

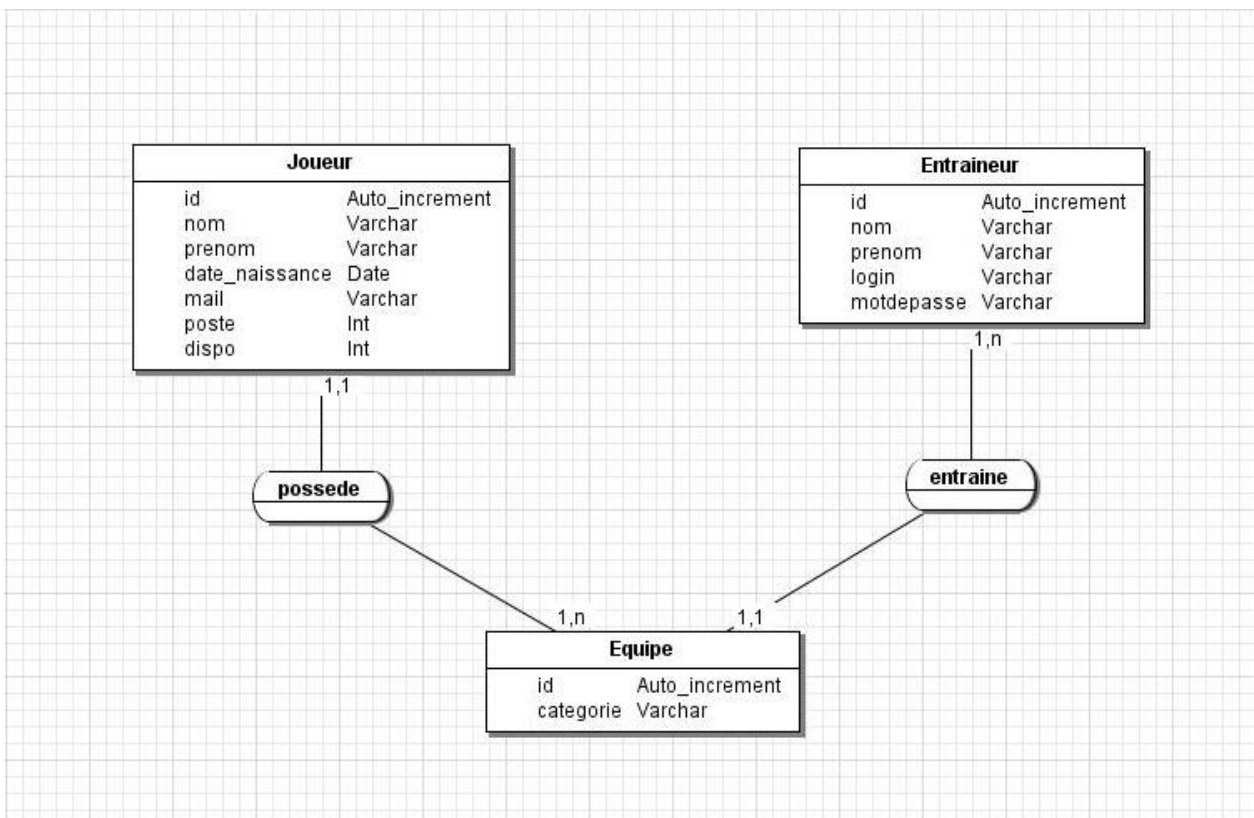
Contexte et MCD :

Cette application a été réalisée dans le contexte Maison Des Liges (MDL).

Les technologies utilisées sont : Java, JavaEE, CSS, Bootstrap, JavaScript, JQuery, Ajax, MySQL, phpMyadmin, serveur Tomcat.

L'application est destinée aux entraîneurs de basket-ball.

Le but de l'application est d'aider l'entraîneur à faire ses compositions d'équipes pour les matchs et de gérer plusieurs catégories d'équipe ainsi que leurs joueurs.



Ceci est le MCD final, il a été raccourci par rapport à la première version (des tables et des champs ont été inutilisés). La base de données n'est pas compliquée, la difficulté du projet réside sur les technologies utilisées, le JavaEE, le JavaScript et l'Ajax étant plus ou moins inconnus au départ.

SupportEntraîneur

L'utilisateur, ici un entraîneur, doit d'abord passer par un formulaire basique de connexion relié à la base de données.

