

RAPPORT DE STAGE

Benoît CATILLON

Concepteur Développeur Informatique

Stage effectué dans l'entreprise SpotMyDive

Du 6 Mars au 5 Mai 2017



Projet basé sur



« Il y existe deux manières de concevoir un logiciel.

La première, c'est de le faire si simple qu'il est évident qu'il ne présente aucun problème.

La seconde, c'est de le faire si compliqué qu'il ne présente aucun problème évident.

La première méthode est de loin la plus complexe »

Charles Antony Richard HOARE

Table des Matières

CHAPITRE 1 - PRESENTATION	- 5 -
1.1 – AVANT-PROPOS	- 6 -
1.2 – ABSTRACT	- 7 -
1.3 – REMERCIEMENTS	- 8 -
1.4 – LA DEMARCHE GENERALE	- 9 -
CHAPITRE 2 – CAHIER DES CHARGES	- 10 -
2.1 – PRESENTATION DE L'ENTREPRISE	- 11 -
2.2 – TECHNOLOGIES IMPOSEES	- 13 -
2.3 – CONTEXTE DE DEVELOPPEMENT	- 14 -
CHAPITRE 3 - ANALYSE	- 15 -
3.1 – DIAGRAMMES DE CAS D'UTILISATION	- 16 -
3.1.1 – REPRESENTATION GRAPHIQUE	- 17 -
3.1.2 – DCU FRONT-END	- 18 -
3.1.3 – DCU BACK-END	- 19 -
3.1.4 – DCU ADMIN	- 20 -
3.1.5 – FICHE DE DESCRIPTION TEXTUELLE D'UN CAS D'UTILISATION	- 21 -
3.1.6 – DIAGRAMME D'ACTIVITES	- 22 -
3.2 – LES MAQUETTES	- 24 -
3.3 – DIAGRAMME DE NAVIGATION	- 28 -
CHAPITRE 4 – CONCEPTION DE LA BASE DE DONNEES	- 29 -
4.1 – INTRODUCTION	- 30 -
4.2 – LA DEMARCHE UTILISEE	- 30 -
4.3 – DIAGRAMME DE CLASSES	- 31 -
4.4 – MODELISATION PHYSIQUE DES DONNEES	- 32 -
4.5 – DOCTRINE2	- 34 -
4.6 – SQL : LDD	- 35 -
4.6.1 – CREATION DE LA BASE DE DONNEES	- 36 -
4.6.2 – STRUCTURE ET CREATION DE LA TABLE « AFFILIATION »	- 36 -

CHAPITRE 5 – CONCEPTION DE L'APPLICATION	- 37 -
5.1 – DIAGRAMME DE SEQUENCE SYSTEME	- 38 -
5.2 – DIAGRAMME DE SEQUENCE DETAILLE	- 40 -
CHAPITRE 6 – DEVELOPPEMENT	- 41 -
6.1 – TECHNOLOGIES UTILISEES	- 42 -
6.1.1 – SYMFONY	- 42 -
6.1.2 – JQUERY	- 42 -
6.2 – ARCHITECTURE MVC	- 43 -
6.3 – LES BUNDLES SYMFONY	- 44 -
6.3.1 – CONTROLLER	- 45 -
6.3.2 – ENTITY	- 45 -
6.3.3 – FORM	- 46 -
6.3.4 – RESOURCES	- 46 -
6.4 – STRUCTURE D'UN CONTROLLER SYMFONY	- 47 -
6.5 – STRUCTURE D'UNE ENTITE SYMFONY	- 49 -
6.5.1 – LES PROPRIETES	- 49 -
6.5.2 – LES ACCESSEURS ET LES MUTATEURS	- 50 -
6.6 – STRUCTURE D'UN FORMULAIRE « FORM »	- 51 -
6.7 – STRUCTURE D'UNE PAGE TWIG	- 52 -
6.8 – JQUERY	- 53 -
6.9 – AUTRES LANGAGES	- 55 -
6.9.1 – BOOTSTRAP	- 55 -
6.9.2 – HTML / CSS	- 55 -
CHAPITRE 7 – DEPLOIEMENT DE L'APPLICATION	- 56 -
7.1 – DEPLOIEMENT	- 57 -
7.2 – DIAGRAMME DE DEPLOIEMENT	- 61 -
CHAPITRE 8 – GESTION DE PROJET	- 62 -
8.1 – GIT	- 63 -
8.2 – GESTION DE GIT AVEC PHPSTORM	- 64 -
8.3 – GESTION DE GIT AVEC GITG	- 65 -

CHAPITRE 9 – SECURITE	- 66 -
9.1 – LA SECURITE AVEC SYMFONY	- 67 -
9.2 – FAILLES XSS, CSRF, ET INJECTIONS SQL	- 67 -
CHAPITRE 10 – CONCLUSION	- 68 -
10.1 – LA FORMATION	- 69 -
10.2 – LE STAGE	- 69 -
10.3 – PROJET FUTUR	- 69 -
CHAPITRE 11 – ANNEXES	- 70 -
11.1 – CORRESPONDANCE PROJET / REAC	- 71 -
11.2 – LES OUTILS UTILISES	- 72 -
11.3 – BIBLIOGRAPHIE ET WEBOGRAPHIE	- 73 -
11.3.1 – BIBLIOGRAPHIE	- 73 -
11.3.2 – WEBOGRAPHIE	- 73 -
11.4 – CAHIER DES CHARGES	- 74 -
11.5 – CODE COMPLET DE CREATION DE LA BD	- 82 -
11.6 – GLOSSAIRE / LEXIQUE	- 87 -

Chapitre 1 - Présentation

1.1 – Avant-Propos

Mon parcours professionnel a commencé par un stage d'un an dans un magasin de réparation de matériels informatiques. J'ai pu y apprendre à monter un ordinateur pièce par pièce, déceler les anomalies courantes pour du dépannage à domicile ou en magasin, et optimiser les performances d'un ordinateur.

J'ai alors émis le souhait de poursuivre mon parcours scolaire dans ce domaine. Mais j'ai dû faire face à la réalité. Peu d'établissements en France à l'époque proposaient des formations dans le domaine informatique. Originaire de Picardie, l'accès m'y était impossible. J'ai donc été réorienté à l'âge de 15 ans vers un CAP Installation Sanitaire. Diplôme que j'ai obtenu après un cursus de 2 ans en apprentissage. Aucunement passionné par ce métier, j'ai alors effectué diverses missions d'intérim ainsi que des contrats à durée déterminée.

Mais désireux de retrouver un poste dans l'informatique, j'ai continué mes recherches. Et c'est avec le Pôle Emploi que j'ai pu intégrer, en 2010, la FNAC, via son programme d'initiation aux métiers de l'entreprise, organisé par le Groupe PPR, avant d'être confirmé à mon poste de « vendeur conseiller en produits techniques » en CDI dans le magasin de La Défense.

Après presque 7 ans passés à mon poste, en constante évolution, allant du conseil à la vente d'ordinateurs, de périphériques, en passant par les GPS et les objets connectés, pour finir responsable de l'univers du smartphone, j'ai aujourd'hui l'envie de continuer ma carrière professionnelle dans la conception et le développement web.



1.2 – Abstract

In this report, I am going to display my professional experience in SpotMyDive, a community platform of scuba diving. Their business model is based on reservation of diving courses all around the world.

During this internship, I was able to apply my knowledge learned at M2i Formation, to strengthen it and learn a lot of new things.

This report displays every step I made to accomplish various tasks in the company that welcomed me during two months as well as my strategic reflexion and the tools I used to meet the expectations.

Dans ce rapport, je vais vous présenter mon expérience professionnelle au sein de l'entreprise SpotMyDive, plateforme communautaire dédiée à la plongée sous-marine.

Le modèle économique est basé sur la réservation de cours de plongée à travers le monde.

Lors de cette expérience, j'ai pu mettre en pratique les connaissances acquises chez M2i Formation, tout en les consolidant, avec une bonne dose de nouveautés au passage.

Ce rapport regroupe toutes les étapes par lesquelles je suis passé pour accomplir les différentes missions données par l'entreprise qui m'a accueilli pendant ces deux mois. Mais également la réflexion ainsi que les outils utilisés pour répondre à ces demandes.



1.3 – Remerciements

Je tiens à remercier **Pascal BUGUET** ainsi que **Sébastien MALORON**, qui ont tenu un enseignement exemplaire, et sans qui cette aventure aurait été des plus sinieuse.

Je remercie également **Lou BERNARDINI**, **Lukas JOSEPH** et **Damien PARA**, qui m'ont fait confiance au point de me laisser tenir une des rennes de leur bébé.

Ainsi que **Mahery RALAINDIMBY**, mon binôme en cours comme en stage, avec qui j'ai été très complémentaire.

Et par-dessus tout, **ma femme**, pour sa patience et son soutien, qui m'a supporté bien avant l'entrée en formation, et qui devra continuer encore quelques temps après...

Pour finir, je veux remercier **tous mes camarades de formation**, avec qui j'ai passé d'excellents moments, et qui m'ont beaucoup apporté.



1.4 – La démarche générale

SpotMyDive est un site développé sous le framework Symfony 2. J'ai donc passé la première semaine du stage à découvrir le projet.

Le développeur en charge du projet est basé à Biarritz. Nous avons travaillé ensemble dès la première semaine de stage. Il m'a accompagné dans la découverte et la compréhension du travail qu'il avait accompli jusqu'à présent sur ce projet. Nous avons donc passé en revue les besoins de SpotMyDive, puis nous avons établi le cahier des charges, ce que j'allais devoir mettre en œuvre.

