PROJET INFORMATIQUE PSI

Aurore DAUPHIN
Mouhamed SEYDI

UNIVERSITE D'ORLEANS
COLLEGIUM
Sciences & Techniques
Orléans - Bourges - Chartres
miage

Sommaire

1 Introduction

Pour ce projet de Programmation des Systèmes d'Information. Notre mission est la suite du projet du premier semestre qui consistait en une partie analyse, et maquetage. Cette fois si, nous devons mettre en application notre travail initial et créer l'application web proprement dite. Pour se faire nous devons utiliser le frameworks PHP Symphony 2, qui est une boite à outils de composants logiciels pour créer la structure d'une application, ainsi que ses grandes lignes, mais également organiser le code.

Nous avons également réalisé une partie Base de Données, dans le but de l'intégrer au site web, qui contiendra alors du PHP, et qui deviendra de ce fait dynamique. Pour ce faire, nous avons, à partir des différents MCD, généré les MLDR associés, puis dans la continuité le script MySQL qui en découle.

2 Présentation du projet

2.1 Expression des besoins

Le périmètre d'étude inclut les domaines d'activités de l'association suivants :
La gestion de la pré-saison et l'édition des maquettes de flyers.
La gestion des inscriptions des adhérents aux cours de danse incluant l'adhésion à l'association.
La gestion des inscriptions aux stages organisés par l'association et les soirées.
Ceci est la deuxième partie du projet et qui consiste en la réalisation du système (proposé au premier semestre) sous la forme d'un site web responsive en PHP

2.2 But et objectifs

Suite à l'analyse et la conception du Système de Gestion d'une Association de danse, il vous est demandé de développer une application Web concrétisant la solution retenue.

2.3 Références des ressources Web et bibliographies

Les sites Web :

Open Classroom Symfony
www.w3schools.com
getbootstrap.com
Symfony.com
stackoverflow.com

Les tutoriels :

Bootsrtap de WebAcappellaTuto, Coder's Guide,
Symfony de DevAndClick

2.4 Outils et environnements utilisés (AGL, framwork, etc.)

Logiciels utilisés :

Win'Design 14.1.2	Photoshop 7.0
HTML 5	CSS 3
Notepad++	Sublimtext
Opéra	Google Chrome
Firefox	Bootsrtap
Wamp	Git
Symfony	Open Office
netbeans	

Pour les tests du site : Opera, Internet explorer 11, Chrome et Firefox.

3 Conduite du projet

3.1 Découpage du domaine d'étude en processus

Nous avons décidé de découper le domaine d'étude en deux processus principaux, pour deux membres de groupe, le premier englobe tout ce qui touche à Symfony, et le deuxième regroupe tout le reste.

3.2 Répartition des tâches entre les membres du groupe

Nous avons décidé de nous répartir les tâches, en suivant nos domaines de compétences de chacun, Mouhamed a eu à sa charge la réalisation de la programmation en Symfony et Aurore, la responsivité du site, les diagrammes et la base de donnée.

3.3 Planning (prévisionnel et effectif)

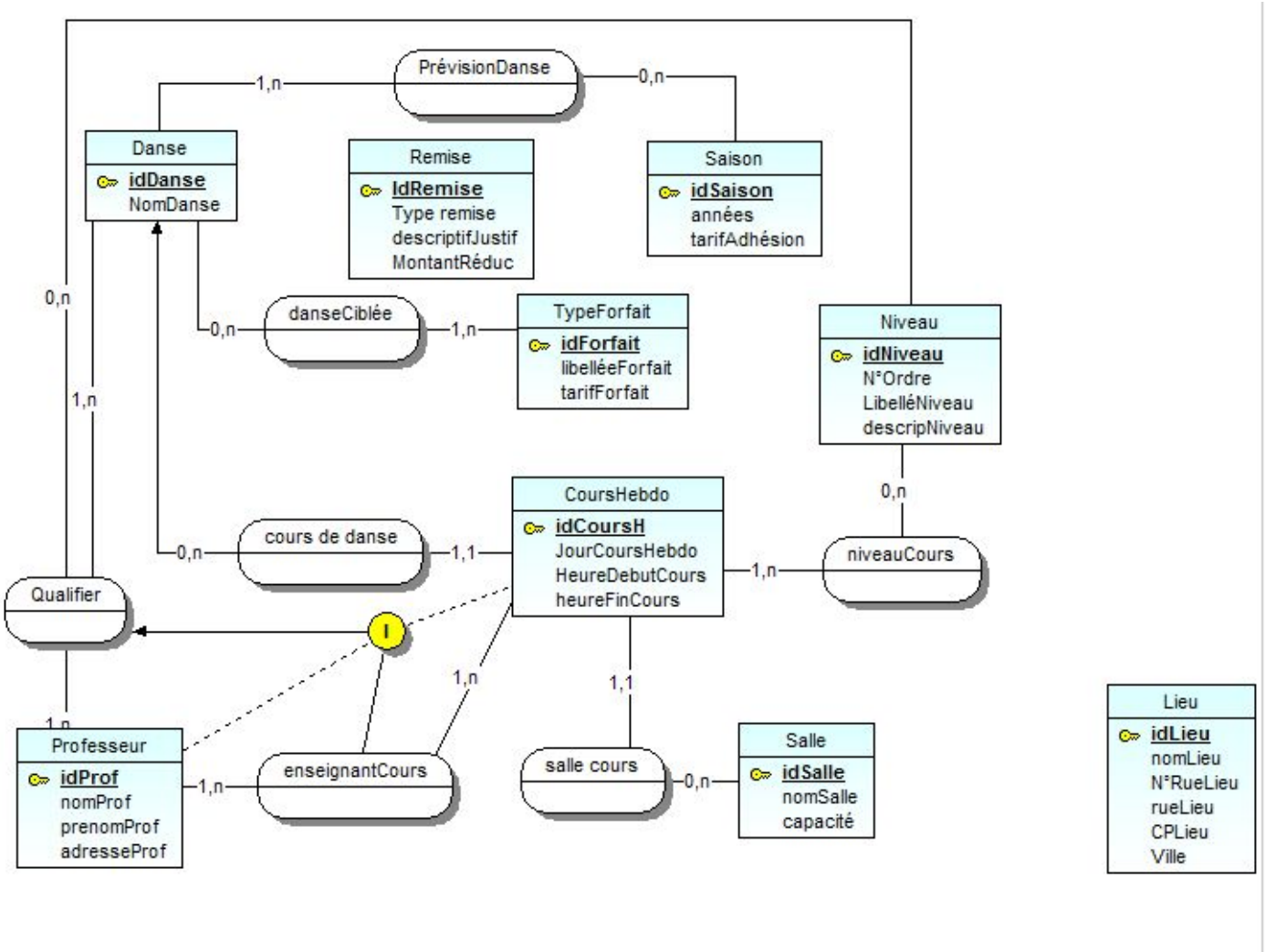
Quand	Qui	Fait	A faire	Prochaine date convenu
05/01/16	*Groupe*		Autoformation Symfony et bootstrap	*février*
Février	*Groupe*	Correction du MCD	Autoformation Symfony et bootstrap	*02/02/16*
02/02/16	*Groupe*	Point sur les avancements des autoformations, mise en avants des difficultés, décision du MCD à adopter	Continuer les autoformations, mise au propre du MCD adopté	*02/03/16*
01/03/16	*Mickaël*	Démission de la formation	Récupérer toutes les informations et documents utile à la suite du projet	
02/03/16	*Mouhamed*	Commencer la création sous Symfony	Symfony	*10/03/16*
	Aurore	réalisation des MCD finaux, génération du MLDR et des scripts PHP	Créer le jeu de données, rendre le site responsive	
10/03/16	*Mouhamed*	Symfony	Symfony	*22/03/16*
	Aurore	jeu de données, rendre le site responsive	Refaire les MCD à partir de la diffusion de la correction, MLDR, script , bootstrap, jeu de données, le dossier	
22/03/16	*Groupe*	Finalisation du projet		

