Département d'Informatique
Université de Fribourg, Suisse
http://diuf.unifr.ch

UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

AES : Site proposant des activités extra-scolaires

Un projet réalisé avec PHP/MySQL

Martin Masoni
No étudiant : 14-200-984

Travail de séminaire en Informatique de Gestion
Encadré par :
Prof. Dr. Jacques Pasquier – Rocha
Nadia Aebischer

Software Engineering Group

Fribourg, Octobre 2016

Table des matières

1 Introduction 4

2 Domaine d'application 5

3 Modélisation des données 6

3.1 Modèle entité-association 6

3.2 Transformation en tables relationnelles 8

3.3 Les tables 9

3.3.1 Personne 9

3.3.2 Adulte 9

3.3.3 Enfant 9

3.3.4 Type 10

3.3.5 Activité 10

3.3.6 Organise 10

3.3.7 Participe 11

4 Le site d'un point de vue de l'utilisateur final et éléments de programmation 12

4.1 Le rendu visuel 12

4.2 Les fonctions 14

5 Conclusion 19

A Fichier .zip du code source 20

Bibliographie 21

Liste des figures

Figure 1 Modèle entité-association 6
Figure 2 Diagramme relationnel 8
Figure 3 Table personne 9
Figure 4 Table adulte 9
Figure 5 Table enfant 9
Figure 6 Table type 10
Figure 7 Table activite 10
Figure 8 Table organise 10
Figure 9 Table participe 11
Figure 10 Page 1 12
Figure 11 Page 2 13
Figure 12 Page 3 13

Liste des codes

Code 1: Code du menu déroulant de la page 1 14
Code 2: Code pour la création du titre et tableau de la page 2 16
Code 3: Code pour la création tableau de la page 3 18

1
Introduction

Une grande partie de l'information est actuellement numérisée offrant l'énorme avantage de pouvoir la dupliquer indéfiniment, rendue facilement accessible à tous et sans frais de gestion. La grande majorité des entreprises, organisations et même nations ont trouvé dans cette technologie un moyen efficace de gérer, administrer ou tout simplement communiquer. L'omniprésence et la diversification des supports informatiques ont accru l'utilisation de ces nouvelles technologies. Ils nous ont aussi amené à harmoniser la manière dont nous nous informons. En conséquence, les personnes d'aujourd'hui se renseignent essentiellement sur internet.

Ce travail de séminaire tente de mettre au goût du jour certaines habitudes relatives à la diffusion d'informations. J'ai donc choisi de prendre l'exemple d'une petite association dans laquelle des adultes organisent des activités extra-scolaires pour des enfants. Ayant moi-même fait partie de sociétés similaires dans mon enfance, j'ai souhaité mettre à disposition un outil qui faciliterait l'organisation et ceci en toute transparence. Par la même occasion, il pourrait permettre de minimiser les frais liés à l'envoi de programmes ou de fiches informatives, et ainsi d'allouer ces ressources à d'autre buts.

Le *chapitre 2* de ce rapport exposera les raisons et les objectifs de la plateforme. Le *chapitre 3* quant à lui sera consacré à la modélisation des données, à savoir une présentation du modèle entité-association et sa traduction en un schéma relationnel. Le *chapitre 4* présentera le site du point de vue de l'utilisateur final avec des commentaires sur la programmation sous-jacente.

2

Domaine d'application

L'objet choisi pour ce travail est une petite association proposant des activités extra-scolaires à des enfants. Les associations sont pour la plupart créées et gérées par des bénévoles. Ces personnes consacrent une part de leur temps libre à développer une harmonie sociale au sein d'une collectivité. Il est intéressant de voir quelles possibilités d'améliorations cette technologie propose concernant certains points de la gestion administrative d'une petite organisation. L'informatisation d'une partie de la documentation pourrait déjà améliorer plusieurs facteurs, tels que le temps investi par les bénévoles pour distribuer l'information à tous, l'énergie que cela implique et les coûts qui en résultent.

Dans notre cas, cette plateforme a pour but de mettre à disposition toute une palette d'activités destinées aux enfants. Elle est aussi conçue pour permettre aux parents de trouver toutes les informations relatives aux activités proposées, avec le détail des dates, des horaires et des lieux correspondants. Dans les différentes colonnes seront annotées les coordonnées des organisateurs afin de pouvoir les contacter par email. Sur cette plateforme figurera également une liste des participants. Cette dernière permettra aux accompagnants une meilleure organisation et une connaissance des infrastructures à mettre en place en fonction du nombre d'inscriptions. La liste des participants servira également aux enfants qui souhaiteraient connaître quels amis seront présents pour tel événement. Dans la rubrique des enfants inscrits seront annotées les coordonnées téléphoniques des parents à contacter, ceci en cas de besoin.

