-Z0-9]+)$/)) {
      // expression régulière qui teste si la chaîne ne contient rien d'autre
      // que des caractères alphabétiques minuscules ou majuscules et des chiffres
         this.erreur.edtNum.statut = 'inconnu';
      }
      else if ( (this.params[0] === 'ajout')  &&  (lesSalles.idExiste(chaine)) ) {
         this.erreur.edtNum.statut = 'doublon';
      }
   }
   else err.statut = 'vide';
}

verifEtage(valeur : string):void {
   const err = this.erreur.edtEtage
   err.statut = "correct";
   const chaine : string = valeur.trim();
   if (chaine.length === 0) {
      err.statut = 'vide';
   }
}

traiteErreur(uneErreur:TErreur, zone : HTMLElement):boolean {
   let correct = true;
   zone.textContent ="";
   if (uneErreur.statut !== "correct") { // non correct ==> erreur
      if (uneErreur.msg[uneErreur.statut] !== '') {  // erreur
         zone.textContent = uneErreur.msg[uneErreur.statut];
         correct = false;
      }
   }
   return correct;
}
```

```
validerClick():void {
      let correct = true;
      this.verifNum(this._form.edtNum.value);
      this.verifEtage(this._form.edtEtage.value);

      if (JSON.stringify(this.grille) === '{}' ) {   this._erreur.equipt.statut  = 'vide'}
      else this._erreur.equipt.statut = "correct";

      correct = this.traiteErreur(this._erreur.edtNum, this.form.lblNumErreur) && correct;
      correct = this.traiteErreur(this._erreur.edtEtage, this.form.lblEtageErreur) && correct;
      correct = this.traiteErreur(this._erreur.edtCodeDept, this.form.lblDeptErreur) && correct;
      correct = this.traiteErreur(this._erreur.equipt, this.form.lblEquiptErreur) && correct;

      const lesSalles = new LesSalles;
      const salle   = new UneSalle;
      if (correct) {
         salle.numSalle  = this.form.edtNum.value;
         salle.libSalle  = this.form.edtLib.value;
         salle.etage   = this.form.edtEtage.value;
         salle.codeDept  = this.form.edtCodeDept.value;
         if (this._params[0] === 'ajout') {
            lesSalles.insert(salle);
         }
         else {
            lesSalles.update(salle);
         }

         const lesTypEquiptsBySalle : LesTypEquiptsBySalle = new LesTypEquiptsBySalle;
         lesTypEquiptsBySalle.delete(salle.numSalle);
         lesTypEquiptsBySalle.insert(salle.numSalle, this.grille);

         this.retourClick();
      }
   }

   retourClick():void {
      location.href = "salle_liste.html";
   }

   // gestion des équipements de la salle
   modifierEquiptClick(id : string):void {
      this.afficherEquitpEdit():
      const lesTypEquipts = new LesTypEquipts();
      const unTypEquipt : UnTypEquipt = lesTypEquipts.byIdEquipt(id) ;
      this.form.listeEquipt.length = 0;
      this.form.listeEquipt.options.add(new Option(unTypEquipt.libEquipt, id));    // text, value = 0;
      this.form.listeEquipt.selectedIndex = 0;
      this.form.edtQte.value  = this._grille[id].qte.toFixed(0);

   }
```

```
ajouterEquiptClick():void {
   this.afficherEquitpEdit();