4 Spécifications fonctionnelles de la solution retenue

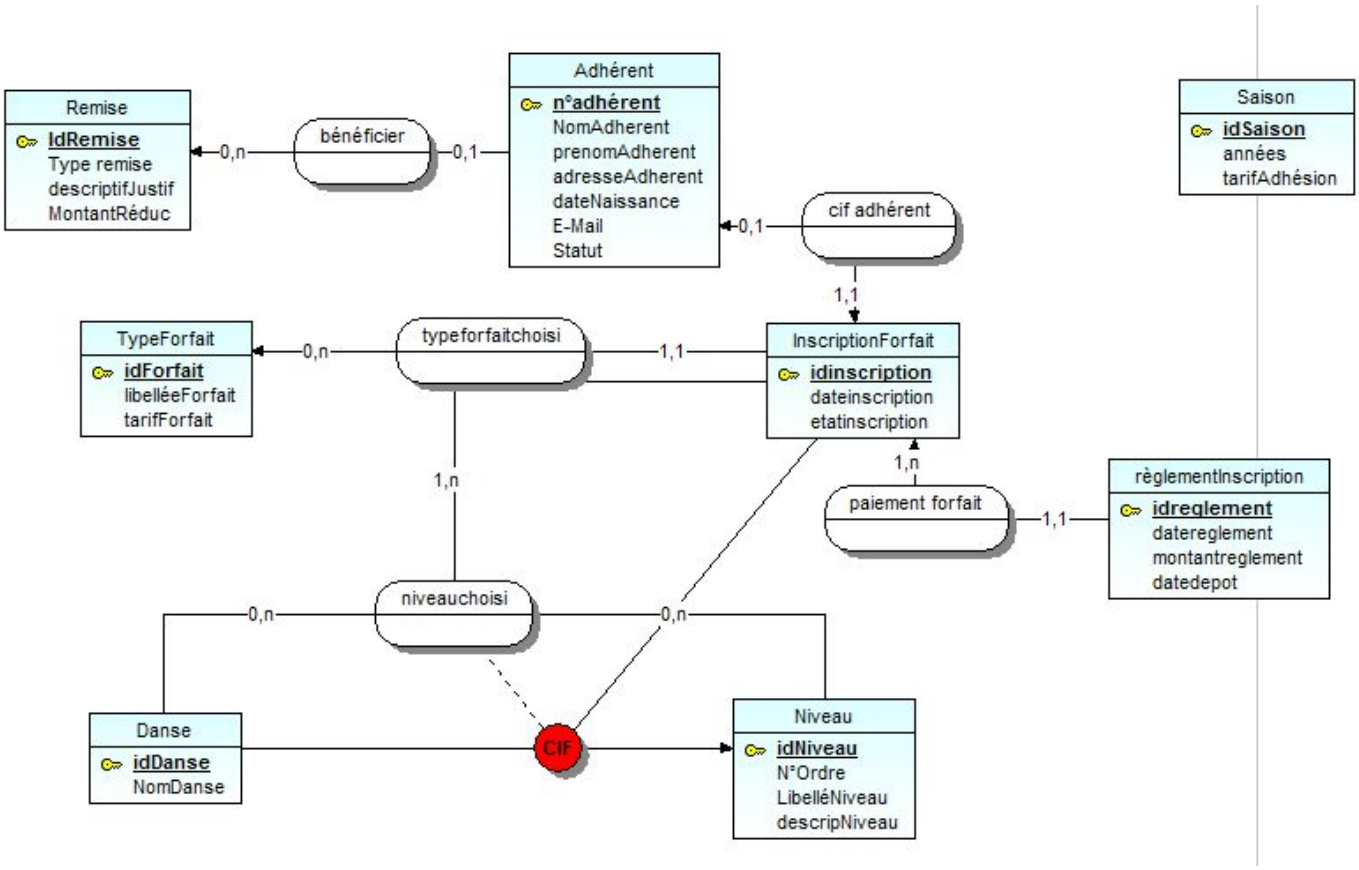
4.1 Modélisation conceptuelle des données

Nous avons du retravailler les différents MCD que nous avions proposé pour la première partie du projet, ils ont été modifiés à plusieurs reprises, mais nous n'arrivions pas à un résultat satisfaisant. Nous avons alors pris le parti de nous appuyer sur les schémas projetés en cours, en guise de correction.