La distance ainsi que la multiplicité des projets gérés par ce développeur étant des problématiques rapidement rencontrées, nous avons dû mettre en place une organisation rigoureuse et une bonne communication. Nous échangeons par mail et faisons des points d'étapes toutes les semaines. Il a toujours été d'une grande aide des lors que je rencontrais des difficultés.

Pour l'analyse ainsi que pour la conception, j'ai employé la démarche UML, découverte avec Monsieur Pascal BUGUET lors de ma formation chez M2i.

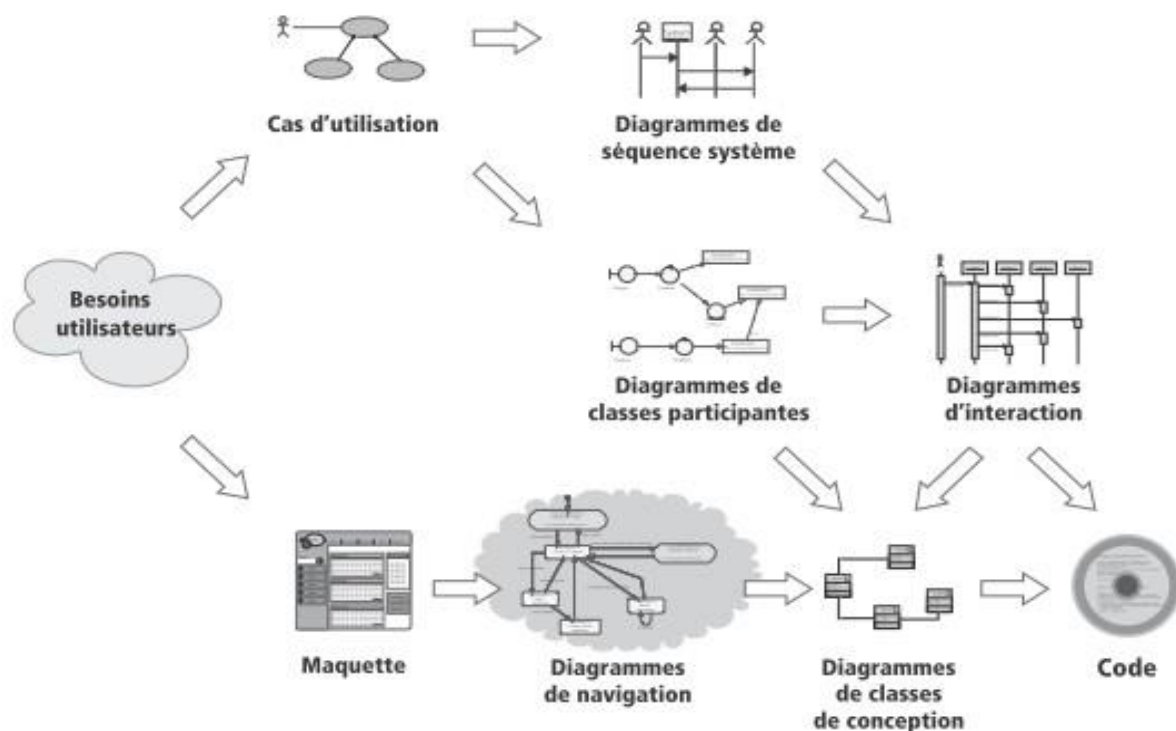


Figure 1-20 Schéma complet du processus de modélisation d'une application web

Chapitre 2 – Cahier des Charges

2.1 – Présentation de l'entreprise

SpotMyDive est une plateforme communautaire qui recense les sites de plongées sous-marine à travers le monde, pour permettre à des centres de plongées de proposer des offres liées à ces sites.

Le modèle économique se décompose en deux axes :

- Des frais de service de 15% sont pris à chaque réservation effectuée sur la plateforme
- Mise en place de campagnes de communication et vente d'espaces publicitaires

Le projet consiste à implémenter diverses fonctionnalités manquantes à la version actuelle :

- Implémentation d'une fonction d'affiliation à un type de plongée en particulier. En effet, il existe plusieurs organisations de formation, avec une démarche d'apprentissage propre à chacune. Les centres proposent donc un ou plusieurs types de cours basés sur ces organisations. Il fallait donc que le propriétaire d'un centre puisse définir lui-même quel type de cours il propose dans son centre.

C'est de cette partie-là dont je vais principalement devoir m'occuper, et que je vais vous présenter dans ce rapport.

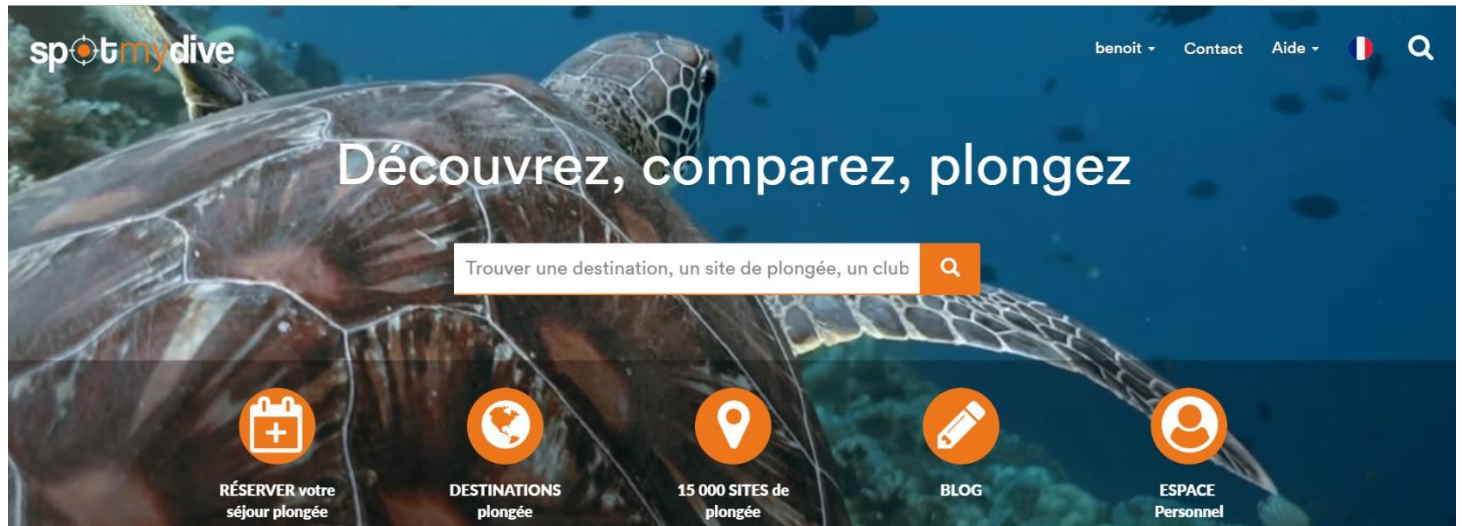
- Création d'un formulaire de contact pour pouvoir faire des demandes de devis en plusieurs étapes : un questionnaire qui s'adapte à la réponse précédente pour la question suivante.
- Mise en place des offres croisière. Proposer aux plongeurs la possibilité de réserver des croisières selon un large choix de destinations. Création d'un user-compagnie de bateau et d'un back office permettant de :
 - Gérer ses informations
 - Créer un ou plusieurs bateaux
 - Associer des offres
 - Ajouter une galerie photos

Ce nouveau développement renforce le modèle économique des frais de service.

- Modification de la vue actuelle de certaines pages du site.

- CF l'annexe n° 11.4 pour consulter le cahier des charges fournis par l'entreprise -

Screenshot de l'index du site



VOYAGE PLONGEE



2.2 – Technologies Imposées

Le projet a été développé sous **Symfony 2.6**, une version un peu différente de celle apprise en formation. Mais cela n'a pas été un obstacle à la compréhension de l'architecture, qui est très similaire à la version 3.

Symfony utilise la technologies **PHP**, ainsi que le **moteur de template Twig** pour le rendu de la vue.

La mise en forme est gérée principalement par **Bootstrap**, un framework de **CSS**, ainsi qu'une feuille de style complémentaire propre au projet.

La partie dynamique est propulsée par le framework **JavaScript**, **jQuery**, pour une plus grande fluidité de navigation.

Il m'a également fallut apprendre à travailler sous **Linux**, ce qui m'a permis d'approfondir mes connaissances.

Toute la gestion du projet s'est faite grâce à **Git**, un outil de gestion de version du code source décentralisé, permettant une cohésion de travail exemplaire, ainsi qu'une traçabilité du code source grâce à un historique permettant de revenir très facilement sur une version antérieure.

2.3 – Contexte de développement

Aucun poste de travail n'étant disponible au sein de l'entreprise, il m'a été demandé de travailler avec ma machine personnelle. J'ai donc installé l'environnement **Linux**, et plus particulièrement **Ubuntu 16.04**, version demandée par le développeur du projet.

Durant ce stage, j'ai travaillé en collaboration avec un autre stagiaire, élève de ma promotion M2i. Nous avons travaillé ensemble en nous répartissant les missions, avec les connaissances acquises lors de la formation que nous avons suivies.

La découverte du projet a duré une semaine.

Nous avons des points hebdomadaires sur **Skype** avec le programmeur en charge du projet, pour lui faire part de l'avancement de nos projets respectifs, lui exposer les problématiques rencontrées et ainsi nous aider à résoudre ces dernières.

N'ayant pas beaucoup de temps à nous accorder, un travail de recherche perpétuel a été fait sur toute la durée du stage.