Il va sans dire que ce système pourrait être modifié et utilisé pour tout autre type d'organisations similaires.

3

Modélisation des données

Ce chapitre présente la structure de la base de données *acexsc*. Dans un premier temps, j'expose les diverses relations qui unissent chacune des tables. Puis, ces mêmes tables seront détaillées individuellement afin de vérifier que chaque colonne qui les compose aura été définie par un attribut spécifique. Certains de ces attributs serviront ainsi de clés primaires et étrangères afin de relier les tables entre elles.

3.1 Modèle entité-association

La création d'un modèle entité-association a été la première étape pratique de mon travail.

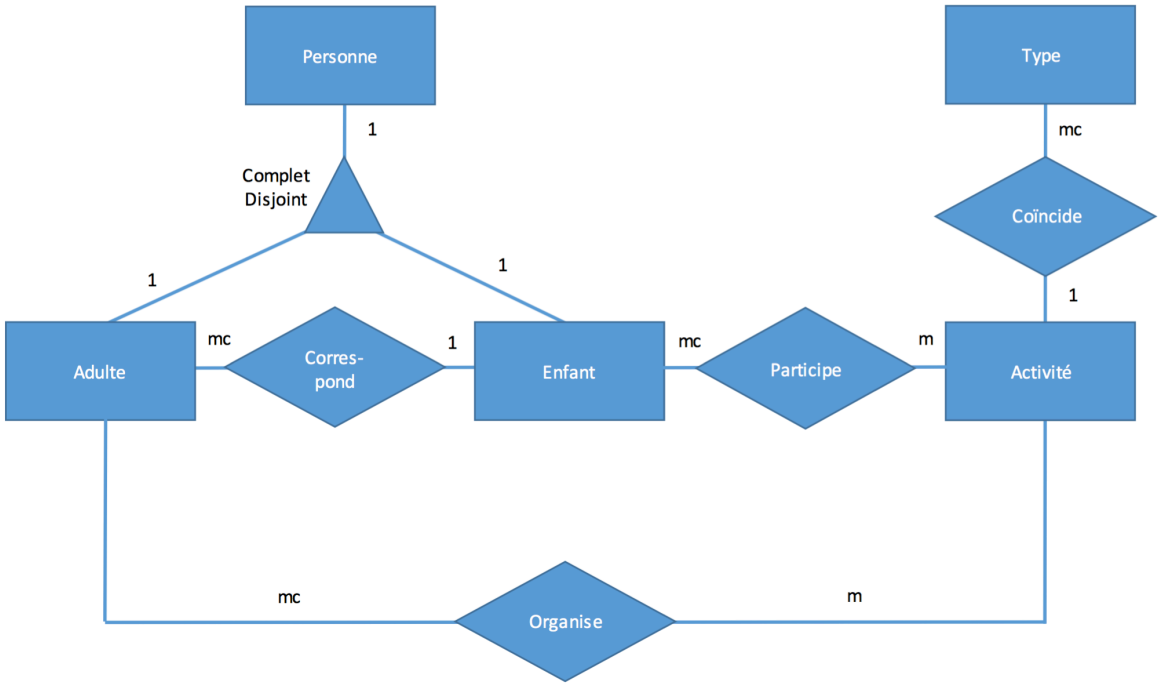

Figure 1 Modèle entité-association

Je l'ai réalisée avec Microsoft Word selon le standard présenté par Andreas Meyer[1]. Ce schéma peut paraître simpliste, mais il est indispensable. Il m'a permis d'avoir une vue globale de ma base de données *asexsc* et ceci tout au long de mon travail. Je m'y suis fié à chaque étape afin de m'assurer d'être cohérent et de rester ainsi sur le bon chemin.

[1] Selon le livre d'Andreas Meier, *Introduction pratique aux bases de données relationnelles*, Springer, 2006.

Voici une interprétation textuelle de mon modèle :

- Une personne est soit un enfant, soit un adulte.
- Un enfant participe à plusieurs ou aucune activité.
- Un adulte organise plusieurs ou aucune activité.
- Un enfant a forcément un adulte comme parent.
- Un adulte peut avoir plusieurs ou aucun enfant.
- Une activité a au moins un enfant et un adulte.
- Un type peut avoir plusieurs ou aucune activité.
- Une activité a forcément un type.

Nous constatons que la base de données nécessite 5 tables. Toutefois, elle nécessite 2 tables supplémentaires pour reproduire les relations dites multiples-multiples *participe* et *organise*.