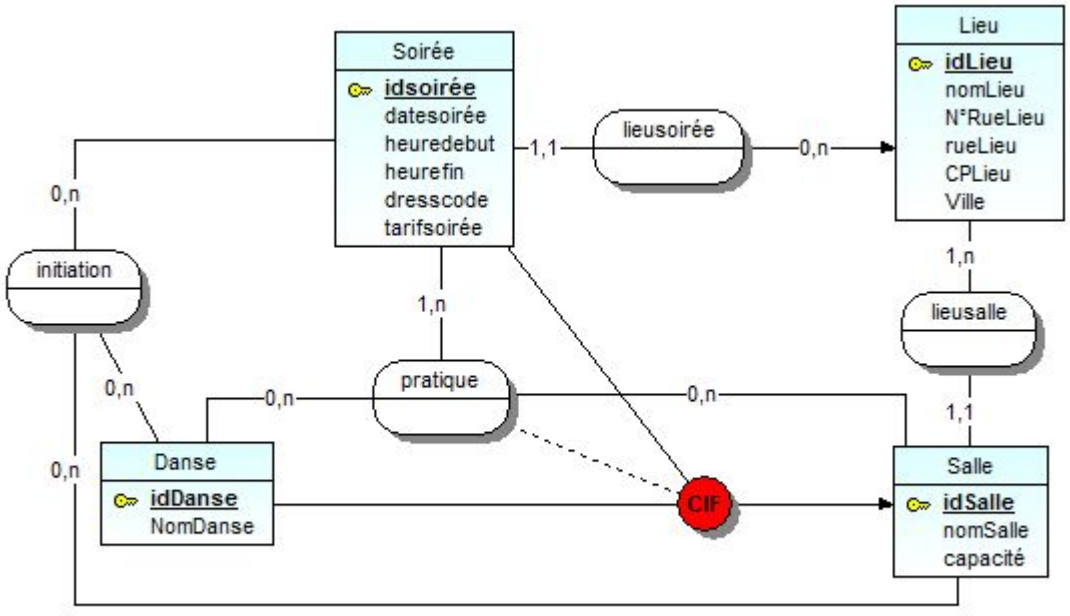
MCD Pré-saison



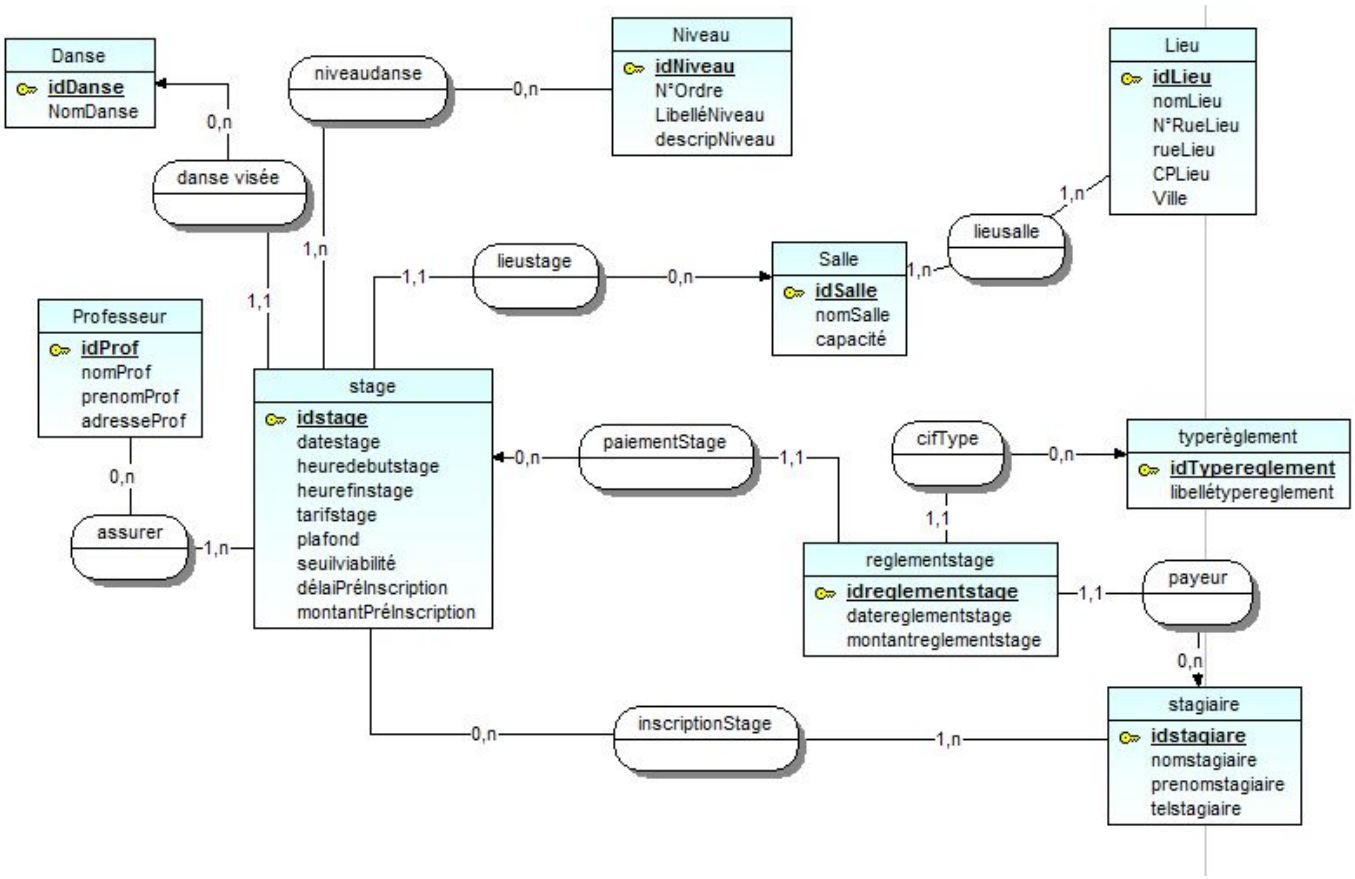
MCD Inscription



MCD Soirée



MCD Stage

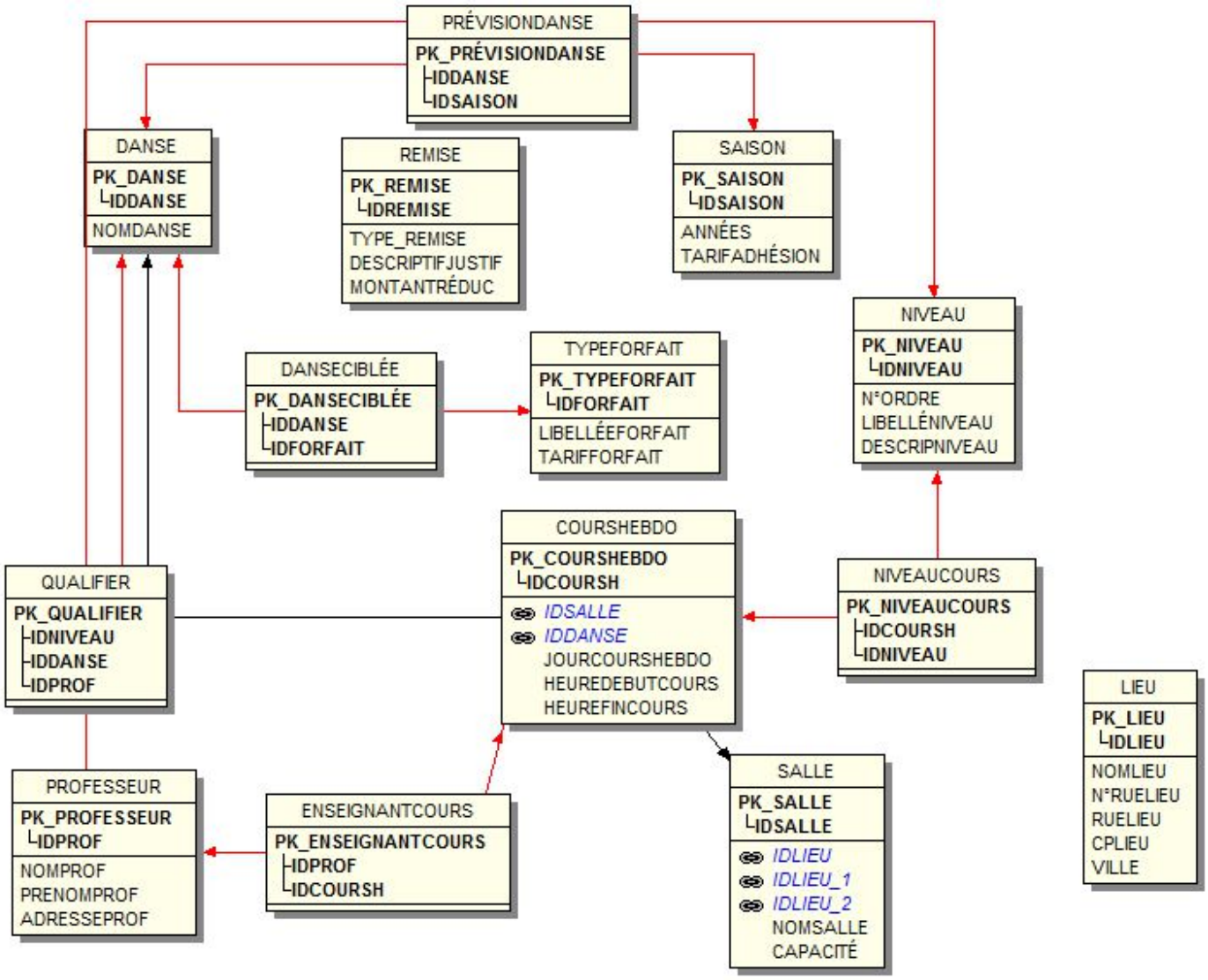

Danse
idDanse
NomDanse
niveaudanse
Niveau
idNiveau
N°Ordre
LibelléNiveau
descripNiveau
Lieu
idLieu
nomLieu
N°RueLieu
rueLieu
CPLieu
Ville
danse visée
lieusalle
lieustage
Salle
idSalle
nomSalle
capacité
Professeur
idProf
nomProf
prenomProf
adresseProf
stage
idstage
datestage
heuredebutstage
heurefinstage
tarifstage
plafond
seuilviabilité
délaiPréInscription
montantPréInscription
paiementStage
cifType
typerèglement
idTypereglement
libelléTypereglement
assurer
reglementstage
idreglementstage
datereglementstage
montantreglementstage
payeur
stagiaire
idstagiaire
nomstagiaire
prenomstagiaire
telstagiaire
inscriptionStage
0,n
1,n
1,1