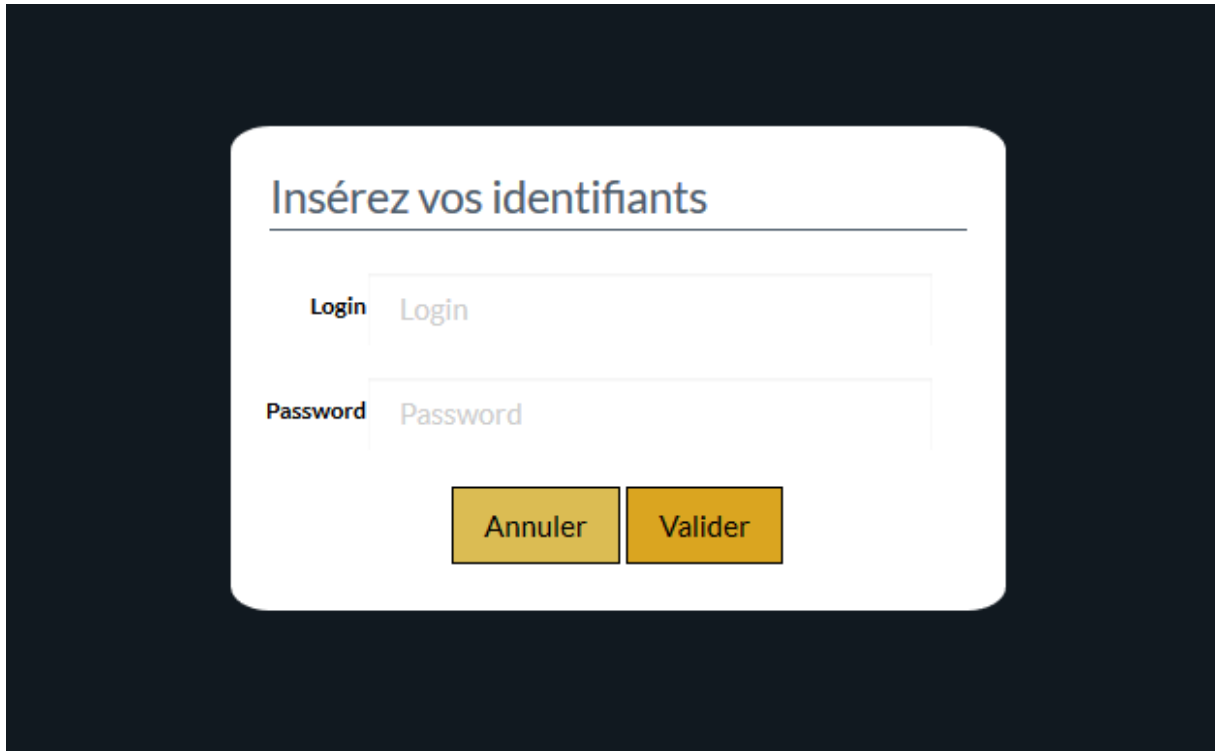


Image 1 : formulaire de connexion

PROCESSUS D'IDENTIFICATION :

```
<div id="filter">
  <div id="box">
    <form class="form-horizontal" method="POST" action="servletpe">
      <fieldset>
        <legend>Insérez vos identifiants</legend>
        <div class="form-group">
          <label for="inputEmail" class="col-lg-2 control-label">Login</label>
          <div class="col-lg-10">
            <input class="form-control" name="inputLogin" placeholder="Login"
              type="text">
          </div>
        </div>
        <div class="form-group">
          <label for="inputPassword" class="col-lg-2 control-label">Password</label>
          <div class="col-lg-10">
            <input class="form-control" name="inputPassword"
              placeholder="Password" type="password">
          </div>
        </div>
        <div class="form-group">
          <div class="col-lg-10 col-lg-offset-2">
            <button type="reset" class="btn btn-default">Annuler</button>
            <button type="submit" class="btn btn-primary">Valider</button>
          </div>
        </div>
      </fieldset>
    </form>
  </div>
</div>
```

Image 2 : formulaire de connexion

En rouge, nous avons le type d'envoi des données avec la méthode POST afin d'envoyer les informations de façon invisible, ainsi que l'action /servletppe qui est notre contrôleur où le processus s'exécute. Les « name » signifient qu'on attribue un nom au champ texte afin de pouvoir réutiliser son contenu. Une fois les contenus remplis, l'action se déroule dans la servlet avec le code suivant :

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    HttpSession session = request.getSession();

    if (request.getParameter("inputLogin") != null) {
        // formulaire de connexion
        System.out.println("servletppe access");
        String email = request.getParameter("inputLogin");
        String pwd = request.getParameter("inputPassword");

        EntraîneurDAO aa = new EntraîneurDAO();
        Entraîneur a = aa.auth(email, pwd);
        System.out.println(a);

        session.setAttribute("user", a);
    }
}
```

Image 3 : formulaire de connexion

Ici, dans notre méthode POST, nous avons l'objet HttpSession, propre à J2EE, qui va nous permettre d'effectuer notre authentification. Les « names » sont récupérés et attribués à des variables qui seront utilisées dans une fonction « auth() » se trouvant dans notre DAO.