   // réinitialiser la liste des équipements à choisir
   this.form.listeEquipt.length = 0;
   const lesTypEquipts = new LesTypEquipts;
   const data  = lesTypEquipts.all();
   const idEquipts = [];
   for (let i in this._grille) {
      idEquipts.push(this._grille[i].unTypEquipt.idEquipt);
   }
   for (let i in data) {
      const id = data[i].idEquipt;
      if (idEquipts.indexOf(id) === -1) { // pas dans la liste des équipements déjà dans la salle
         this._form.listeEquipt.options.add(new Option(data[i].libEquipt, id));  // text, value
      }
   }
}
supprimerEquiptClick(id : string):void {
   if (confirm("Confirmez-vous le retrait de l'équipement de la salle ")) {
      delete(this._grille[id]);
      this.affiGrilleEquipement();
   }
}
afficherEquitpEdit():void {
   this.form.divSalleEquiptEdit.hidden = false;
   this.form.divDetail.style.pointerEvents = 'none';
   this.form.divSalleEquiptEdit.style.pointerEvents = 'auto';
   this.form.btnAjouterEquipt.hidden = true;
   this.form.btnAnnuler.hidden = true;
   this.form.btnValider.hidden = true;
}
cacherEquitpEdit():void {
   this.form.divSalleEquiptEdit.hidden = true;
   this.form.divDetail.style.pointerEvents = 'auto';
   this.form.btnAjouterEquipt.hidden = false;
   this.form.btnAnnuler.hidden = false;
   this.form.btnValider.hidden = false;
}
verifListeEquipt():void {
   const err = this._erreur.listeEquipt
   err.statut = "correct";
   const cible  = this._form.listeEquipt;
   if (cible.value === "") {
      err.statut = 'vide'
   }
}
```

```
   verifQte():void {
      const err = this._erreur.edtQte
      err.statut = "correct";
      const valeur : string = this._form.edtQte.value;
      if ( ! ( (Number.isInteger(Number(valeur))) && (Number(valeur)>0) ) ) {
         err.statut = 'vide'
      }
   }

   validerEquiptClick():void {
      let correct = true;
      this.verifListeEquipt();
      this.verifQte();

      correct = this.traiteErreur(this._erreur.listeEquipt, this.form.lblSelectEquiptErreur) && correct;
      correct = this.traiteErreur(this._erreur.edtQte, this.form.lblQteErreur) && correct;

      if (correct) {
         const lesTypEquipts = new LesTypEquipts;
         // ajout visuel de la ligne dans la grille tabulaire de la liste des équipements d'une salle
         const unTypEquipt : UnTypEquipt = lesTypEquipts.byIdEquipt(this._form.listeEquipt.value);
         const unTypEquiptBySalle: UnTypEquiptBySalle
               = new UnTypEquiptBySalle(unTypEquipt, parseInt(this._form.edtQte.value));

         this._grille[unTypEquipt.idEquipt]  = unTypEquiptBySalle;
         this.affiGrilleEquipement();
         this.annulerEquiptClick();
      }
   }

   annulerEquiptClick():void {
      this.cacherEquitpEdit();
   }
}

let vueSalleEdit = new VueSalleEdit;

export { vueSalleEdit }
```

transpiler et corriger les éventuelles erreurs

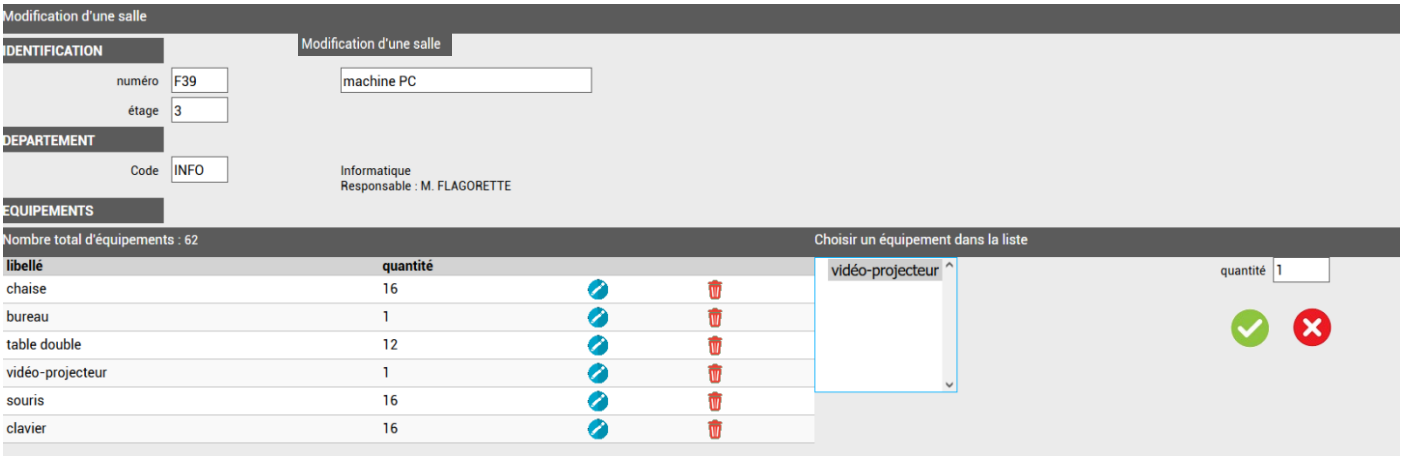
créer le fichier « src/controleur/salle_edit.ts » : initialisation avec passage des composants du identifiés « id » du fichier HTML, définition des événements