Le jQuery a été une problématique que j'ai dû surmonter car c'est un chapitre de la formation qui a été survolé, j'ai dû apprendre son fonctionnement et ses subtilités, pour comprendre ce qui avait déjà été fait, et pouvoir mettre en place ce qui était à faire.

ENVIRONNEMENT DE DEVELOPPEMENT

LINUX UBUNTU 16.04 LTS	Pour le côté gestion et programmation
WINDOWS 7	Pour les tests
JETBRAINS PHPSTORM	Pour le développement, PHPStorm est un IDE (Integrated Development Environment), c'est un environnement de développement qui augmente la productivité
GITG	Pour la gestion de Git
STARUML	Pour la création de diagrammes simples
POWERDESIGNER (EX POWERAMC)	Pour la modélisation des traitements informatiques et des bases de données associées

Chapitre 3 - Analyse



3.1 – Diagrammes de Cas d'Utilisation

Recueillir, analyser et organiser les besoins, voilà l'objectif d'un Diagramme de Cas d'Utilisation.

A partir d'un document fourni par l'entreprise, j'ai établi les diagrammes nécessaires à la conception du système d'affiliations de plongée. (CF l'annexe 11.4)

Un **DCU** est un moyen simple d'exprimer le besoin d'un utilisateur, en illustrant les fonctionnalités de l'application dont il aura besoin, via des schémas très compréhensibles.

Ces derniers débutent généralement par un diagramme de cas d'utilisation, qui est un diagramme **UML** qui sert à donner une vision simple et globale du comportement d'une application ou d'un logiciel.

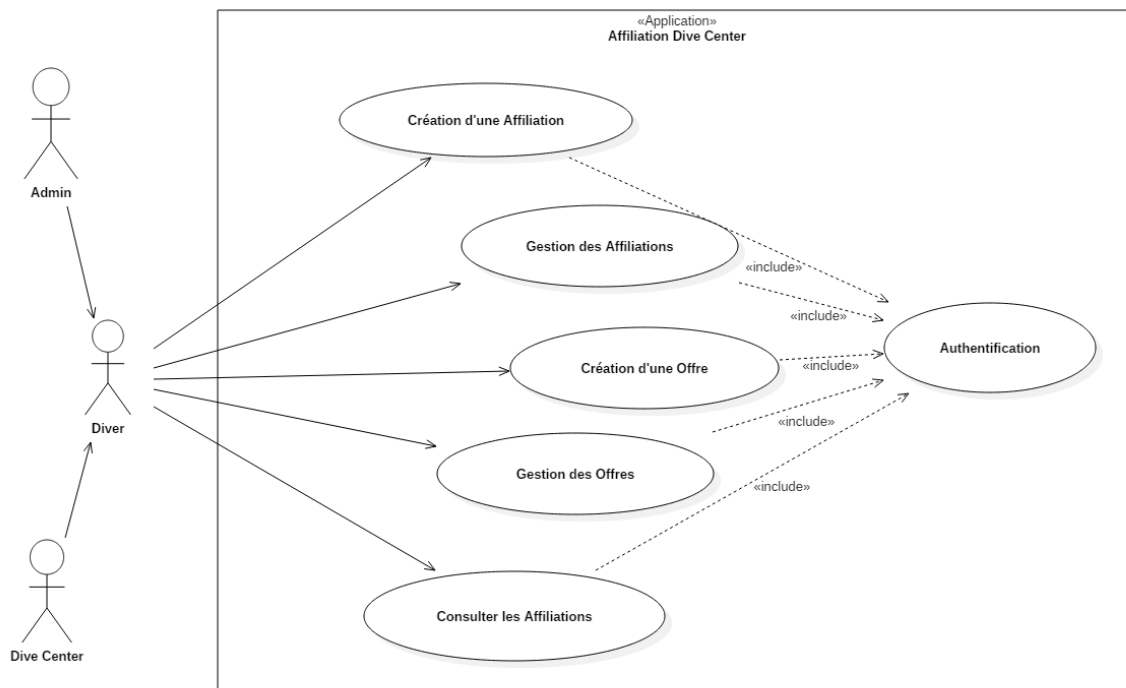
J'ai d'abord défini les différents **acteurs** représentant les utilisateurs intervenant sur le **système**, puis j'en ai déduit les différents **rôles** qui leur seront attribués.

3.1.1 – Représentation graphique

Ces diagrammes mettent en avant 3 **acteurs** :

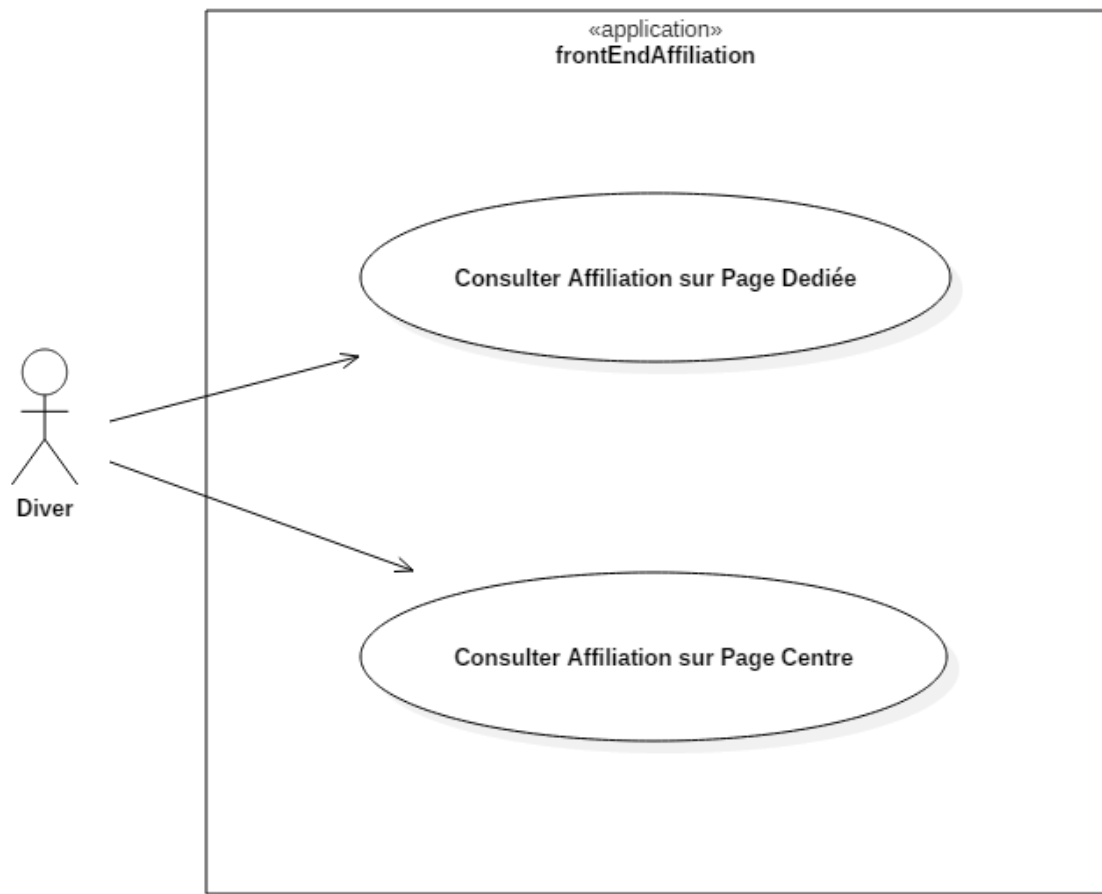
1. L'administrateur du site (Admin)
2. Le propriétaire d'un Centre de Plongée (Dive Center)
3. Le plongeur (Diver)

Voici une représentation simple d'un DCU basé sur l'application « Affiliation », qui représente tous les acteurs.