```
public Entraîneur auth(String email, String pwd){
    try{
        getConnection();
        conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_UPDATABLE);

        String sql = "SELECT * FROM entraineur WHERE login = ? AND motdepasse = ?";
        prepare = conn.prepareStatement(sql); //on crée la requête
        prepare.setString(1,email); //on place les variables dans la requête
        prepare.setString(2,pwd);
        rs = prepare.executeQuery(); //on exécute la requête et on récupère le résultat

        if(rs.first()){
            return new Entraîneur(rs.getInt("id"),rs.getString("Nom"),rs.getString("Prenom"),rs.getString("Login"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            prepare.close();
            closeConnection(); //on ferme la base de donnée
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    return null;
}
```

Image 4 : fonction "auth()"

Cette fonction fait le lien entre les champs texte remplis et les informations de notre base de données.

		id	nom	prenom	login	motdepasse
<input type="checkbox"/>	Modifier	1	Toto	Fischer	toto	toto123
<input type="checkbox"/>	Modifier	2	John	John	john	john

Une simple ligne de code est ainsi nécessaire à l'index pour récupérer notre session.

```
Entraîneur ent = (Entraîneur) (session.getAttribute("user"));
```

En cas d'identifiants incorrects ou vides, un message d'erreur provenant du serveur est activé.

Etat HTTP 404 - login/mdp vide

type Rapport d'état

message login/mdp vide

description La ressource demandée n'est pas disponible.

```
if ((email.trim().equals("")) || (pwd.trim().equals(""))) {  
    response.sendError(HttpServletResponse.SC_NOT_FOUND, "login/mdp vide");  
    return;  
}
```

Apache Tomcat/8.0.27

Etat HTTP 404 - login/mdp incorrect

type Rapport d'état

message login/mdp incorrect

description La ressource demandée n'est pas disponible.

```
if (a == null) {  
    response.sendError(HttpServletResponse.SC_NOT_FOUND, "login/mdp incorrect");  
    System.out.println("Recommence");  
    return;  
}
```

Apache Tomcat/8.0.27

PAGE PRINCIPALE :

Maintenant que l'entraîneur a rentré avec succès ses identifiants, il arrive directement sur une de ses équipes avec les joueurs rentrés au préalable.

ID	Nom	Status	1	2	3	4	5
2	Vincent Blot	●	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
0	Miguel Francisco	●	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
1	Anthony Chtinbach	●	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	Jean-Christophe Joubier	●	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5	Maxime Delaruelle	●	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	Danny Marechal	●	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
0	Alexis Rouillon	●	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
0	Sélim Abdelmoumen-Zeguendri	●	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Image 5 : page principale

La liste de joueur provient de la base de données grâce à cette boucle :

```
<tbody>

<%
    ArrayList<Joueur> listeJoueur = dao3.selectForTeam(listeEquipe.get(i));
    for (int j = 0; j < listeJoueur.size(); j++) {
%>

<tr id="<%=listeJoueur.get(j).getId()%>">
    <td style="width: 10%;"><%=listeJoueur.get(j).getPoste()%></td>
    <td style="width: 20%;"><%=listeJoueur.get(j).getPrenom() + " " + listeJoueur.get(j).getNom()%>
    </td>
```

Ici, la fonction « selectForTeam() » fait une requête simple à la base de données qui sert à **récupérer les joueurs en fonction de l'équipe actuellement sélectionnée**.

```
String sql = "SELECT * FROM joueur WHERE id_equipe = ?";
```

La liste de numéro à côté de chaque joueur permet **de modifier le poste du joueur et d'en avoir un aperçu visuel dynamique** sur le terrain en 2D que l'on voit à droite.