```
import { vueSalleEdit } from "../controleur/class_salle_edit.js";

vueSalleEdit.init( {
          divDetail          :document.querySelector('[id=div_salle_detail]')
          ,edtNum            :document.querySelector('[id=edt_salle_num]')
          ,divTitre          :document.querySelector('[id=div_salle_titre]')
          ,edtLib            :document.querySelector('[id=edt_salle_lib]')
          ,edtEtage          :document.querySelector('[id=edt_salle_etage]')
          ,edtCodeDept       :document.querySelector('[id=edt_salle_codedept]')
          ,btnRetour         :document.querySelector('[id=btn_salle_retour]')
          ,btnValider        :document.querySelector('[id=btn_salle_valider]')
          ,btnAnnuler        :document.querySelector('[id=btn_salle_annuler]')
          ,lblDetailDept     :document.querySelector('[id=lbl_salle_detail_dept]')
          ,lblNumErreur      :document.querySelector('[id=lbl_erreur_num]')
          ,lblEtageErreur    :document.querySelector('[id=lbl_erreur_etage]')
          ,lblDeptErreur     :document.querySelector('[id=lbl_erreur_dept]')
          ,lblEquiptErreur   :document.querySelector('[id=lbl_erreur_equipt]')
          ,divSalleEquipt    :document.querySelector('[id=div_salle_equipement]')
          ,divSalleEquiptEdit:document.querySelector('[id=div_salle_equipement_edit]')
          ,btnAjouterEquipt  :document.querySelector('[id=btn_equipement_ajouter]')
          ,lblTotal          :document.querySelector('[id=lbl_equipement_total]')
          ,tableEquipement   :document.querySelector('[id=table_equipement]')
          ,listeEquipt       :document.querySelector('[id=select_equipement]')
          ,edtQte            :document.querySelector('[id=edt_equipement_qte]')
          ,btnValiderEquipt  :document.querySelector('[id=btn_equipement_valider]')
          ,btnAnnulerEquipt  :document.querySelector('[id=btn_equipement_annuler]')
          ,lblSelectEquiptErreur:document.querySelector('[id=lbl_erreur_select_equipement]')
          ,lblQteErreur      :document.querySelector('[id=lbl_erreur_qte]')
} );
```

transpiler et corriger les éventuelles erreurs

Résultat à obtenir avec édition de la salle F39 en mode modification, quantité vidéo-projecteur à modifier



Le développement de l'application est presque terminé.
La dernière opération consiste à tester son application avec une série de tests : c'est l'étape du test d'acceptation.

*En informatique, le **test d'acceptation** (ou recette) est une phase de développement des projets, visant à assurer formellement que le produit est conforme aux spécifications.*