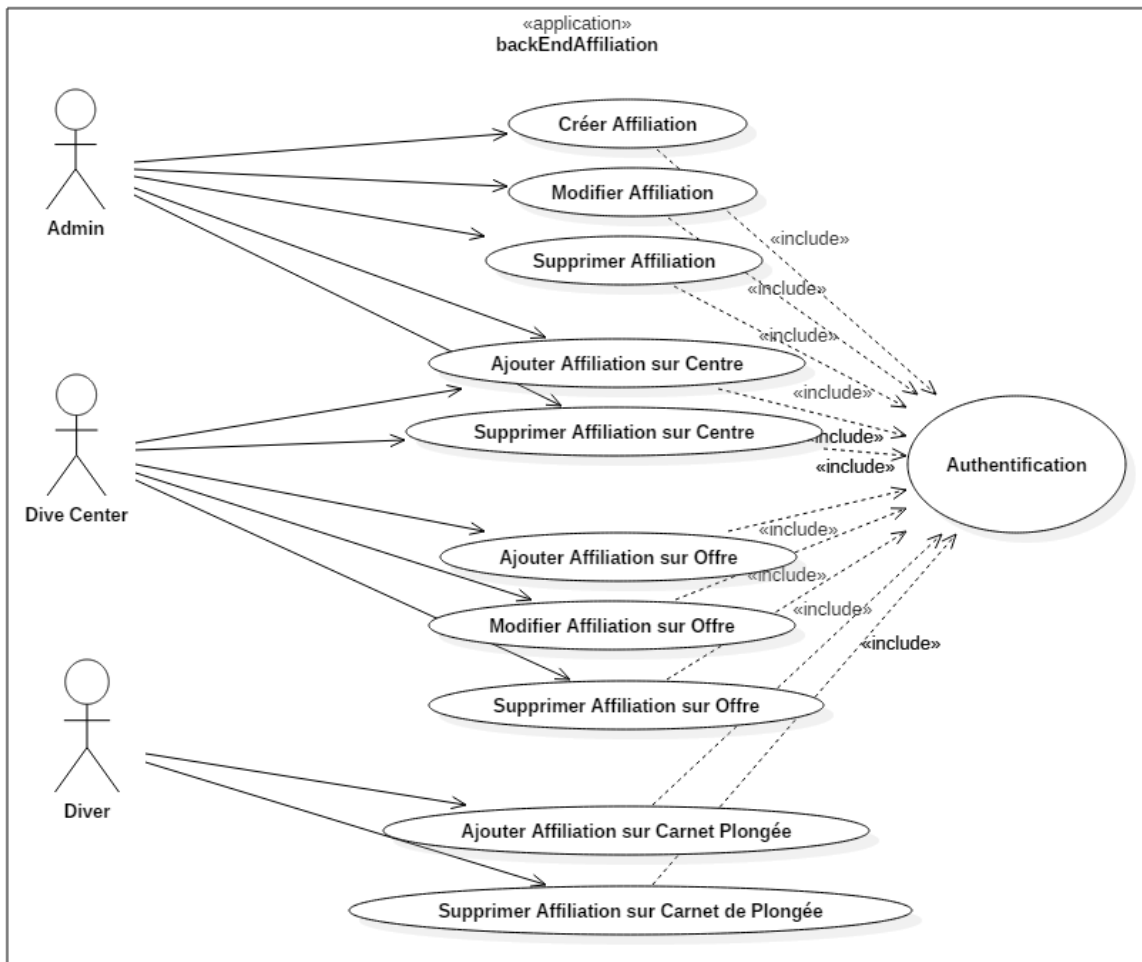
3.1.2 – DCU Front-End

Le diagramme ci-dessous représente l'affiliation coté front, c'est-à-dire la partie « visible » du site internet, et représente l'**acteur** Diver.



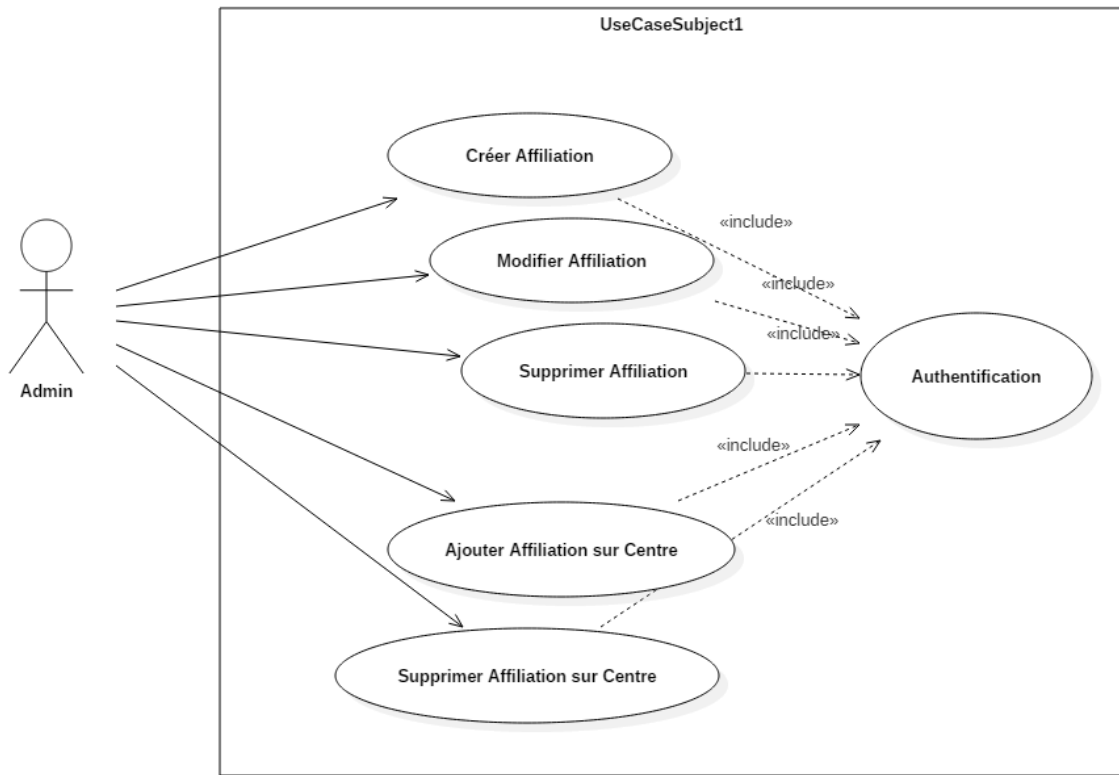
3.1.3 – DCU Back-End

Le diagramme ci-dessous représente l'affiliation coté Back, c'est-à-dire la partie « administrative » du site internet, et met en avant les 3 acteurs.



3.1.4 – DCU Admin

Ce cas d'utilisation est spécifique à l'acteur « Admin », ce sont les fonctionnalités qu'il aura à disposition une fois l'application terminée.



3.1.5 – Fiche de description textuelle d'un Cas d'Utilisation

La fiche de description textuelle d'un cas d'utilisation est très utile pour clarifier le déroulement et décrire les actions qui seront réalisées par une fonctionnalité.

La fiche ci-dessous vous décrit le scénario nominal de « l'Ajout d'une Affiliation » pour un Admin.

ÉTAPES	DESCRIPTION
Identification du CU	<p>Titre : Créer les affiliations</p> <p>Résumé : Remplir l'affiliation liée à un centre de plongée</p> <p>Acteurs : Administrateurs, Centre de Plongée</p> <p>Date de création : 20/03/2017</p> <p>Date de mise à jour : 29/03/2017</p> <p>Version : 1.1</p> <p>Auteur : Benoît Catillon</p>
Pré-conditions	<p>L'utilisateur est identifiée</p> <p>La base de données est disponible</p>
Scénario nominal	<p>1 – Cliquer sur « Ajouter une affiliation »</p> <p>2 – Remplir les différents champs requis</p> <p>3 – Choisir l'affiliation correspondante</p> <p>5 – Ajouter une image</p> <p>4 – Valider le formulaire</p> <p>5 – Afficher le message de validation</p>
Scénarii d'erreurs du cas nominal	<p>1 – Un ou plusieurs champs du formulaire vide</p> <p>2 – Nom de l'affiliation déjà existante</p> <p>3 – Erreur lors de l'envoi des données en BD</p> <p>4 – Afficher le message d'erreur correspondant</p>
Post-conditions	<p>Les informations ont été persistées en base de données.</p> <p>Un message confirme la persistance.</p>
IHM	Formulaire



3.1.6 – Diagramme d'Activités

Un diagramme d'activités est un diagramme comportemental qui permet de modéliser le cheminement de l'activité de l'utilisateur, en représentant graphiquement son parcours.

Les diagrammes d'activités sont réalisés avec PowerDesigner 16.5 de Sybase.

Diagramme d'Activités
Création Affiliation

Le gros rond plein au sommet du diagramme est le **pseudo état initial** de cette partie de l'application.

Le **pseudo état final** est à son opposé, c'est le rond plein entouré présent au bas du diagramme.

S'en suit une **transition**, représentée par une flèche bleue, pointant vers une activité.

Chacune des **activités** sont représentées par un rectangle bleu aux coins arrondis.

Les **conditions**, ou « **gardes** », sont représentées par un losange jaune. Elles permettent de mettre en avant les éventuels rejets du système.

Le tout forme un **flot d'objets**.

Cette représentation nous permet de vraiment définir les bases de ce qui sera codé.