```
<%
    for (int num = 1; num <= 5; num++) {
%>

<a href="#" class="a<%=num%> btn btn-default
<%if (listeJoueur.get(j).getPoste() == num)
{
    poste[num-1] = listeJoueur.get(j);
    %> active<%
}
else if(listeJoueur.get(j).getDispo()==0){
    %> disabled<%
}%>"><%=num%></a>
```

Dans la boucle ci-contre, on ajoute la classe « active » au poste que possède le joueur (grise la case du poste). Si le joueur n'est pas disponible les postes obtiennent la classe « disabled » qui rend impossible le clic sur les postes.

DISPONIBILITE :

Le bouton de couleur rouge ou vert permet à l'entraîneur de **définir la disponibilité du joueur** pour le match prochain. Ainsi, si le bouton est rouge alors il n'est pas possible de choisir son poste parce qu'il ne participera pas au match.

```
<td><input class="dispo" type="button" style="border: 1px solid black; background-color:
<% if(listeJoueur.get(j).getDispo()==1){ %>#00FF00<% }
    else {>#FF0000 <%} %> ; border-radius:10%; height:20px; width:20px" /></td>
<td>
```

Ce « td » ci-dessus est le bouton en question. Quand la page est chargée, on regarde dans la BDD la disponibilité du joueur, si « 1 » le bouton est vert, « 0 » le bouton est rouge.

Au clic, un script JavaScript est exécuté permettant de faire ce changement de couleur en temps réel.

```
$(".dispo").click(function() {
    //rond de couleur pour la disponibilité

    //on recupere la couleur du bouton
    var couleur = $(this).css("background-color");

    //on recupere l'id du <tr> (correspond à l'id du joueur)
    var id = $(this).closest('tr').attr('id');

    if (couleur == 'rgb(0, 255, 0)') {
        //le bouton est vert
        for (var i = 1; i < 6; i++) {
            $(this).closest('tr').find(".a"+i).addClass("disabled");
        }
        $(this).closest('tr').find(".active").removeClass("active");
        $(this).css("background-color", "#FF0000");

        var dispo = 0;
    } else {
        //le bouton est rouge
        for (var i = 1; i < 6; i++) {
            $(this).closest('tr').find(".a"+i).removeClass("disabled");
        }
        $(this).css("background-color", "#00FF00");

        var dispo = 1;
    }
})
```

HAUT DE LA PAGE PRINCIPALE :



Image 6 : haut de la page principale

```
<ul class="nav nav-tabs">
  <%
    for (int i = 0; i < listeEquipe.size(); i++) {
      if (i == 0) {
        <li id="equipe<%=listeEquipe.get(i).getId()%>" class="active"><a aria-expanded="true"
          href="#<%=listeEquipe.get(i).getCategorie()%>" data-toggle="tab"><%=listeEquipe.get(i).getCategorie()%></a></li>
        <%
          } else {
            <li id="equipe<%=listeEquipe.get(i).getId()%>" class=""><a aria-expanded="false"
              href="#<%=listeEquipe.get(i).getCategorie()%>" data-toggle="tab"><%=listeEquipe.get(i).getCategorie()%></a></li>
            <%
              }
            }
  }
  <%
  </ul>
```

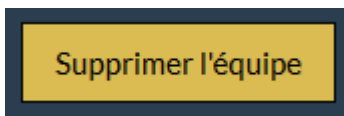
Ici, nous avons des **onglets dynamiques** qui représentent les catégories (= les équipes) que l'entraîneur a sous contrôle.

Nous récupérons les Catégories qui sont liées à l'entraîneur. Nous distinguons l' « aria-expanded » true et false signifiant l'ouverture de l'onglet : lorsque un onglet s'ouvre, l'autre se ferme.

Le bouton « Ajouter un joueur » permet à l'entraîneur, **en fonction de l'équipe dans lequel il se situe**, d'ajouter un joueur dans la liste en-dessous (image 5) et sera intégré dans la base de données.

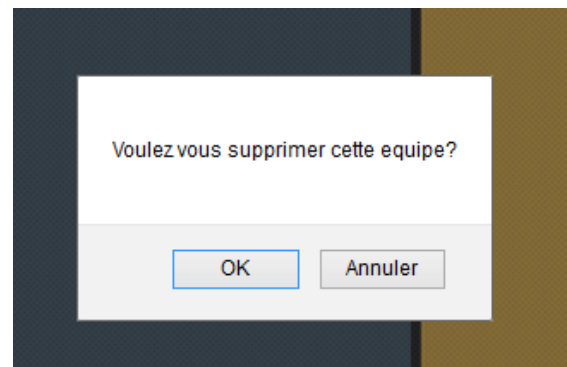
Le bouton « Ajouter une équipe » possède un champ texte qui **permet d'insérer et de créer le nom de l'équipe souhaité**. Ensuite, il faut cliquer sur le bouton à sa droite pour effectuer la requête.

Un « refresh » (rafraîchissement de page) est alors enclenché pour avoir de façon instantanée l'équipe dans les onglets.



Ce bouton, situé au-dessus du terrain 2D, permet de **supprimer l'équipe où l'entraîneur se trouve**.

Une confirmation est alors demandée pour éviter les mauvaises manipulations. De nouveau, un « refresh » est effectué pour que la suppression soit visible.