5 Spécifications techniques et programmation de la solution retenue

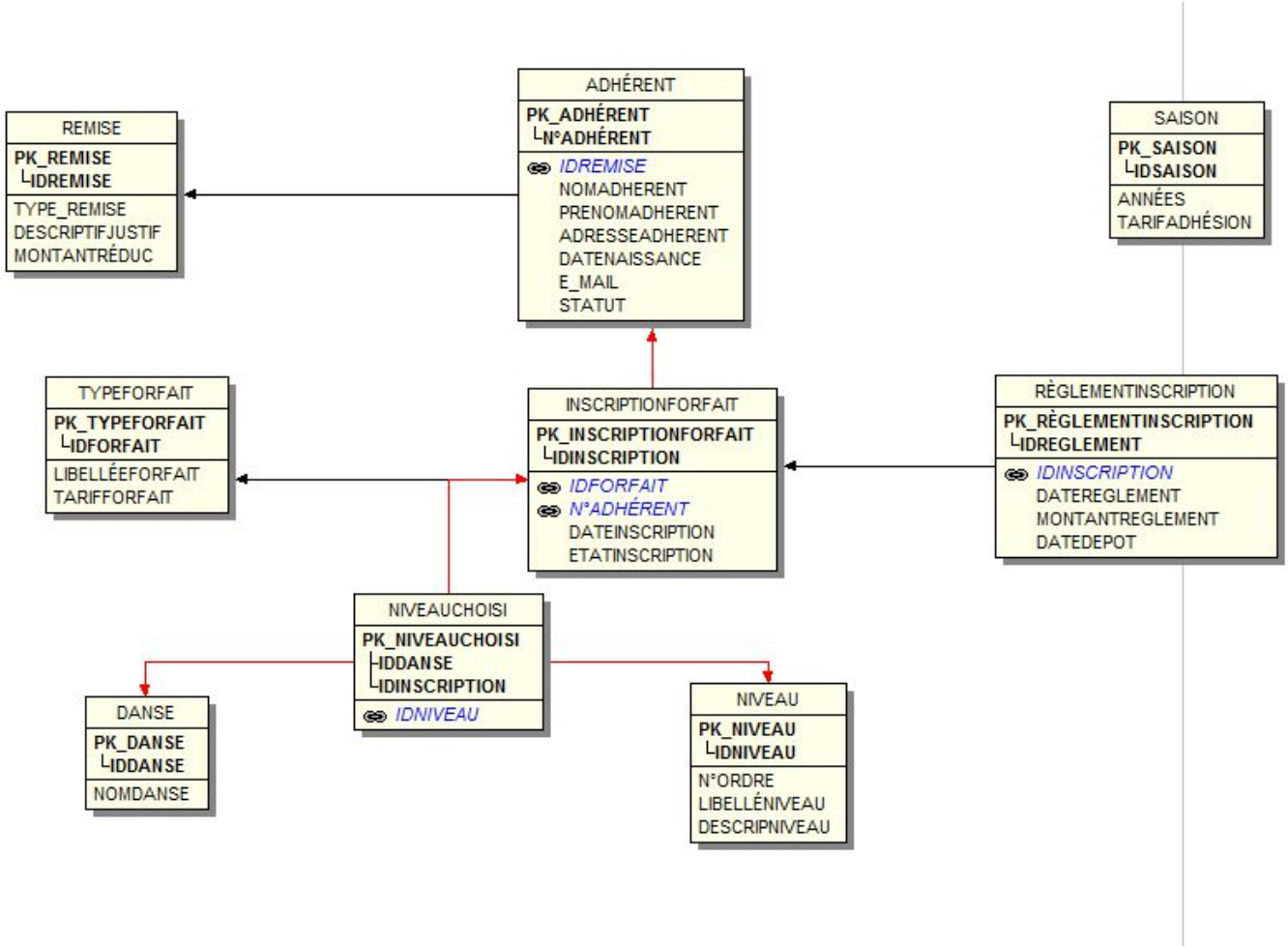
5.1 Modélisation logique de données

Après la réalisation et l'adoption des MCD ci-avant, nous en avons tiré les MLDR, étape indispensable pour générer, par la suite, le script pour créer les entités (via console), les clefs primaires et secondaires, les associations et par la suite compléter cette base de donnée.

