Projet de spécialité : Ski Club
Document d'implémentation

CLAUDOT Nathan - DOUMBIA Fadio - KITH Robert - BERRAZAGA Rania

11 juin 2015

Première partie
Choix techniques

Cette application web a été développée à l'aide du langage PHP et de MySQL, ce qui était imposé par le sujet. Le tuteur ayant fortement insisté sur la propreté et l'organisation du code, nous avons décidé d'utiliser le framework *Symfony2* qui permet d'une part de reproduire de façon très performante l'architecture Modèle Vue Contrôle et qui offre également une utilisation simplifiée de la base de données via l'outil *Doctrine*. *Symfony2* nous a vraiment été très utile notamment grâce à sa gestion des formulaires et au fait qu'il est possible de récupérer des parties de codes déjà implémentées (telles des formulaires d'inscription incluant la gestion de la sécurité). De plus, il est possible de créer des entités à partir de la console. Cela nous a permis de gagner un temps considérable. Aussi, *Symfony2* fournit un mode de "développement", facillitant le debuggage ce qui est très appreciable. En revanche, Doctrine n'est pas optimal : il est impossible de référencer un objet contenant plusieurs clés. Dans certaines tables, aucun référencement n'a pu être fait ce qui nous empêche d'effectuer, par exemple, la commande "on delete cascade". Tout doit donc être "fait à la main" (lors de modification, de suppression). *Doctrine* gère également mal l'objet DateTime qui ne peut être une clé.
Afin de comprendre au mieux le framework *Symfony2*, nous nous sommes basés tout au long du projet sur la documentation officielle de *Symfony2* `http://symfony.com/fr/doc/2.7/book/index.html` ainsi que le tutoriel de openclassroom `http://openclassrooms.com/courses/developpez-votre-site-web-avec-le-framework-symfony2` .

Deuxième partie
Base de données

Le schéma relationnel de notre base de données est le suivant :
Vous trouvez en annexe le diagramme UML.

- Table stockant les utilisateurs :
  User(e-mail, nom, prenom, telephone, adresse, commune, password, salt, type, isActive, isPrimaire, e-mailprimaire)
  type appartient à Client, Admin
  isPrimaire est un booléen (1 si le compte est primaire 0 sinon)
  isActive booléen
  e-mailprimaire référence User, peut être nul

- Table stockant les enfants des utilisateurs :
  Enfant(e-mail, nomEnfant, prenomEnfant, niveauSki, dateNaissance)
  niveau Ski peut être nul, référence niveauSki
  e-mail référence User
- Table stockant les niveaux de ski :
  niveauSki(niveau)
  niveau apppartient à Pas d'étoile, Flocon, 1ère étoile, 2ème étoile,3ème étoile, Etoile d'or, Etoile de bronze
- Table stockant les lieux des différentes sorties :
  Lieu(nomLieu)
- Table stockant les saisons :
  Saison(annee)
- Table stockant les licences :
  Licence(typeLicence, prixLicence)
- Table stockant les licences qu'utilise un enfant pour une saison donnée :
  LicenceEnfant(e-mail, nomEnfant, prenomEnfant, TypeLicence, annee)
  e-mail, nomEnfant, prenomEnfant référence Enfant
  TypeLicence référence Licence
  annee référence Saison
- Table stockant les adhérents au club et tout ce qui est en rapport avec leurs paiements :
  Adhesion(e-mail, date, modalite, remise, montantPaye, adhesionAnnuel)
  e-mail référence User
  annee référence Saison
  adhesionAnnuel boolean
- Table stockant les activités :
  Activite(id, nomActivite, description, prixActivite, TypeLicence, e-mail)
  e-mail référence User de type Admin
  TypeLicence référence Licence
  TypeLicence peut être nul
  description peut etre nul
- Table stockant les stages :
  Stage(id, debutStage, finStage, nomStage, description, prixStage, charges, nomLieu, email, annee)
  e-mail référence User de type Admin
  id référence Activite
  nomLieu référence Lieu
  annee référence Saison
- Table stockant les sorties :
  Sortie(id, dateSortie, annee, nomLieu, email)
  id référence Activite
  nomLieu référence Lieu
  annee référence Saison
  email référence User de type Admin
- Table permettant de savoir si une activité est programmée pour une saison donnée :
  Activites_Saisons(id, annee)
  année référence Saison
  id référence Activite
- Table permettant de connaitre les inscriptions des enfants aux activités pour une saison donnée :
  InscriptionActivite(email, nomEnfant, prenomEnfant, id, annee, groupe)

e-mail, nomEnfant, prenomEnfant référence Enfant
idActivité et date référence Activites_Saisons
groupe peut être nul (string)