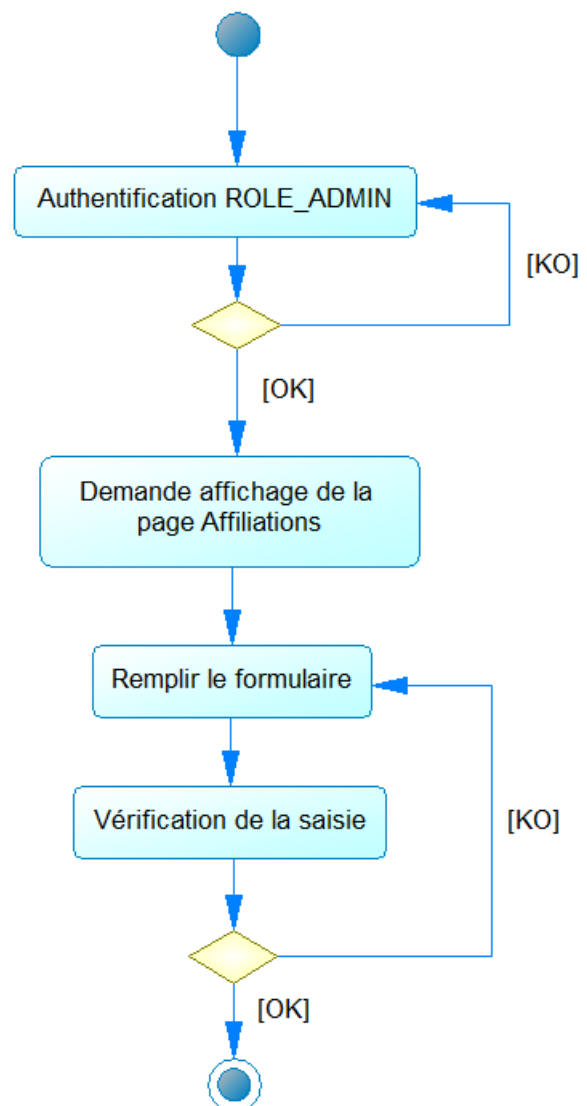
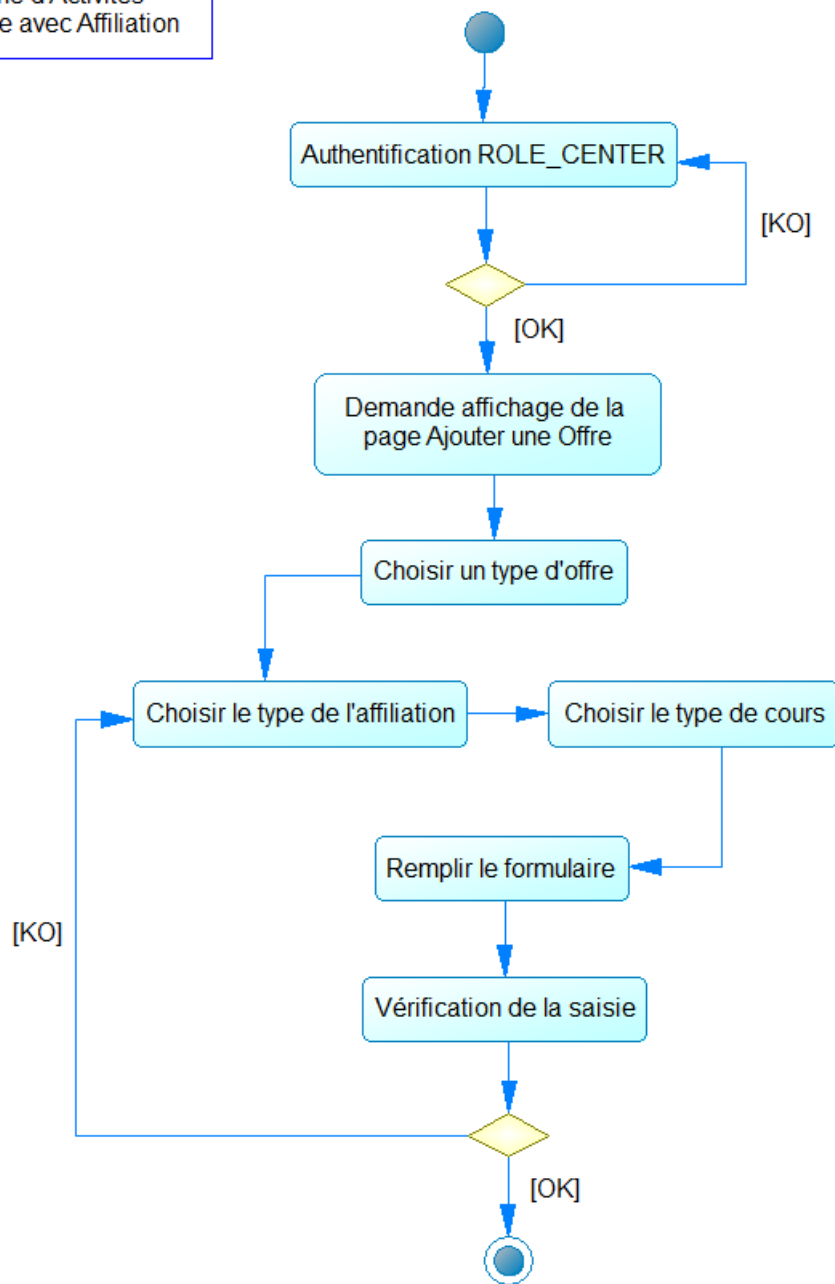


Diagramme d'Activités
Création Offre avec Affiliation



3.2 – Les maquettes

Super-Administration SpotMyDive

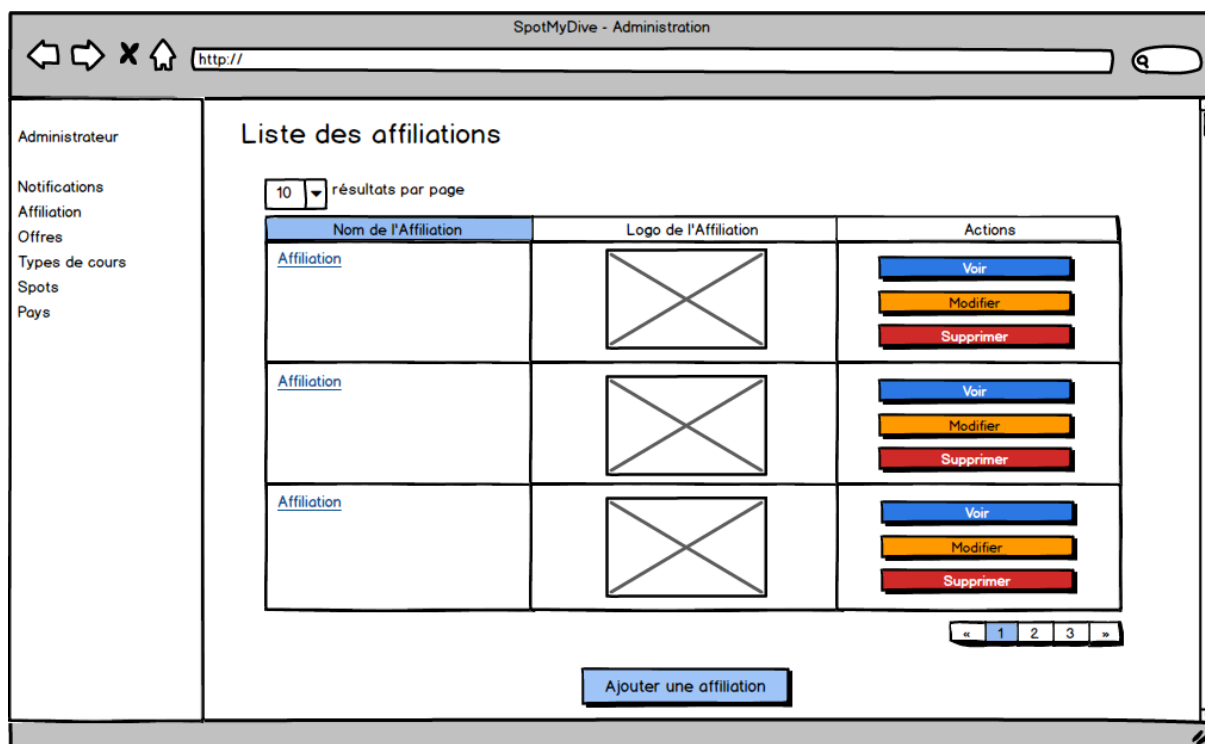
Voici la page d'accueil du menu « Affiliation » du dashboard d'un Admin.

Sur cette page, l'administrateur aura un **listing des affiliations** qu'il a précédemment ajoutées.

Pour chaque affiliation, il aura la possibilité de la consulter, de la modifier, et de la supprimer.

Il existe un menu déroulant pour choisir le nombre de résultat par page, ainsi qu'une pagination s'il y a plus d'affiliations qu'il n'y a de résultats.

Tout en bas de la page, un bouton qui mène à la création d'une nouvelle affiliation.





Super-Administration SpotMyDive

Voici la page de création d'une affiliation.

L'administrateur entre dans le champ « Nom » le nom de l'affiliation qu'il souhaite ajouter. Ensuite il sélectionne dans le menu déroulant le type de l'affiliation qui doit lui être associé.

Il existe deux types d'affiliation : La Plongée et l'Apnée

Puis il ajoute le logo de l'affiliation, en téléchargeant l'image depuis son ordinateur.

Il existe un bouton « Créer une Affiliation » pour valider l'ajout, ainsi qu'un bouton « Retour à la liste », s'il souhaite annuler son action et revenir au listing.

The screenshot shows a web browser window titled "SpotMyDive - Administration". The address bar shows "http://". On the left is a sidebar menu with the following items: Administrateur, Notifications, Affiliation, Offres, Types de cours, Spots, and Pays. The main content area is titled "Création d'une affiliation". It contains the following form elements: a text input field for "Nom", a dropdown menu for "Type de l'affiliation" with "Plongée" selected and "Apnée" as an option, and a file upload section for "Logo" with a button labeled "Choisissez votre fichier". At the bottom right of the form are two buttons: "Créer une Affiliation" and "Retour à la liste".

Administration Centre de Plongée

Voici la page d'accueil d'ajout d'une offre du propriétaire d'un centre de plongée.

Sur cette page, il pourra retrouver toutes les offres qu'il aura ajouté précédemment, ainsi qu'en ajouter.

Elles seront classées par type d'affiliation, Plongée et Apnée.

Comme pour l'écran du super-admin, il aura la possibilité de consulter, de modifier, et de supprimer chacune de ses offres.

Un bouton « Ajouter une offre » est présent en haut de page.

SpotMyDive - Administration Centre

Admin Centre
Infos centre
Mes offres
Mes spots de plongée
Galerie photo

Mes offres

Ajouter une offre

Liste des offres

Baptême de plongée

Nom de l'Affiliation	Date	Prix	Actions
Affiliation	du 01-01-2017 au 31-01-2017	à partir de 99 EUR	Modifier Supprimer
Affiliation	du 01-01-2017 au 31-01-2017	à partir de 99 EUR	Modifier Supprimer

Apnée

Nom de l'Affiliation	Date	Prix	Actions
Affiliation	du 01-01-2017 au 31-01-2017	à partir de 99 EUR	Modifier Supprimer
Affiliation	du 01-01-2017 au 31-01-2017	à partir de 99 EUR	Modifier Supprimer

Administration Centre de Plongée

Pour finir, voici la page de création d'une offre d'un centre de plongée.

C'est sur cette page que le propriétaire d'un centre va pouvoir ajouter une offre, en fonction du type de cours qu'il propose.