3.2 Transformation en tables relationnelles

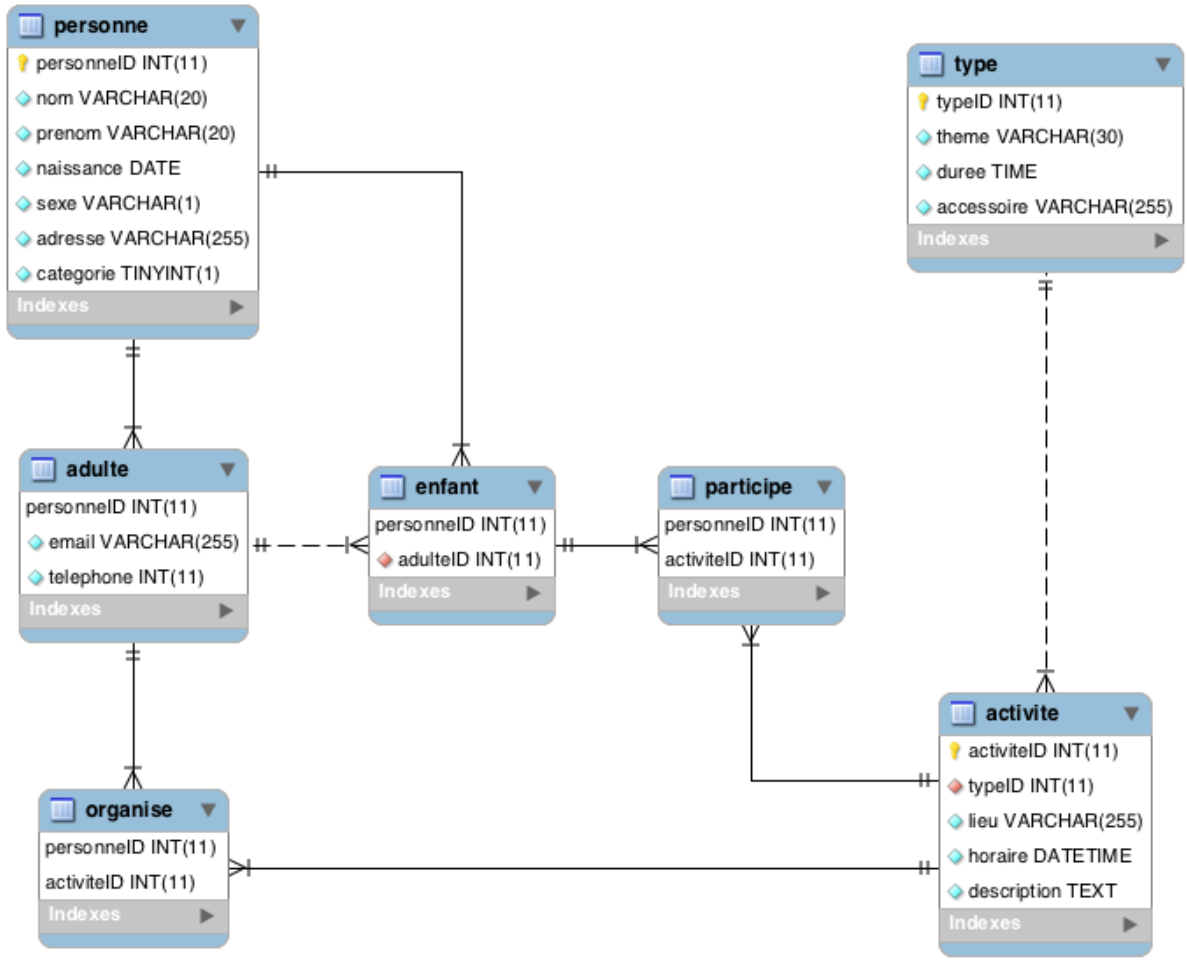

Figure 2 Diagramme relationnel

Afin de convertir le modèle précédent (voir Figure 1) en un diagramme relationnel plus détaillé, j'ai utilisé MySQLWorkbench[2]. Ce logiciel possède un outil spécifique pour importer les attributs contenus dans ma base de données et produire un schéma de manière automatique. Pour disposer de ce résultat, il m'a été nécessaire dans un premier temps de créer mes tables sur phpMYAdmin[3] et de les nommer.

Chaque table est composée d'une clé primaire, d'attributs et parfois aussi de clés étrangères. Une clé primaire est généralement auto incrémentée et permet de définir de manière unique une entrée et ainsi éviter toutes confusions lors de nos futures requêtes. Cette clé primaire peut certaines fois être une clé étrangère qui renvoie vers une autre table ascendante. Dans mes tables

---

[2] Téléchargement de MySQLWorkbench, https://dev.mysql.com/downloads/workbench/, dernière visite: 10.10.2016.

[3] Téléchargement de MAMP pour utiliser phpMYAdmin, https://www.mamp.info/en/, dernière visite: 10.10.2016.