- Table permettant de connaitre les inscriptions des enfants aux sorties pour une saison donnée :
  InscriptionSortie(email, nomEnfant, prenomEnfant, id, dateSortie, annee, participation)
  e-mail, nomEnfant, prenomEnfant référence Enfant
  id, dateSortie référence Sortie
  annee référence Saison
  participation booléen : indique si l'utilisateur a confirmé ou non la présence de l'enfant
- Table permettant de connaitre les inscriptions des enfants aux stages pour une saison donnée :
  InscriptionStage(e-mail, nomEnfant, prenomEnfant, id, debutStage, finStage, prixPayeStage, annee)
  e-mail, nomEnfant, prenomEnfant référence Enfant
  idActivité,debutStage,finStage référence Stage
  année référence Saison
  prixPayeStage est le montant payé du stage : sert à vérifier si le stage a été payé.

Troisième partie
Architecture

1 Architecture générale du site

Notre application web contient les répertoires ci-dessous :

- *app* : contient le cache, les logs et les templates dédiés à l'application entière ainsi que la classe *AppKernel*, qui est l'objet de base de l'application.
- *src* : contient les bundles, là où est écrit le code source.
- *vendor* : contient les bibliothèques utilisées telles que *Doctrine*, *Twig* ...
- *web* : contient les fichiers CSS, JavaScript et les images. Il contient également deux contrôleurs frontaux *app.php* utilisé en mode production et *app _dev.php* utilisé en mode développement. Le contrôleur frontal en *Symfony2* est l'équivalent de *index.php*. Ce dernier appelle le *Kernel* (le noyau de *Symfony2*) pour traiter la requête.

Les Bundles

Un Bundle est une brique de l'application qui porte sur une fonctionnalité déterminée. Grâce aux Bundles, on a pu découper notre projet selon les différentes fonctionnalités.
Ci-dessous sa structure qui illustre bien l'architecture Modèle Vue Contrôleur :

/ *Controller* : Contient les contrôleurs
/*DependencyInjection* : Contient des informations sur le Bundle (chargement automatique de la configuration par exemple)
/ *Entity* : Contient les modèles
/ *Form* : Contient les éventuels formulaires
/ *Resources*
- - / *config* : Contient les fichiers de configuration du Bundle (comme les routes dans *routing.yml*)
- - / *public* : Contient les fichiers publics du bundle : fichiers CSS, JavaScript, images ...
- - / *views* : Contient les vues du Bundle, les templates Twig (fichiers twig.html)

Les routes

Le routage est un concept clé en *Symfony2*, il permet de créer un lien entre une URL et un contrôleur. Ci-dessous un exemple de route implémentée dans le Bundle ActiviteBundle :

sc _activite _view :
path : /activite/view/id
defaults : {*_controller* : *SCActiviteBundle* : *Activite* : *view*}
requirements :
id : \d +

Lorsqu'on appelle par exemple l'URL : /activite/view/5, le routeur cherche une correspondance entre un des paths existants (présent dans les fichiers *routing.yml*) et l'URL. Ici, le chemin est /activite/view/id. Ensuite, il détermine les paramètres : le contrôleur : Activite, la méthode à exécuter : viewAction et le paramètre id = 5 et les envoie au Kernel qui exécute le contrôleur approprié avec les bons paramètres.

Doctrine

L'ORM (Object Relation Mapper) a pour objectif d'enregistrer une entité dans la base de données. L'ORM utilisé par *Symfony2* est *Doctrine2*. Voir la Documentation de Doctrine `http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/index.html` .
Du point de vue de *Doctrine*, une entité est un objet complété avec des informations de mapping (des métadata qui peuvent être en annotation, en XML, YAML ou en PHP voir `http://symfony.com/fr/doc/current/book/doctrine.html` pour plus de détails. Ce sont ces données qui lui permettent d'enregistrer l'objet en base de données. Dans notre projet, on a utilisé les annotations.

2 Les Bundles implémentés

ActiviteBundle

Il gère les activités, les sorties et les stages du SkiClub. Les contrôleurs permettent de les afficher, de faire des ajouts, des modifications et des suppressions. C'est également eux qui s'occupent d'inscrire les enfants, de leur attribuer des licences si nécessaire et d'afficher les différentes inscriptions aux activités, stages et sorties. Ils gèrent également la confirmation des paiements des stages, des activités et la confirmation où l'annulation de la participation à une sortie.