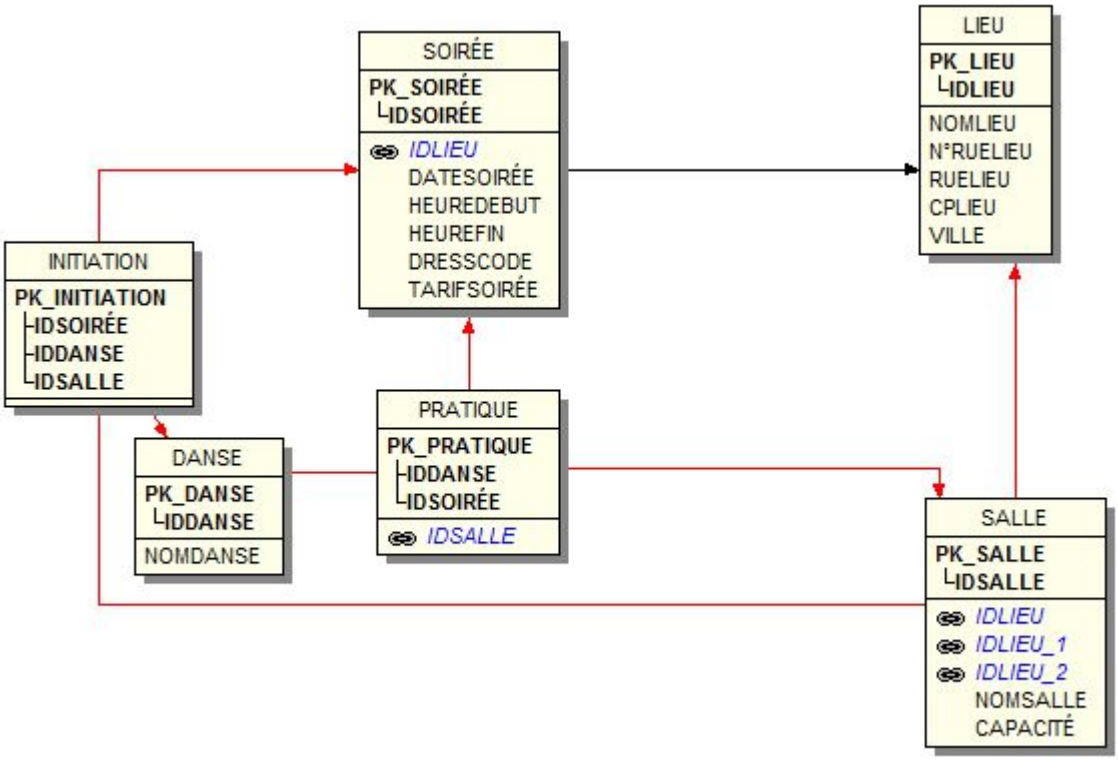
MLDR Pré-saison



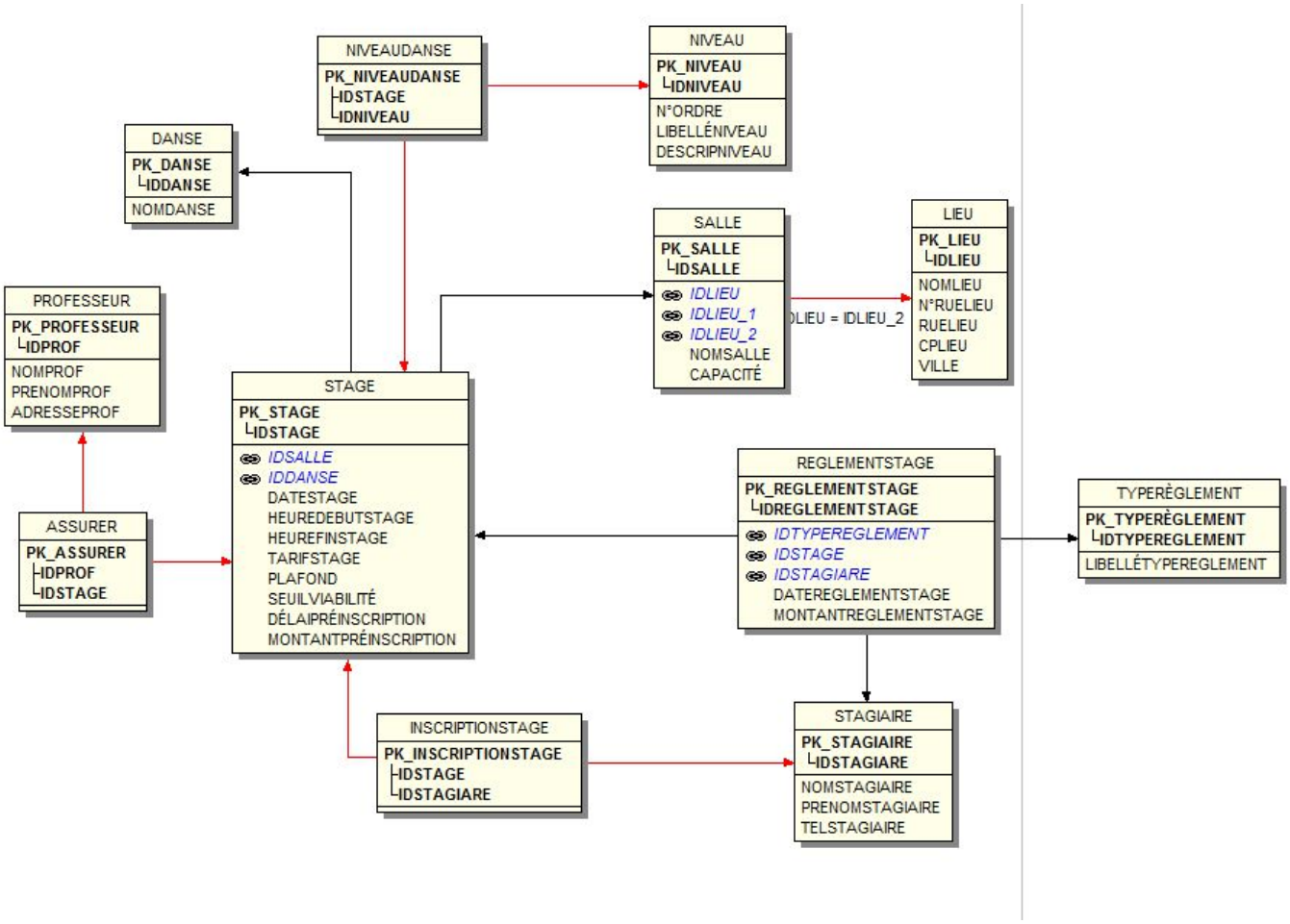
MLDR Inscription



MLDR Soirée

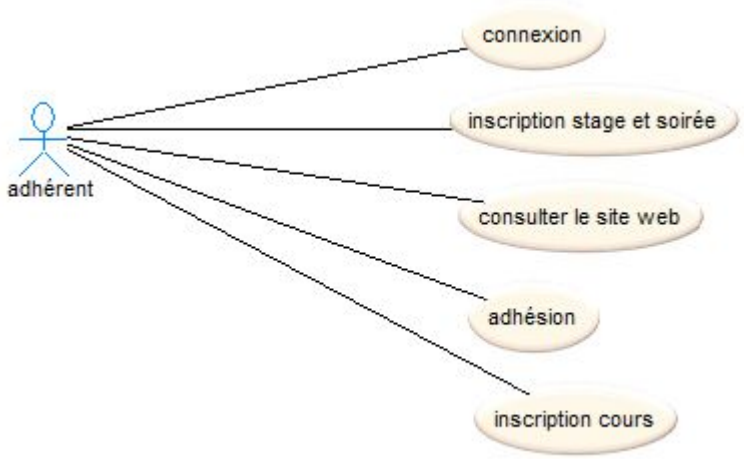

SOIRÉE
PK_SOIRÉE
IDSOIRÉE
IDLIEU
DATESOIRÉE
HEUREDEBUT
HEUREFIN
DRESSCODE
TARIFSOIRÉE
LIEU
PK_LIEU
IDLIEU
NOMLIEU
N°RUELIEU
RUELIEU
CPLIEU
VILLE
INITIATION
PK_INITIATION
IDSOIRÉE
IDDANSE
IDSALLE
PRATIQUE
PK_PRATIQUE
IDDANSE
IDSOIRÉE
IDSALLE
DANSE
PK_DANSE
IDDANSE
NOMDANSE
SALLE
PK_SALLE
IDSALLE
IDLIEU
IDLIEU_1
IDLIEU_2
NOMSALLE
CAPACITÉ

MLDR stage



5.2 Diagramme de cas d'utilisation

Voici un exemple de diagramme de cas d'utilisation, il s'agit d'une version plus pousser de l'application qu'il aurait été possible de faire, nous avons pu proposé ce diagramme en entendant les nos collègues parler de leurs applications. Cependant notre site ne proposait que la consultation du site et l'adhésion, nous sommes alors restés fidèle à notre idée première.



5.3 Script de création de la base de donnée *(extrait)*

```
php app/console doctrine : generate : entity     -- création d'entité
namespace VIVADanse Bundle:Image                 -- nommer
configuration : annotation                       -- annotation
new field name : alt                             -- attribut de l'entité
field type : string                              -- le type (chaîne de caractère)
field length : 255                               -- la taille (par défaut 255)
nullable : false                                 -- peut il être nul ?
unique : false                                   -- peut il être unique ?
```

5.4 Architecture de l'application

Vues

On a le layout générale dans SymfonyProject/app/Ressources/vues, le layout du Bundle VIVADance dans SymfonyPoject/VIVA/DanceBundle/Ressources/vues qui étend le layout de base et puis chaque fichier html.twig dans SymfonyPoject/VIVA/DanceBundle/Ressources/vues/Dance qui sont propres à chaque page, on a aussi un fichier formulaire.html.twig qui construit un formulaire qui est lui-même inclus dans inscription.