```
} else if (request.getParameter("inputEquipe") != null) {
    // formulaire ajout joueur
    String nom = request.getParameter("inputNom");
    String prenom = request.getParameter("inputPrenom");
    String date = request.getParameter("inputDate");
    String mail = request.getParameter("inputMail");
    String equipe = request.getParameter("inputEquipe");

    JoueurDAO ab = new JoueurDAO();
    Joueur j = new Joueur();
    j.setNom(nom);
    j.setPrenom(prenom);
    j.setEmail(mail);
    j.setAnneeNaissance(date);
    j.setId_equi(Integer.parseInt(equipe));
    j.setSurclas(false);

    ab.insert(j);
} else if (request.getParameter("nomEquipe") != null) {
    // formulaire ajout equipe
    String idEnt = request.getParameter("idEntraîneur");
    String nomEquipe = request.getParameter("nomEquipe");

    EquipeDAO abc = new EquipeDAO();
    Equipe e = new Equipe();
    e.setCategorie(nomEquipe);
    e.setId_entraîneur(Integer.parseInt(idEnt));

    abc.insert(e);
}
```

Pour permettre à ces formulaires de fonctionner, nous sommes obligés de passer des paramètres en méthode POST à notre Servlet qui va faire le lien avec la base de données en utilisant les fonctions classiques de DAO comme ici l'insert.

AJAX :

Pour modifier la base de données, il est nécessaire d'utiliser de l'ajax dans une fonction JavaScript. Les données sont transmises en JSON.

```
$("#a").click(function() {
    //pour tous les a (postes)

    //on enleve la classe active pour les colonnes (un poste est occupé par un seul joueur)
    $(". " + $(this)[0].className.split(' ')[0]).removeClass("active");
    //on enleve la classe active pour les lignes (un joueur ne peut pas avoir plusieurs poste
    $(this).parent().children().removeClass("active");
    //on ajoute la classe active au a sur lequel on a cliqué
    $(this).addClass("active");
    //on recupere l'id du <tr> (correspond à l'id du joueur)
    var id = $(this).closest('tr').attr('id');
    //on recupere le a qui à la classe active (correspond au poste du joueur)
    var poste = $(this)[0].className.split(' ')[0].substring(1)
    //on recupere l'id de l'équipe
    var idEquipe = getIdEquipe();

    $.ajax({
        url : 'servletppe' ,
        type : 'GET',
        dataType : 'json',

        data : {
            id : id,
            poste : poste,
            idEquipe : idEquipe
        },
        success: function(data) {},
        error: function(data){}
    });
});
```

La Servlet récupère les données envoyées par l'ajax pour effectuer la mise à jour grâce à la fonction ci-dessous. (N.B. : La fonction n'est pas complète en photo, ceci est juste l'essentiel).

```
if(request.getParameter("poste") != null){
    //mise à jour poste joueur

    int id = Integer.parseInt(request.getParameter("id"));
    int poste = Integer.parseInt(request.getParameter("poste"));
    int idEquipe = Integer.parseInt(request.getParameter("idEquipe"));

    Joueur j = new Joueur();
    j.setId(id);
    j.setId_equi(idEquipe);
    j.setPoste(poste);

    System.out.println(j);

    JoueurDAO dao = new JoueurDAO();
    dao.updatePoste(j);
}

getConnection();
conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_UPDATABLE);

//on met tous les joueurs de l'équipe avec ca poste à 0
String sql = "UPDATE joueur SET poste=0 WHERE id_equipe=? AND poste =?";
prepare = conn.prepareStatement(sql); // on créer la requete
prepare.setInt(1, obj.getId_equi());
prepare.setInt(2, obj.getPoste());

prepare.execute(); // on execute la requete

//on met à jour le poste du joueur
sql = "UPDATE joueur SET poste=? WHERE id=?";
prepare = conn.prepareStatement(sql); // on créer la requete
prepare.setInt(1, obj.getPoste());
prepare.setInt(2, obj.getId());

prepare.execute();
```