à relations multiples-multiples tel que *organise* et *participe*, leurs clés primaires sont directement le résultat composé de deux clés étrangères.

3.3 Les tables

3.3.1 Personne

	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
☐	1	**personneID**	int(11)			Non	*Aucune*		AUTO_INCREMENT	Modifier Supprimer plus
☐	2	**nom**	varchar(20)	utf8_general_ci		Non	*Aucune*			Modifier Supprimer plus
☐	3	**prenom**	varchar(20)	utf8_general_ci		Non	*Aucune*			Modifier Supprimer plus
☐	4	**naissance**	date			Non	*Aucune*			Modifier Supprimer plus
☐	5	**sexe**	varchar(1)	utf8_general_ci		Non	*Aucune*			Modifier Supprimer plus
☐	6	**adresse**	varchar(255)	utf8_general_ci		Non	*Aucune*			Modifier Supprimer plus
☐	7	**categorie**	tinyint(1)			Non	*Aucune*			Modifier Supprimer plus

Figure 3 Table personne

Pour éviter de dupliquer des attributs communs aux tables *adulte* et *enfant*, j'ai décidé de créer une table *personne* afin de les regrouper. La clé primaire *personneID* identifie chaque personne de façon unique. Cela permet, plus tard, de collecter les diverses informations relatives à chacune d'elle. Pour mon projet, j'ai tout de même créé un attribut *catégorie* pour différencier les classes d'âge. Il n'est pas indispensable pour l'instant, mais le serait dans le cas où je créerais une interface nécessitant un login. L'*adresse*, quant à elle, aurait pu également être déplacée dans une table expressément créée pour l'occasion, avec des attributs plus précis.

3.3.2 Adulte

	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
☐	1	**personneID**	int(11)			Non	*Aucune*			Modifier Supprimer Primaire plus
☐	2	**email**	varchar(60)	utf8_general_ci		Non	*Aucune*			Modifier Supprimer Primaire plus
☐	3	**telephone**	int(11)			Non	*Aucune*			Modifier Supprimer Primaire plus

Figure 4 *Table adulte*

Dans cette table *adulte*, je référence les personnes ayant plus de 18 ans qui sont soit des parents d'enfants, soit des organisateurs. Ici, la clé primaire *personneID* se trouve être une clé étrangère reliant à la table *adulte* à la table *personne*. Dans cette table, j'ai profité de rajouter des attributs qui n'étaient pas essentiels dans le cas d'un enfant, tels que l'*email* ou le *téléphone*.

3.3.3 Enfant

	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
☐	1	**personneID**	int(11)			Non	*Aucune*			Modifier Supprimer Primaire plus
☐	2	**adulteID**	int(11)			Non	*Aucune*			Modifier Supprimer Primaire plus

Figure 5 *Table enfant*

Pour cette nouvelle table *enfant*, j'ai à nouveau utilisé la clé primaire *personneID* qui n'est autre qu'une clé étrangère renvoyant à la table *personne*. L'information supplémentaire qui m'intéressait ici était celle d'une personne de contact majeure pour chaque enfant. Au lieu de récrire toutes ces données, j'ai profité de ne relier qu'un attribut à la table *adulte*. J'ai donc

utilisé une clé étrangère *adulteID* qui n'est autre que la clé primaire *personneID* de la table *adulte*. Je peux ainsi récolter tous les éléments utiles tels que le téléphone des représentants correspondant à chaque enfant.

3.3.4 Type

	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action		
☐	1	**typeID**	int(11)			Non	*Aucune*		AUTO_INCREMENT	Modifier	Supprimer	plus
☐	2	**theme**	varchar(30)	utf8_general_ci		Non	*Aucune*			Modifier	Supprimer	plus
☐	3	**duree**	time			Non	*Aucune*			Modifier	Supprimer	plus
☐	4	**accessoire**	varchar(255)	utf8_general_ci		Non	*Aucune*			Modifier	Supprimer	plus

Figure 6 *Table type*

Dans la table *type*, nous trouvons tous les attributs généraux relatifs à toutes les activités du même thème. Afin de conserver une clé primaire *typeID* dotée d'une incrémentation automatique, j'ai créé un second attribut *theme*. Lui aussi est unique, mais je souhaitais éviter d'avoir une clé primaire de caractère varchar.