Contrôleur

DanceController qui se trouve dans src/VIVA/DanceBundle/Controller/Dance qui gère toutes les actions du site.

Routage

Routage par défaut définit au niveau de app/config/route.yml avec dance

Routage des différentes parties du sites est définit au niveau src/VIVA/DanceBundle/Ressources/config/route.yml.

Formulaires

Tous les objets formulaires sont définis dans src/VIVA/DanceBundle/Form

Entity

Les entités qui permettent de définir notre base de données avec Doctrine/ ORM sont définis dans src/VIVA/DanceBundle/Entity.

Repository

Permet la récupération des données à partir de la base de données et là où peuvent être écrites nos requêtes sont définis dans src/VIVA/DanceBundle/Repository.

Fixtures

Permettent de faire un jeu de données, ses jeux de données sont dans src/VIVA/DanceBundle/DataFixtures

Creation des Entités

On génère la base de données [vivadance] en lancant la commande [php app/console dotrine :database :create].

Toutes les entités à la suite sont généré par la commande [php app/console doctrine :generate :entity] et ont in id par défaut.

Adhérent : est créé avec les attributs de base nom, prénom, émail, codePostal, adresse et ville et statut, par la suite on a une relation **ManyToOne** avec remise caractérisé par l'attribut remise car on peut avoir qu'une seule remise par adhérent mais la même remise peut être accordée à plusieurs adhérent mais on peut avoir des adhérents sans remise aussi.

Cours : est créé avec les attributs de base jour, heure de début, de fin, par la suite on a suite relation **ManyToMany** avec l'entité professeur caractérisé par l'attribut professeur car on peut avoir plusieurs professeurs qui peuvent enseigner le même cours et le même professeur qui enseigne plusieurs cours, et c'est la même relation avec l'entité niveau caractérisé par l'attribut niveau.