Il existe 5 types de cours :


- Baptême de Plongée
- Plongée Loisirs
- Cours de Plongée (en lien avec une affiliation)
- Apnée (en lien avec une affiliation)
- Snorkeling

L'exemple pointe sur « Cours de Plongée ».

Le propriétaire d'un centre va tout d'abord choisir le type de l'affiliation spécifique à un Cours. Puis un nouveau champs « Cours » apparait, avec un menu déroulant qui proposera les types de cours liés au type de l'affiliation choisi précédemment.

Il devra ensuite renseigner le nom de l'offre, une courte description, ainsi que le tarif du cours.

Le bouton « Créer » ajoutera cette offre, et sur le listing de son dashbord, et sur la page de son centre.



SpotMyDive - Administration Centre

Admin Centre

Infos centre

Mes offres

Mes spots de plongée

Galerie photo

Création d'une offre

[Baptême](#) [Loisirs](#) **[Cours Plongée](#)** [Apnée](#) [Snorkeling](#)

Cours de Plongée

Affiliation

Cours

Nom de l'offre

Description

Tarifs (EUR)

Créer



3.3 – Diagramme de Navigation

Un diagramme de navigation sert à représenter le cheminement de l'application entre les différentes vues.

Le **début de la navigation** est représenté par le **rond noir**.

Les **transitions** sont représentées par une **flèche** qui pointe vers un **état**, comme celle qui part du début de la navigation pour arriver à la page d'accueil de l'administration.

C'est la **Route** qu'empruntera Symfony pour accéder à la page, guidé par le **Controller**.

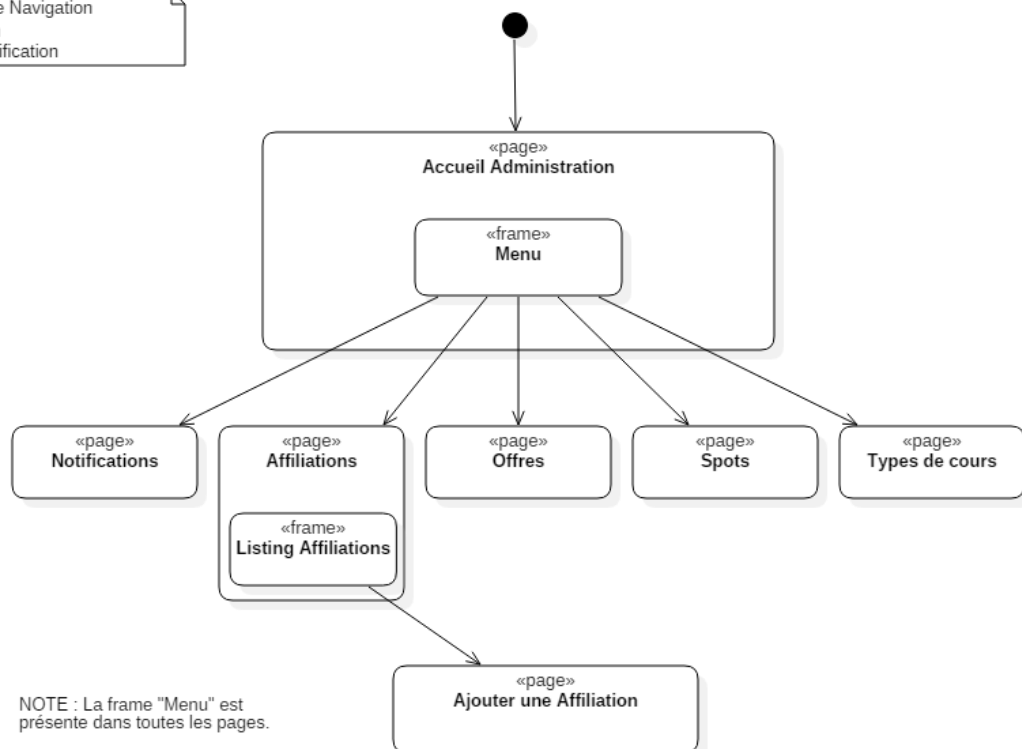
Un **état** est un **écran**, une **page**, qui est représenté par un rectangle avec les coins arrondis.

L'utilisateur arrive sur la page d'accueil, sur laquelle est présent un menu.

Il clique sur le bouton du menu « Affiliation », qui contient le listing des affiliations déjà ajoutées, ainsi que le bouton de création d'une affiliation.

Il clique ensuite sur ce bouton de création pour arriver au formulaire.

Diagramme de Navigation
Administration
Après authentification



Chapitre 4 – Conception de la Base de Données

4.1 – Introduction

Une base de données sert à stocker des informations pour ensuite être capable de les traiter, afin d'en extraire les données.

Pour accéder à la base de données, nous allons soumettre des requêtes en utilisant le langage **SQL** au SGBDR.

SGBDR = Systèmes de Gestion de Bases de Données Relationnelle.

Les SGBDR sont les programmes qui se chargent du stockage de données.

La base de données travaille avec des **tables**. Chaque table est composée de champs que l'on appelle des **colonnes**, et chaque table contient de 0 à plusieurs enregistrements qu'on appelle des **lignes**.

La base de données utilisée par SpotMyDive est gérée par **MySQL**, et les tables sont basées sur le moteur **InnoDB**.

Celui-ci prend en charge les clefs étrangères, et la gestion des transactions. (COMMIT et ROLLBACK)

Les **clés primaires** servent à identifier une ligne de manière unique.

Les **clés étrangères** permettent de gérer des relations entre plusieurs tables, garantissent la cohérence des données, et l'intégrité référentielle.

4.2 – La démarche utilisée

La modélisation conceptuelle est une étape fondamentale dans la démarche de conception d'une base de données.

C'est cette dernière que j'ai utilisée pour déterminer à quoi doit ressembler la BD, ses tables, ses attributs, son contenu général, ainsi que les relations obligatoires avec l'application principale existante, via un diagramme de classes.



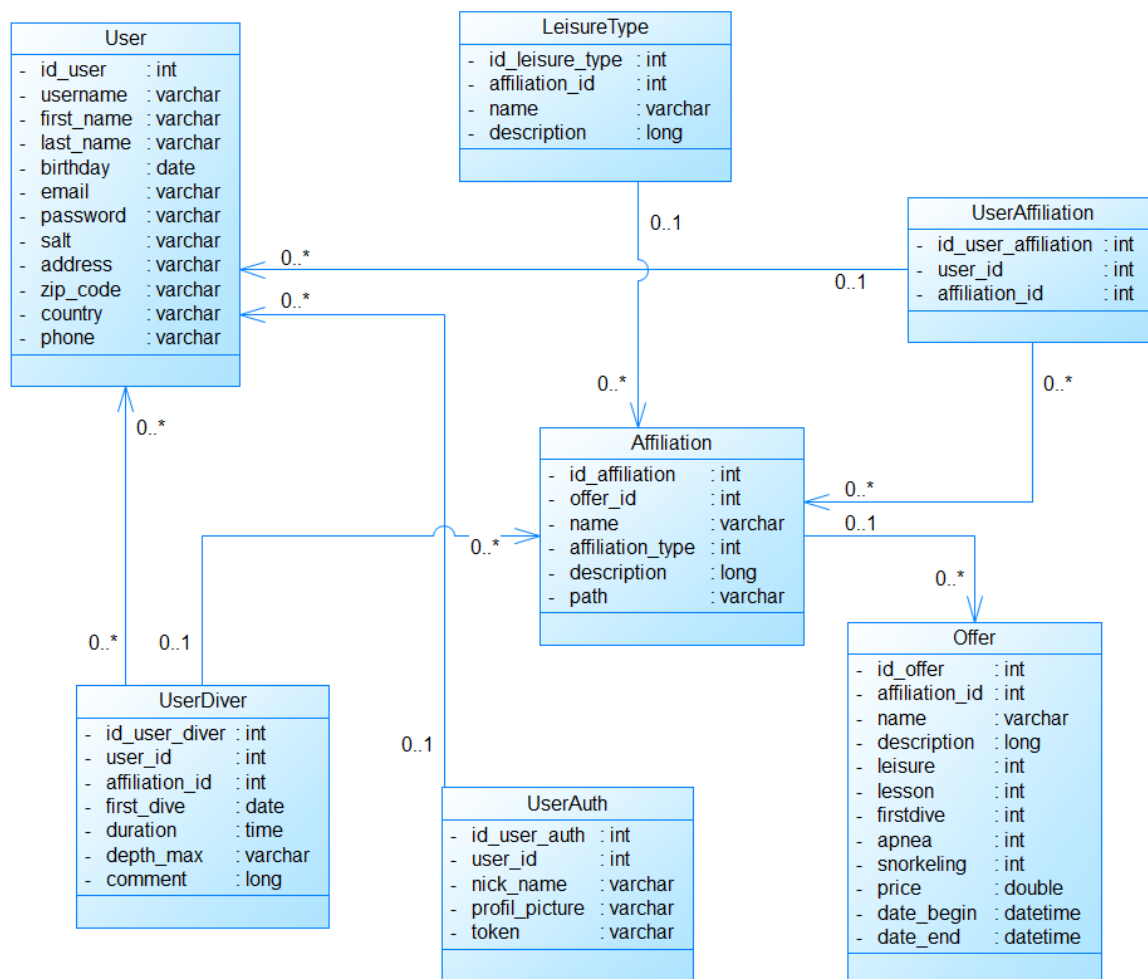
4.3 – Diagramme de Classes

Un diagramme de classes est un diagramme statique d'UML.

Il schématise les classes et les interfaces du système ainsi que ses différentes relations (multiplicités).