3.3.5 Activité

	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action		
☐	1	**activiteID**	int(11)			Non	*Aucune*		AUTO_INCREMENT	Modifier	Supprimer	plus
☐	2	**typeID**	int(11)			Non	*Aucune*			Modifier	Supprimer	plus
☐	3	**lieu**	varchar(255)	utf8_general_ci		Non	*Aucune*			Modifier	Supprimer	plus
☐	4	**horaire**	datetime			Non	*Aucune*			Modifier	Supprimer	plus
☐	5	**description**	text	utf8_general_ci		Non	*Aucune*			Modifier	Supprimer	plus

Figure 7 *Table activite*

Pour la table *activite*, j'ai défini toutes les informations correspondantes à chacune des activités. La clé primaire *activiteID* identifie spécifiquement chacune d'elle. J'ai rajouté une clé étrangère *typeID* qui joint la table *type* pour compléter mes données. A nouveau, il aurait été possible de créer une table supplémentaire pour l'attribut *lieu*, qui aurait pu être utilisé communément avec *adresse* de la table *personne*.

3.3.6 Organise

	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action		
☐	1	**personneID**	int(11)			Non	*Aucune*			Modifier	Supprimer	Primaire plus
☐	2	**activiteID**	int(11)			Non	*Aucune*			Modifier	Supprimer	Primaire plus

Figure 8 *Table organise*

Cette table-ci à la particularité d'être le fruit de la relation multiples-multiples entre les tables *adulte* et *activite*. Ce type de lien impose donc la création d'une table supplémentaire. La clé primaire est une clé primaire composée de deux clés étrangères, *personneID* provenant de la table *adulte* et *activiteID* originaire de la table *activite*.

3.3.7 Participe

	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action		
☐	1	**personneID**	int(11)			Non	*Aucune*			Modifier	Supprimer	Primaire plus
☐	2	**activiteID**	int(11)			Non	*Aucune*			Modifier	Supprimer	Primaire plus

Figure 9 *Table participe*

A nouveau, cette table représente une relation multiples-multiples, mais cette fois-ci entre les tables *enfant* et *activite*. Au premier abord, elle paraît identique à la table *organise*, mais la clé étrangère *personneID* ne fait pas référence à la même table. Sa clé primaire est une fois de plus une clé composée de clés étrangères, *personneID* provenant de la table *enfant* et *activiteID* originaire de la table *activite*.

4

Le site d'un point de vue de l'utilisateur final et éléments de programmation

Précédemment, je vous ai présenté la structure de ma base de données *asexsc*, ainsi que ses différentes tables. Il était dès lors possible de procéder à des recherches spécifiques à l'aide de requêtes SQL. Cependant, l'interface reste très complexe et pas à la portée de tout le monde pour une utilisation ordinaire. Dans ce chapitre-ci j'ai donc créé une interface web en HTML incrémentée de fonctions PHP.

4.1 Le rendu visuel

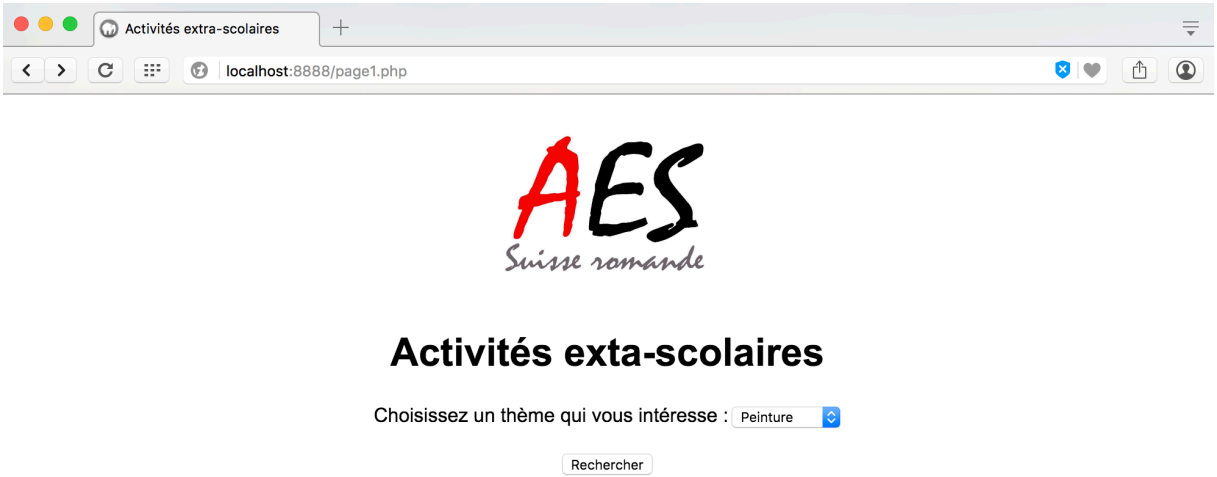

Figure 10 *Page 1*

Cette première page d'accueil permet, à l'aide d'un menu déroulant, de sélectionner un des différents thèmes d'activité qui suscite l'intérêt de l'utilisateur. Une fois le choix effectué, il suffit de cliquer sur le bouton *Rechercher* pour générer la page correspondante.