On a une relation **ManyToOne** avec l'entité salle et dance car un cours ne se déroule que dans une salle et ne concerne qu'une danse mais une salle peut accueillir plusieurs cours et plusieurs cours peuvent avoir un même niveau et ses relations sont caractérisés par l'attribut salle et dance.

Dance : n'a qu'un attribut de base qui est son nom.

Forfait : a pour attributs de base son libelle et son tarif forfaitaires et par la suite établit une relation ManyToMany avec l'entité danses car plusieurs danses peuvent être dans un forfait comme une même dance peut être dans plusieurs forfaits.

Inscription : a pour attributs de bases date qui a valeur par défaut le temps actuelle, état, on a une relation **ManyToOne** avec forfait car un adhérent ne peut s'inscrire qu'à un seul forfait mais le forfait peut être souscris par plusieurs adhérents.On a une relation **OneToOne** avec adhérent car une inscription ne concerne qu'un seul et unique adhérent et qu'un adhérent ne peut s'inscrire qu'une seule et unique fois dans l'année.

Lieu : a pour attribut nom, numéro de rue, rueLieu, cpLieu comme code postale de lieu le nom de la ville.

Niveau : a pour attribut de base ordre, libelle et descriptif.

Niveau Choisie : n'a aucun attribut de base à part so, $id, mais a deux relations ManyToOne avec l'entité inscription et danse car à l'inscription on peut choisir plusieurs danses et une même danse peut être choisie dans plusieurs inscription différentes mais ce choix inclus le niveau de la dance et c'est pourquoi on a une relation ManyToOne avec l'entité niveau.

Pratique : ressemble à l'entité niveau choisie avec deux relations ManyToOne avec l'entité soiree et dance pour montrer qu'on peut avoir plusieurs dance sans une soirée et de même une danse dans plusieurs soirée mais cela est relatif à la salle dans laquelle se passe la soirée.

Professeur : a pour attribut de bases nom, prénom et adresse.

Règlement Stage : a pour attribut de base la date et le montant a une relation **ManyToOne** avec les entités stagiaires, stage et type de règlement.

Règlement (inscription) : a pour attribut de base date de dépôt, le montant et la date a une relation ManyToOne avec inscription car on a règlement spécifique à une inscription de forfait mais le règlement peut être payé en plusieurs fois.

Remise : a pour attributs de base type descriptif et montant.

Saison : a pour attribut de base année et tarif et a une relation ManyToMany avec car sur une année on peut avoir plusieurs dances et de même une danse sur plusieurs années.

Soiree : a pour attribut de base dateDebut et de Fin dresscode, tarif de la soirée et pour relation ManyToOne avec le lieu une soirée ne se déroule qu'a un seul lieu mais on peut avoir plusieurs soirées dans un lieu.

Stage : a une relation ManyToOne avec salle de même qu'avec dance on peut qu'avoir une seule danse dans un stage et de même se déroule dans une salle.

On a deux relations ManyToMany avec professeurs et stagiaires qui montre qu'on peut avoir plusieurs professeurs dans un stage et de même plusieurs stagiaires et vice versa.

Stagiaires : a pour attribut de base nom, prenom et telephone.

Type de Règlement (stage) : a pour attribut de base le libellé.

Salle : Comporte trois relations OneToOne avec l'entité Lieu qui veut dire chaque lieu a au maximum trois salles

5.5 Principe ergonomique

Une navigation efficace et ergonomique emmène le visiteur rapidement vers l'information recherchée, et toute information recherchée doit être accessible à partir de n'importe quelle page du site visité.

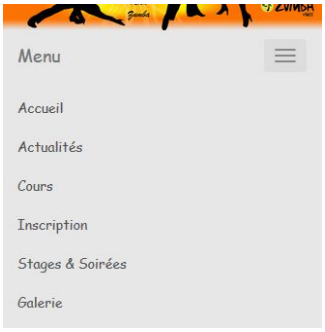
L'objectif premier de l'internaute, c'est de trouver l'information qu'il recherche ou effectuer les tâches qu'il désire faire. La navigation est uniquement le moyen d'y arriver. Pour cela, le système de navigation idéal doit être le plus simple et le plus efficace possible.

Nous avons choisi des couleurs vives et chaudes, pour rappeler le soleil, associé aux ombres chinoises. Le menu se situe en haut de la fenêtre, sous le visuel, une place pour laquelle les usagés sont habitués.

1280x600px

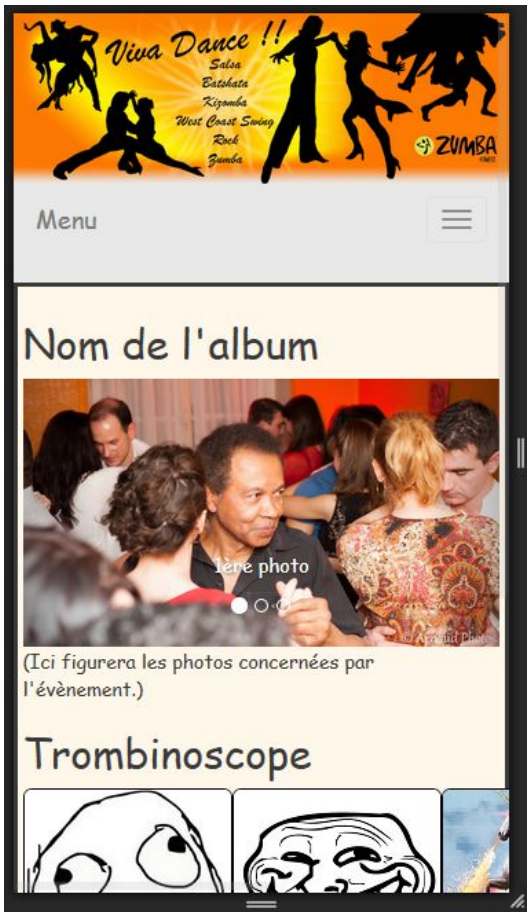
Bootsrtap :