Une classe est un modèle de données, une sorte de moule, qui va nous permettre de créer des « objets », avec une structure commune à ces derniers.

C'est elle qui définit les propriétés d'un objet qu'elle va créer.





4.4 – Modélisation Physique des Données

A partir du diagramme de classes obtenu précédemment, nous pouvons à présent générer un modèle physique de données.

Symfony dispose d'un **ORM** (Object **R**elational **M**apping) appelé DOCTRINE, 2^{ème} du nom.

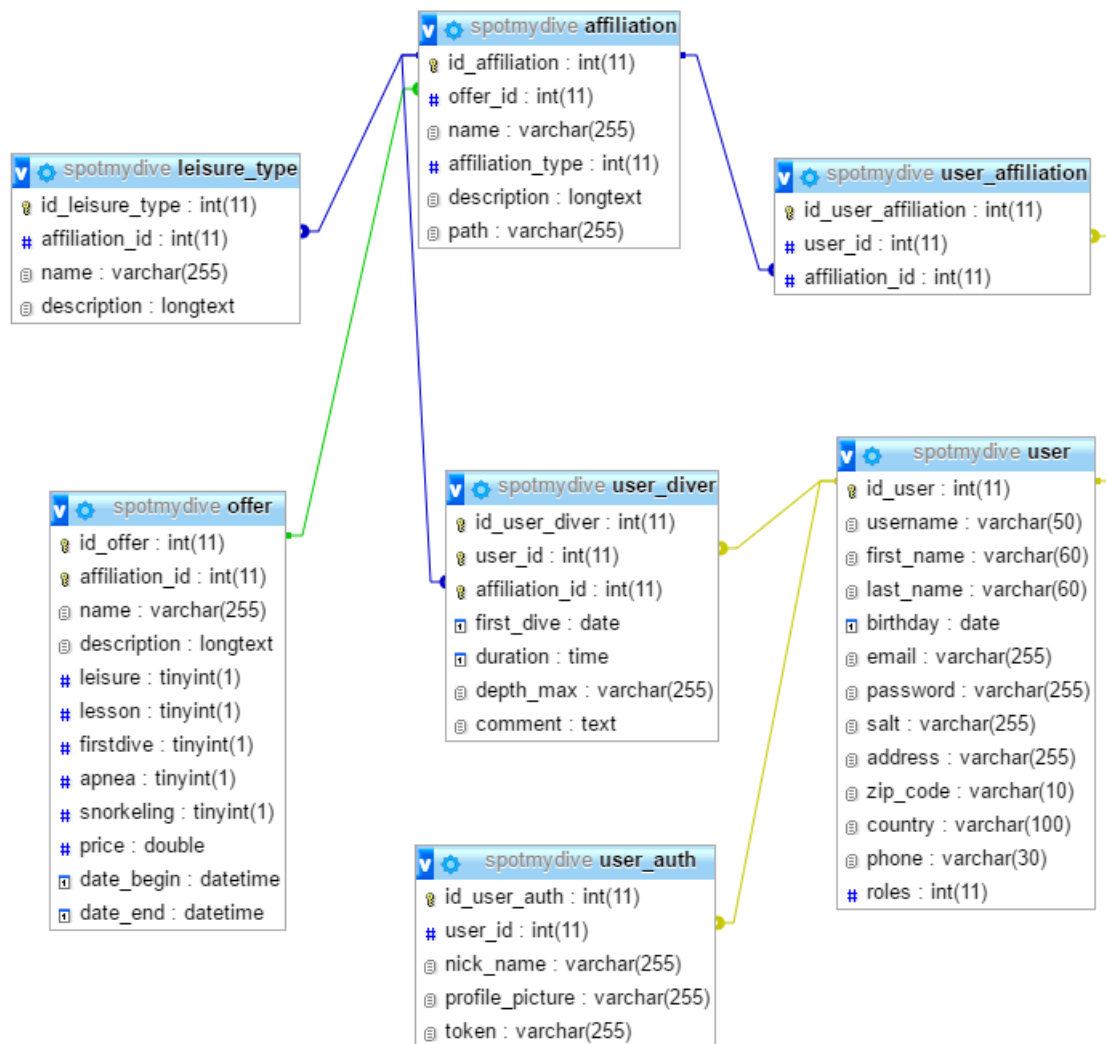
Doctrine2 nous permet de générer une base de données très simplement, dans une console.

Pour cela, nous allons d'abord générer l'**Entité** qui y sera liée, puis créer la base de données à partir de ce modèle.

Voici le résultat obtenu, modélisé avec **phpMyAdmin**, une interface permettant la gestion de la base de données MySQL.



Schéma de la Base de Données



4.5 – Doctrine2



Doctrine est un ORM qui permet de manipuler la base de données au travers d'objets. Un ORM est une couche d'abstraction de base de données, ce qui permet de pouvoir changer de base de données sans avoir à changer la méthode de requêtage.

Un **ORM**, c'est comme une interface entre la base de données et l'utilisateur. Le but est d'accéder au contenu de la base de données avec un langage dit « **orienté objet** », ce qu'utilise **Symfony** avec **PHP**.

Il va se charger de l'enregistrement des données dans la base.

Avec Doctrine, tout est objet, ce qui permet de manipuler les données en tant que telles, et facilite grandement le travail du codeur.

Il gère la connexion à la base de données, l'instanciation des objets, le **Create / Read / Update / Delete (CRUD)**, et permet d'avoir à écrire moins de requêtes SQL dans les cas les plus courants (find, findBy, findAll...) ou d'unifier les requêtes SQL sans se préoccuper du moteur de BD.

Même s'il facilite beaucoup de choses, **les connaissances en SQL sont fondamentales pour savoir comment manipuler l'ORM**, il est d'ailleurs possible de visualiser une requête en SQL avant de laisser Doctrine s'en charger, via une commande.

Quelques commandes indispensables qui permettent de manipuler Doctrine :

ACTION	COMMANDE
Génération d'une entité	<code>php app/console doctrine:generate:entity</code>
Mises à jours des entités	<code>php app/console doctrine:generate:entities</code>
Création de la base de donnée	<code>php app/console doctrine:database:create</code>
Création des tables	<code>php app/console doctrine:schema:create</code>
Visualiser la requête en SQL avant qu'elle ne soit envoyée	<code>php app/console doctrine:schema:update --dump-sql</code>
Mettre à jour les tables	<code>php app/console doctrine:schema:update --force</code>
Création d'un formulaire	<code>php app/console generate:doctrine:form</code>
Générer un CRUD (Controller/Form/View)	<code>php app/console generate:doctrine:crud</code>



4.6 – SQL : LDD

Le SQL est un langage informatique normalisé permettant d'exploiter des bases de données relationnelles.

Il se décompose en 4 sous-langages :

- **LDD** : Langage de Définition de Données
Exemple : CREATE, ALTER, DROP
- **LCD** : Langage de Contrôle de Données
Exemple : GRANT, REVOKE
- **LMD** : Langage de Manipulation de Données
Exemple : INSERT, UPDATE, DELETE, SELECT
- **LCT** : Langage de Contrôle de Transactions
Exemple : COMMIT, ROLLBACK

Une **BDR** (Base de Données Relationnelle) est un ensemble de tables souvent reliées entre elles, même s'il peut exister des tables indépendantes, qui modélisent un domaine du système d'information.

Une table est composée de colonnes et de lignes.

Une table doit posséder une clé primaire (PRIMARY KEY), composée d'une ou plusieurs colonnes, ce qui permet d'identifier chaque ligne.

Chaque valeur est unique et doit être renseignée (elle est **NOT NULL**). La clé primaire est indexée.

Une table peut comprendre zéro, une ou plusieurs clé(s) étrangère(s) (FOREIGN KEY) qui sont des colonnes correspondant à une clé primaire dans une autre table. Une clé étrangère dans une table (enfant) permet de faire un lien vers une autre table (une table parent). Mais c'est surtout une **contrainte (CONSTRAINT)**, dans la mesure où les valeurs de la clé primaire de la table parent sont **références (REFERENCES)** pour la table enfant.

On parle de **contrainte d'intégrité référentielle**.

Une valeur dans la colonne de la clé étrangère de la table enfant doit nécessairement correspondre à une valeur présente dans la colonne clé primaire de la table parent.