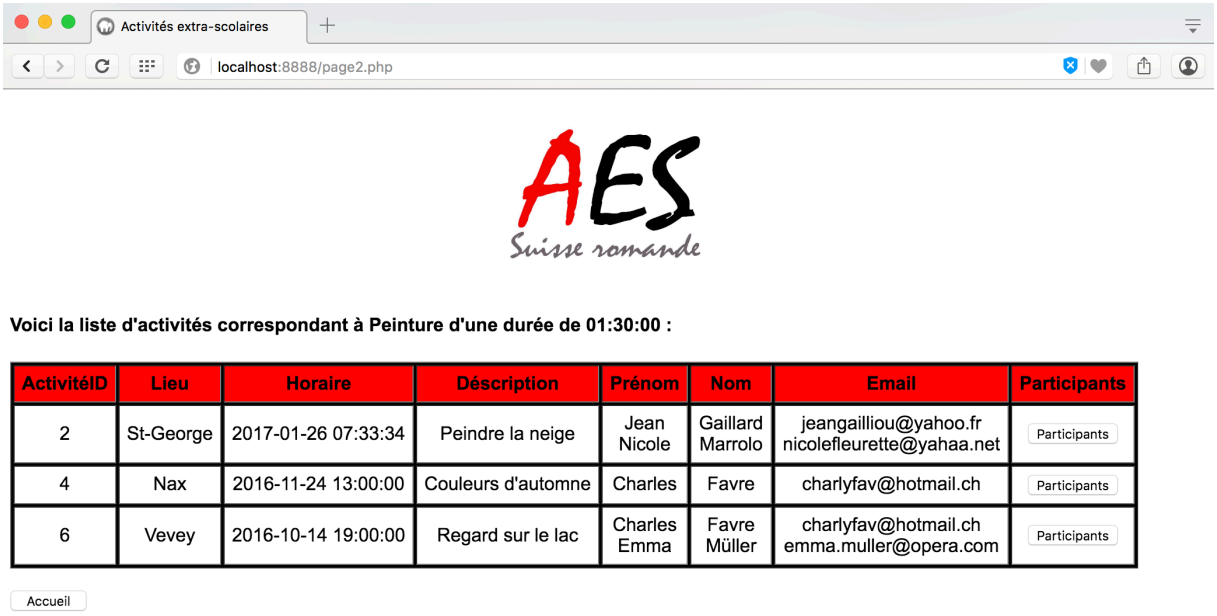

Figure 11 *Page 2*

La deuxième page donne une vue d'ensemble sur toutes les activités. Il y est présenté seulement les colonnes essentielles. Afin de ne pas avoir trop de colonnes sur la même page, j'ai décidé de créer une troisième page. En cliquant sur *Participants*, nous accédons à une liste plus détaillée.

Pour la même raison, j'ai aussi directement inséminé dans le titre du tableau les attributs *theme* et *duree*, ainsi j'évite d'avoir des colonnes supplémentaires avec des cellules qui répètent inutilement la même information.

Figure 12 *Page 3*

Dans cette troisième et dernière page, je liste les participants ainsi que leurs représentants légaux. Au fond de la page se trouve un bouton permettant de retourner sur la page d'accueil.

4.2 Les fonctions

Comme expliqué en début de chapitre, nous voulons rendre les données accessibles aux utilisateurs d'une manière simplifiée et personnalisée. Il s'agit, pour l'usager, d'accéder depuis son navigateur à certaines informations stockées dans la base de données.

Malheureusement le code HTML est insuffisant. Pour cela, il est nécessaire de créer des fonctions PHP. Elles permettent d'interagir avec la base de données. Lorsqu'un client demande une certaine page, le navigateur envoie une requête au serveur qui, lui, détectera si un script PHP est intégré. Dans ce cas présent, le serveur traitera le script et renverra au navigateur le résultat au format HTML. Il pourra enfin l'interpréter et afficher une page web.

```
1   <?php
2   $bdd=new PDO('mysql:host=localhost;dbname=acexsc;charset=utf8',
      'root', 'root');
3   $reponse = $bdd->query('SELECT typeID, theme FROM type') ;
4
5   while ($donnees = $reponse->fetch())
6   {
7   echo '<option value="'. $donnees['typeID'] . '">'.
      $donnees['theme'] .'</option>';
8   }
9   ?>
```

Code 1: *Code du menu déroulant de la page 1*

Ce code est tiré de la première page web. C'est une fonction PHP qui génère automatiquement un menu déroulant en allant puiser tous les thèmes proposés dans la base de données. J'avais d'abord écrit manuellement ces données cependant ce n'était pas commode dans le cas où un thème viendrait à se rajouter.