Carrousel



Menu

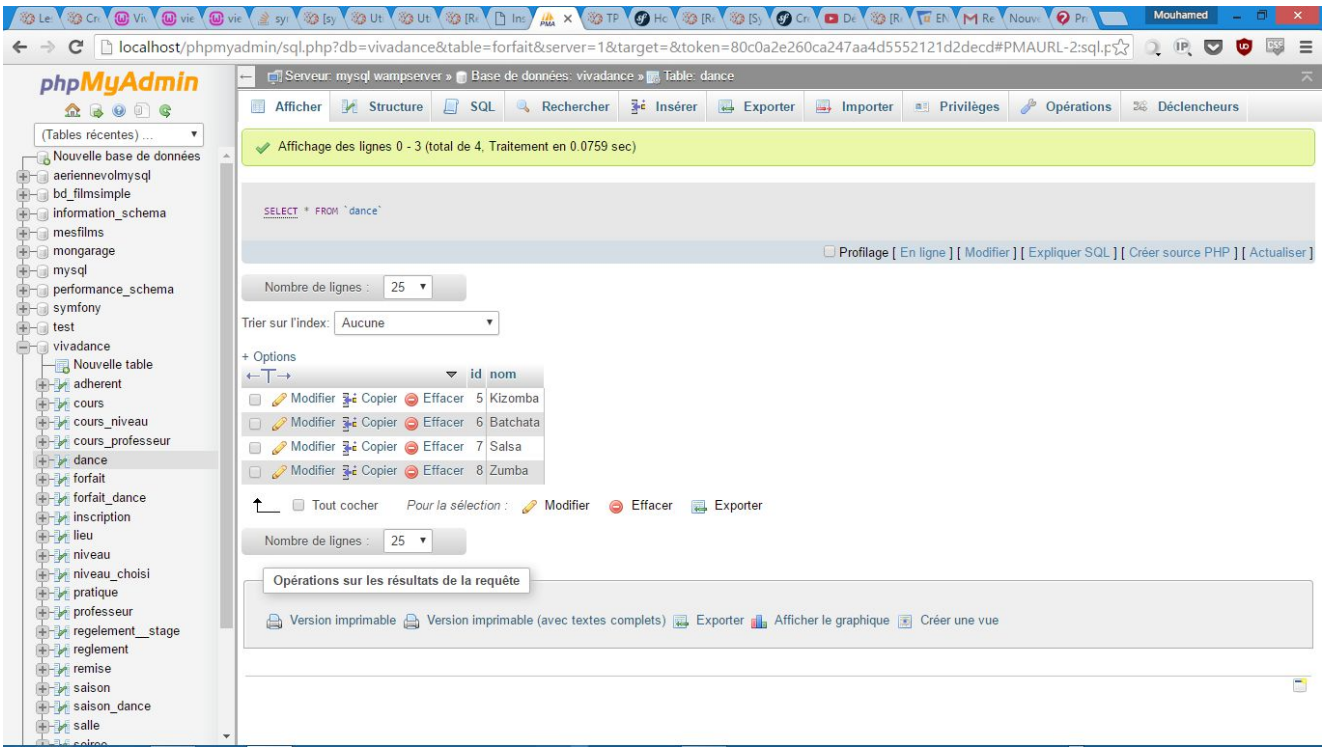
- les images se redimensionnent suivant la taille de la fenêtre
- les albums photos seront présentés sous forme de carrousel (il y aura plus que 3 images)
- le menu se transforme



360x640px

5.6 Spécification des jeux d'essai

un jeu de données danse créé avec des fixtures de doctrine qu'on peut lancer avec la commande : php /app /console doctrine : fixtures : load



6 Bilan et conclusion

6.1 Aurore DAUPHIN

Difficultés rencontrées

La plus grosse difficulté que j'ai rencontré sur ce projet est de me former sur Symfony, je n'ai pas réussi à comprendre comment cela fonctionne, le principe des bundles me parait très flou. J'ai alors, en accord avec mon binôme, convenu de lui confier la partie codage et Symfony, et je m'occupais, alors, du reste du projet, la responsivité du site, les différents schémas et diagrammes et le dossier.

Bilan humain

En ce qui concerne le bilan humain, nous avons choisi de garder le même groupe que précédemment, malheureusement un des membres a décidé de stopper la formation et à alors quitté le groupe, nous aurions eu la possibilité d'intégrer un nouveau membre, cependant nous avons préféré rester à deux et conserver l'intégrité de notre idée initiale. Contrairement au semestre précédent nous avons beaucoup travaillé séparément, en gardant toute fois des instants de concertations.

Bilan Technique

D'un point de vu technique, je n'arrive pas à trouver une satisfaction convenable, n'ayant pas réussi à assimiler le framwork Symfony, je prends cela comme un échec, toute fois, ne pouvant pas tout confier à mon binôme j'ai pu me former à Boostrap, outils que je trouve très intéressant, qui mérite un approfondissement, il permet de rendre un site web plus attractif en se servant de « morceau » de code déjà établi et enregistré, il suffi de « consulter le catalogue » et de choisir les fonctionnalités qui nous plaisent.

6.2 Mouhamed SEYDI

Segmentation en objets et extensibilité

Le framwork Symfony lui-même est très intéressant à apprendre, il permet de séparer tout le code en objets de telle manière que chaque objet n'a qu'une seule et unique fonction. De plus il est très extensible grâce à l'utilisabilité des bundles déjà créés comme les Fixtures, que j'ai eu à crée, mais aussi aux doctrines, extensions bundle, qui permettent de faire plein de choses comme crée des slug mais aussi de suivre des IP des visiteurs du site etc. où encore FOSUserBundle pour la gestion des utilisateurs du site dont je n'ai pas eu assez de temps pour apprendre son utilisation.

Doctrine

Ce qui a aussi été un gros point positif est la manière dont la base de données est gérée en doctrine, tout est tellement objet que même la base de données est gérée comme un objet elle a presque même plus à visiter phpMyAdmin. La mise en place des relations entre entités est aussi très de la manière traditionnelle, où grâces aux annotations on met en relation plusieurs avec des ManyToOne, OneToOne, ManyToOne ou même des combinaisons de ManyToOne pour pouvoir faire une association avec des attributs.
Pas seulement que doctrine de base est elle-même est puissante grâce plusieurs fonction de base déjà présentes pour insérer, ajouter chercher en fonction d'un identifiant ou selon un critère, elle extensible car on créer nous-même nos propres requêtes dans la repository de l'entité.

Gestion facilitée des formulaires

La génération des formulaires est même plus géniale grâce encore à des objets et manière très simple permet la communication avec la base de données sans avoir à écrire plusieurs lignes de codes car de base on construit les formulaires à partir des entités de doctrine donc plus besoin de plusieurs lignes de codes pour insérer supprimer ou même modifier
Ce qui est très intéressante aussi pour quelqu'un qui n'a jamais de MVC auparavant est la manière dont les routeurs et le contrôleur fonctionnent entre elle.