LicenceBundle

Ce Bundle permet la gestion des licences des activités. Le contrôleur permet d'afficher, d'ajouter et de modifier des licences.

UserBundle

Il s'occupe de tout ce qui est en relation avec l'utilisateur. Il fournit une interface permettant de s'inscrire, se connecter, d'ajouter des enfants et des utilisateurs secondaires. Il gère également l'affichage de la liste des inscriptions aux activités des enfants d'un utilisateur donné ainsi que le montant total à payer. Enfin, il gère les actions de l'administrateur du site qui sont : l'activation et la suppression des comptes, la gestion des comptes des utilisateurs (adhésion, remise, modalités, confirmation de paiement, montant restant à payer) et de leurs enfants (affectation de groupes aux enfants inscrits aux activités). Par ailleurs, ce Bundle gère la trésorerie et permet d'afficher le bilan de la saison en cours.

3 Interface

La figure ci-dessous illustre l'interface de notre site web :

FIGURE 1 – Interface du site web

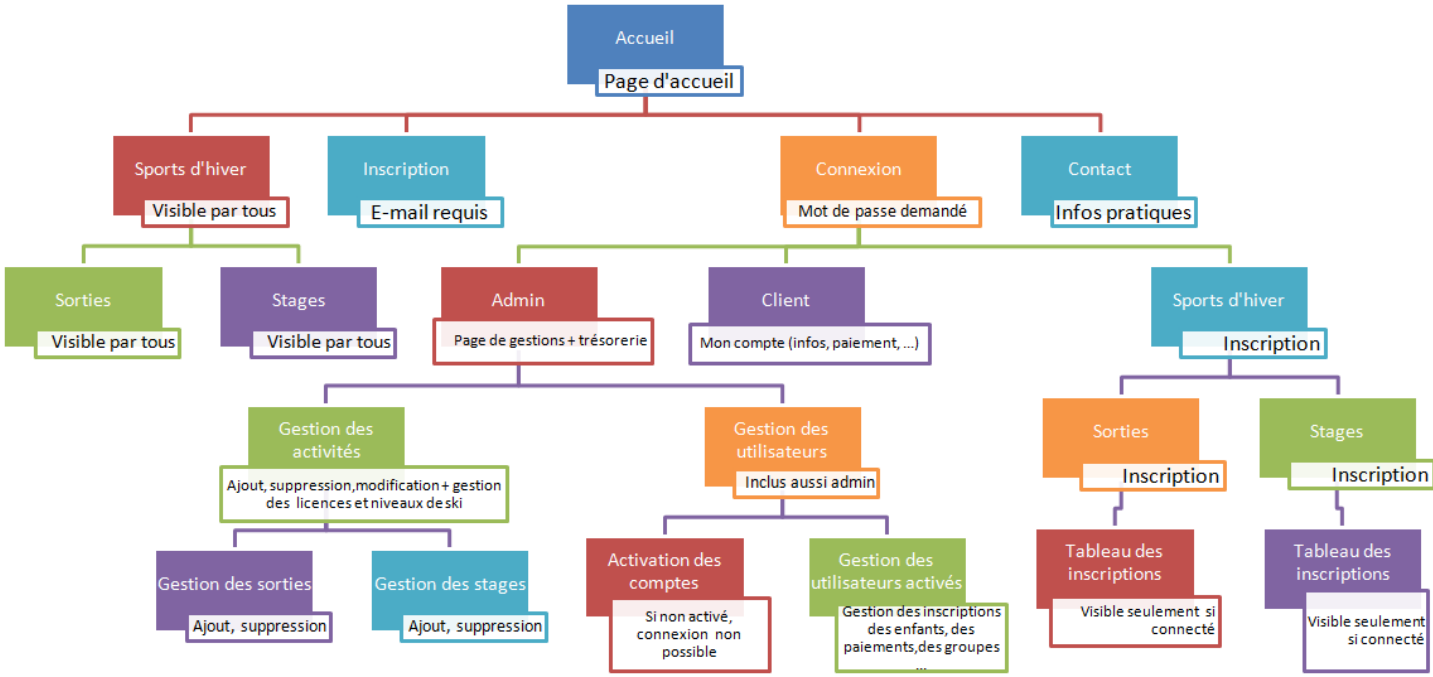
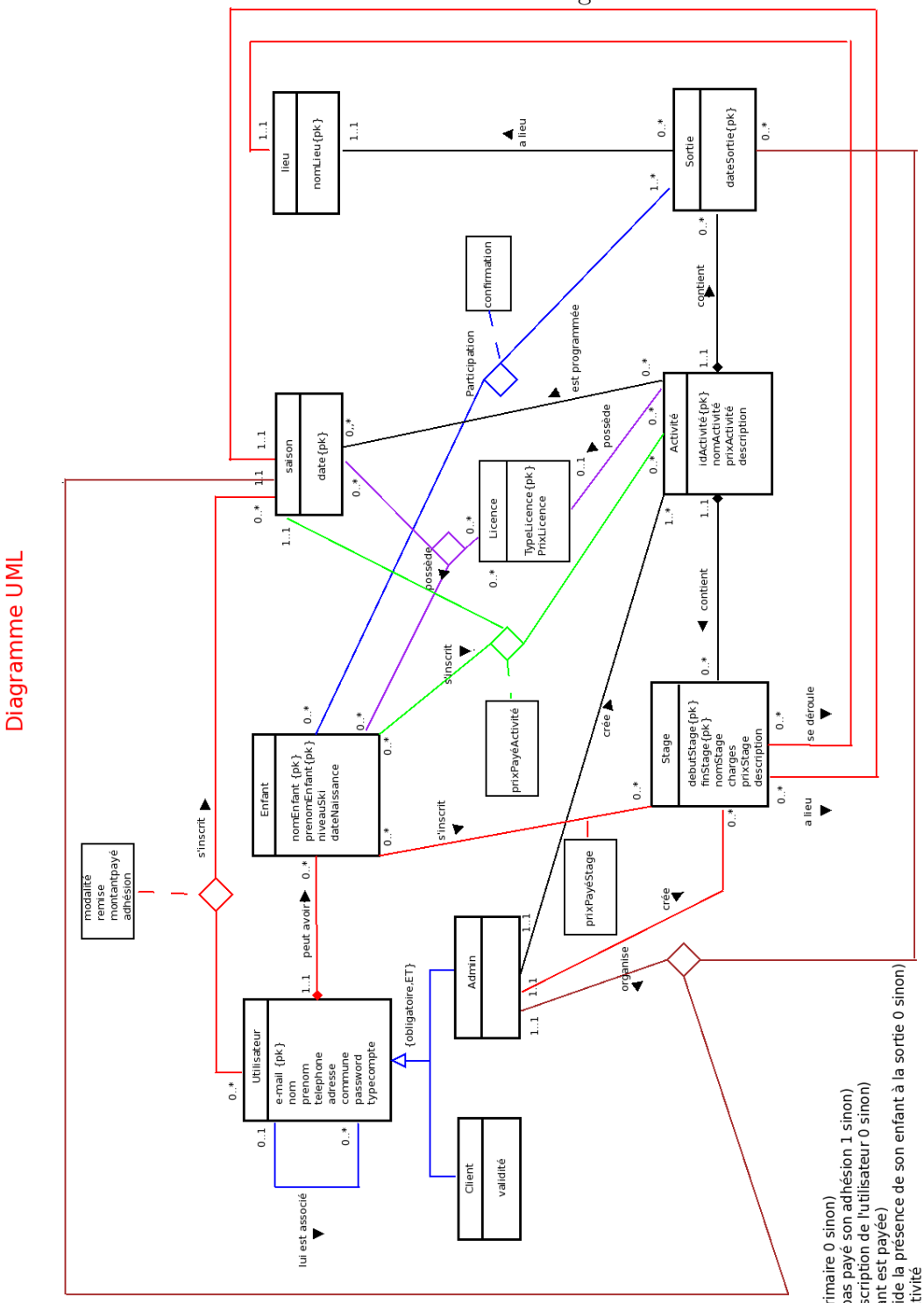


FIGURE 2 – Diagramme UML



Contraintes:
typecompte est un booléen (1 si le e-mail est primaire 0 sinon)
adhésion peut être 0 ou 1 (0 si l'utilisateur n'a pas payé son adhésion 1 sinon)
validité peut être 0 ou 1 (1 si l'admin valide l'inscription de l'utilisateur 0 sinon)
licence peut être 0 ou 1 (1 si la licence de l'enfant est payée)
confirmation peut être 0 ou 1 (1 si le parent valide la présence de son enfant à la sortie 0 sinon)
prixPayéActivité est le montant payé <= prixActivité
prixPayéStage est le montant payé du stage (si l'utilisateur n'a pas payé ça vaut 0)