Nous pouvons observer que la fonction est composée de trois parties principales. A la ligne 2, nous nous connectons à la base de données en indiquant son nom et son accès. Suit une requête SQL qui va trier les données souhaitées pour les stocker dans une variable *$donnees*. A la ligne 7, nous avons un *echo* qui va permettre d'imprimer le résultat demandé. Quant à la boucle *while,* elle va réitérer l'opération jusqu'à ce que toutes les données sélectionnées aient été traitées.

```
1     <table border=1 cellpadding=6 bgcolor=black style="text-
      align:center">
2
3     <!-- les entêtes de colonnes -->
4     <tr bgcolor="red">
5         <th>ActivitéID </th>
6         <th>Lieu</th>
7         <th>Horaire</th>
8         <th>Déscription</th>
9         <th>Prénom </th>
10        <th>Nom</th>
11        <th>Email</th>
12        <th>Participants</th>
13    </tr>
14
15    <?php
16    $bdd=new PDO('mysql:host=localhost;dbname=acexsc;charset=utf8',
      'root', 'root');
17
18    //requête pour remplir les cellules de ma table
19    $requete=$bdd->prepare('SELECT activite.activiteID,
      description, theme, duree, lieu, GROUP_CONCAT(prenom SEPARATOR
      "<br/>"), horaire, GROUP_CONCAT(nom SEPARATOR "<br/>"),
      GROUP_CONCAT(email SEPARATOR "<br/>")
20        FROM type
21              JOIN activite ON type.typeID=activite.typeID
22                 JOIN organise ON
      activite.activiteID=organise.activiteID
23                    JOIN adulte ON
      organise.personneID=adulte.personneID
24                        JOIN personne ON
      adulte.personneID=personne.personneID
25                    WHERE type.typeID=?
26                    GROUP BY activite.activiteID');
27
28    //requête2 pour remplir les informations du titre
29    $requete2 = $bdd->prepare('SELECT typeID, theme, duree FROM
      type WHERE type.typeID=?');
30
31    $requete2->execute(array($_GET['typeID']));
32
33    $donnees2 = $requete2->fetch();
34    echo '<h4> Voici la liste d\'activités correspondant à '.
      $donnees2['theme'] . ' d\'une durée de '. $donnees2['duree'] .
      ' : </h4> ';
35
36    $requete->execute(array($_GET['typeID']));
37
38    while ($donnees = $requete->fetch())
39    {
40
41      echo ' <tr bgcolor=white> <td>'. $donnees['activiteID']
      .'</td>
42                          <td>'.$donnees['lieu'].'</td>
43                          <td>'.$donnees['horaire'].'</td>
44                          <td>'.$donnees['description'].'</td>
45                          <td>'.$donnees['GROUP_CONCAT(prenom
      SEPARATOR "<br/>")'] .' </td>
```

```
46                          <td>'.$donnees['GROUP_CONCAT(nom SEPARATOR
    "<br/>")'].'</td>
47                          <td>'.$donnees['GROUP_CONCAT(email
    SEPARATOR "<br/>")'].'</td>
48              <td> <button
    onclick=location.href="http://localhost:8888/page3.php?activite
    ID='. $donnees['activiteID'] .'"> Participants </button>
    </td></tr>' ;
49    }
50
51    ?>
52    </table>
```

Code 2: *Code pour la création du titre et tableau de la page 2*

Ce code-ci présente les deux requêtes de la deuxième page. L'une permet de générer un titre personnalisé en fonction du choix de la page précédente, l'autre de créer un tableau avec les données récoltées. J'ai utilisé la méthode `GET` afin de récupérer le *typeID* de la page 1 et de trier ma sélection. Vous remarquerez que j'ai dû spécifier, pour les variables relatives aux organisateurs, un `GROUP_CONCAT()` pour regrouper plusieurs données dans la même cellule. Afin d'optimiser la lisibilité, j'ai aussi dû insérer un `SEPARATOR "<br/>"`, car la séparation par défaut était une virgule et non un retour à la ligne.

```
1     <?php
2     $bdd=new  PDO('mysql:host=localhost;dbname=acexsc;charset=utf8',
      'root', 'root');
3     //requête pour remplir les cellules se rapportant à la table
      enfant
4     $requete=$bdd->prepare('SELECT prenom, nom, YEAR(CURRENT_DATE)-
      YEAR(naissance), adulteID
5         FROM type
6               JOIN activite ON type.typeID=activite.typeID
7                 JOIN                  participe                  ON
      activite.activiteID=participe.activiteID
8                     JOIN                  enfant                 ON
      participe.personneID=enfant.personneID
9                       JOIN               personne               ON
      enfant.personneID=personne.personneID
10    WHERE activite.activiteID=?');
11
12    //requête2 pour remplir les informations du titre
13    $requete2 = $bdd->prepare('SELECT theme FROM type
14          JOIN activite ON type.typeID=activite.typeID
15            WHERE activite.activiteID=?
16                    GROUP BY activite.activiteID');
17    //requête pour remplir les cellules se rapportant à la table
      adulte
18    $requete3=$bdd->prepare('SELECT prenom, nom, telephone
19      FROM activite
20                JOIN                  participe                  ON
      activite.activiteID=participe.activiteID
21                    JOIN                  enfant                 ON
      participe.personneID=enfant.personneID
22                      JOIN                adulte                 ON
      enfant.adulteID=adulte.personneID
23                        JOIN              personne               ON
      adulte.personneID=personne.personneID
24
25                        WHERE enfant.adulteID=?');
26
27
28
29    $requete2->execute(array($_GET['activiteID']));
30
31    $donnees2=$requete2->fetch();
32    echo '<h4> Voici la liste de participants pour l\'activité '.
      $donnees2['theme'] . ': </h4> ';
33
34
35
36    $requete3->execute(array($donnees['adulteID']));
37
38
39
40
41    $requete->execute(array($_GET['activiteID']));
42    while ($donnees = $requete->fetch())
43    {
44
45      echo   '<tr bgcolor=white><td>'. $donnees['prenom'] .' </td>
46                                    <td>'. $donnees['nom'] .'</td>
```

```
47                                      <td>'.
      $donnees['YEAR(CURRENT_DATE)-YEAR(naissance)'] .'</td>';
48              $requete3->execute(array($donnees['adulteID']));
49
50              $donnees3 = $requete3->fetch();
51
52              echo '              <td>'. $donnees3['prenom'] . '</td>
53                                   <td>'. $donnees3['nom'] . '</td>
54                                   <td>'.   $donnees3['telephone']   .
      '</td></tr>';
55
56    }
57
58    ?>
```

Code 3: *Code pour la création tableau de la page 3*

Dans cette dernière page, j'ai dû résoudre la problématique liée à la relation complète disjointe de la table *personne*. Effectivement, je souhaitais avoir un tableau exposant des informations sur des enfants et des adultes. Lors de la requête, si je sélectionnais l'attribut *prenom*, il était impossible de dissocier ceux des enfants de ceux des adultes. Pour ce faire j'ai dû créer deux requêtes différentes.

5
Conclusion

Au terme de ce travail, mon site permet d'afficher des données en fonction du choix réalisé par un utilisateur. Nous avons remarqué que l'unique emploi du langage HTML ne permet pas de personnaliser une page web. Il ne fait que montrer une vitrine prédéfinie. Le recourt à PHP donne la possibilité d'avoir une interface interactive avec l'utilisateur en allant rechercher l'information stockée dans une base de données, la trier et la renvoyer.

Ce site web reste toute de même sommaire pour un usage efficace et plus poussé. Il ne présente pas assez de fonctionnalités. Un système de login permettrait d'aller plus loin dans cette démarche, il offrirait par exemple la possibilité de créer ou de s'inscrire à une activité. Ceci donnerait une plus-value à la plateforme.

En résumé, ce travail m'a montré que les bases de cette technologie omniprésente restent relativement accessibles à toutes et tous. Cependant, pour créer des projets plus ambitieux et de ce fait plus complexes, il faudrait approfondir davantage les langages abordés ici et en apprendre de nouveaux.

A

Fichier .zip du code source

Le fichier .zip contient cinq documents :

- locoalhost.sql : la base de données *asexsc* à ouvrir à l'aide de phpMYAdmin.
- page1.php : la page d'accueil
- page2.php : la page regroupant les diverses activités
- page3.php : la page listant les participants et leur personne de contact
- rapport.pdf : le rapport au format PDF

Bibliographie

[Meier, 2006]

A. Meier, *Introduction pratique aux bases de données relationnelles*, Springer, 2006.

[Pasquier, Albreshne, 2014]

J. Pasquier, A. Albreshne, *Informatique de gestion I*, Université de Fribourg, 2014.

[Collaud, 2015]

G. Collaud, *Informatique de gestion II*, Université de Fribourg, 2015.

[Openclassrooms, 2016]

Concevez votre site avec PHP et MySQL, https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql, Dernière visite: 10.10.2016

[MySQL, 2016]

Aggregate (GROUP BY) function descritpions, https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html, [Dernière visite: 10.10.2016].

[MAMP, 2016]

Download MAMP: My Apache - MySQL - PHP, https://www.mamp.info/en/, [Dernière visite: 10.10.2016].

[MySQLWorkbench, 2016]

Download MySQLWorkbench, https://dev.mysql.com/downloads/workbench/, [Dernière visite: 10.10.2016].