4.6.1 – Création de la Base de Données

```
CREATE DATABASE IF NOT EXISTS `spotmydive`  
DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;  
  
USE `spotmydive`;
```

4.6.2 – Structure et création de la Table « affiliation »

```
CREATE TABLE affiliation (  
  id_affiliation int(11) NOT NULL AUTO_INCREMENT,  
  offer_id int(11) NOT NULL,  
  `name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,  
  affiliation_type int(11) NOT NULL,  
  description longtext COLLATE utf8_unicode_ci,  
  path varchar(255) COLLATE utf8_unicode_ci NOT NULL,  
  PRIMARY KEY (id_affiliation),  
  KEY offer_id (offer_id),  
  CONSTRAINT affiliation_ibfk_1  
  FOREIGN KEY (offer_id)  
  REFERENCES offer (id_offer)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

LDD généré avec MySQL WorkBench

- CF l'annexe 11.5 pour consulter le code complet -

Chapitre 5 – Conception de l'Application



5.1 – Diagramme de Séquence Système

Le **Diagramme de Séquence Système** décrit le comportement du système qui découle d'un Cas d'Utilisation définit précédent.

Il formalise les **interactions** entre les **acteurs** et le **système** ou les **classes**, grâce à des **messages**, avec un point de vue chronologique et spatial, le temps étant représenté à la verticale, et l'espace à l'horizontal.

Il représente généralement deux objets : **l'utilisateur du système**, et le **système**.

Chaque objet est représenté par un rectangle, sous lequel se dessine une ligne en pointillés, qui descend vers le bas depuis son centre.

On appelle cela une **ligne de vie**.

Les **messages** sont représentés par des flèches horizontales, pleines ou en pointillés, de l'émetteur vers le destinataire.

Il existe 5 types de message : le **CALL** (synchrone), le **SEND** (asynchrone), le **RETURN** (synchrone), le **CREATE** et enfin le **DESTROY**, ce qui permet de bien définir chaque message.

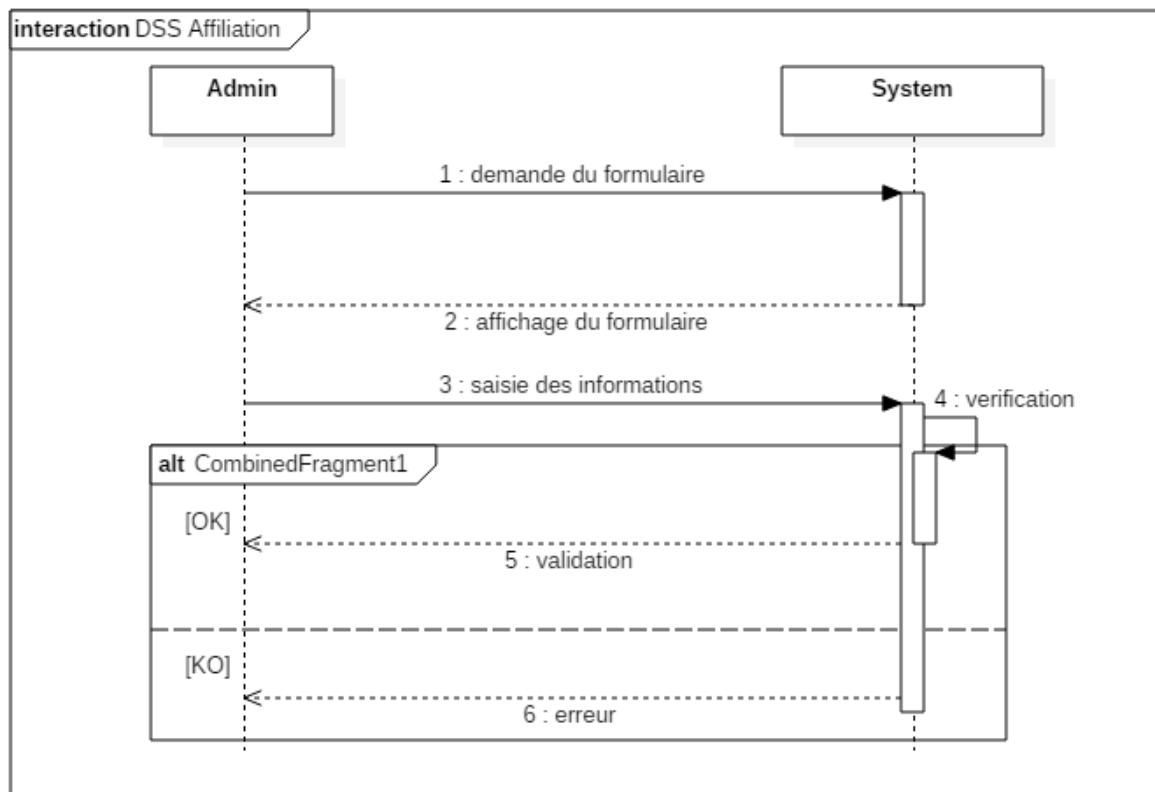
Les **messages CALL**, représentés par une flèche pleine et fermée, pointe vers une **activation**, représentée par un rectangle verticale et fin, positionnée sur la ligne de vie.

Les **messages SEND** sont eux représentés par une flèche pleine et ouverte, ils pointent généralement vers la ligne de vie d'un acteur.

Les **messages RETURN** sont tout simplement les réponses de messages CALL, représentés par une flèche en pointillés et ouverte.

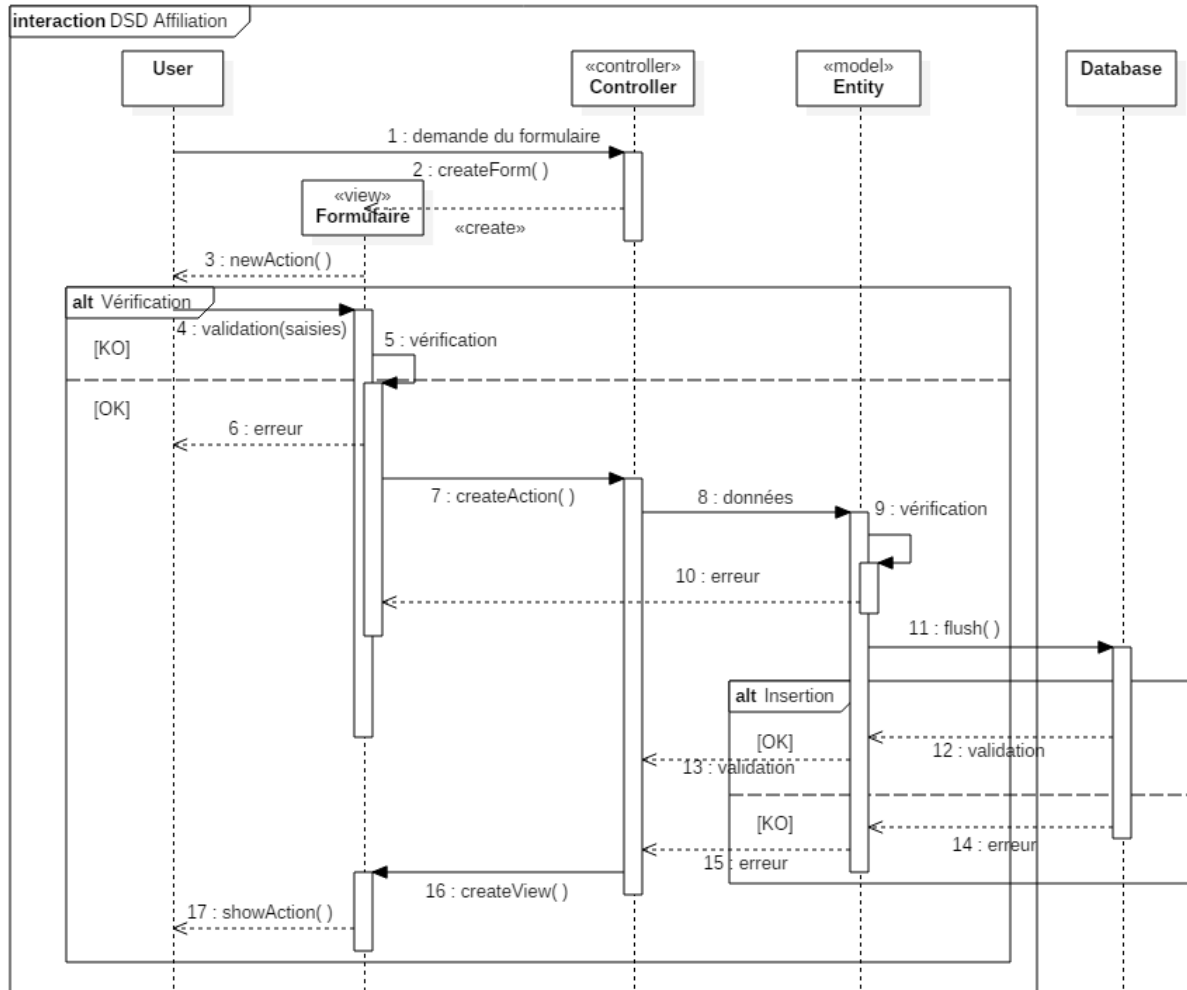
Les **messages CREATE et DESTROY** sont représentés par une flèche pleine et fermée, la ligne de vie d'un DESTROY se terminant par une croix.

Le DSS ci-dessous représente un CU définit précédemment, la création d'une affiliation.





5.2 – Diagramme de Séquence Détaillé



Chapitre 6 – Développement

6.1 – Technologies Utilisées

6.1.1 – Symfony

La principale technologie qu'utilise SpotMyDive est le **PHP**, sous le framework **Symfony 2.6**.

Un framework, ou cadre de travail, est un ensemble d'outils, un regroupement de composants qui aide le programmeur dans sa tâche de développement, tout en l'obligeant à adopter les bonnes pratiques du développement d'un projet.

Symfony est l'un des meilleurs framework PHP disponible gratuitement, il a été développé par **SensioLabs**, une entreprise française.

Comme dit précédemment, il embarque Doctrine2, qui permet la gestion de la base de données.

6.1.2 – jQuery

jQuery est un framework **JavaScript**.

Il permet d'exploiter la puissance de JavaScript pour agir dynamiquement et en temps réel sur une page, mais également de profiter d'une plus grande compatibilité avec les différents navigateurs.

Cette bibliothèque libre et gratuite a été créée dans le but de faciliter l'écriture de scripts JavaScript. D'ailleurs, son slogan est « write less, do more », ce qui le résume très bien !

Ce framework regroupe un ensemble de fonctions JavaScript, dans un seul fichier, ce qui permet de l'obtenir rapidement en téléchargeant un simple fichier très léger, ou de l'appeler sur un serveur distant depuis la page HTML.

Il n'est pas indispensable d'avoir de bonnes bases avec JavaScript pour utiliser jQuery, mais il est préférable d'en connaître les concepts.

6.2 – Architecture MVC

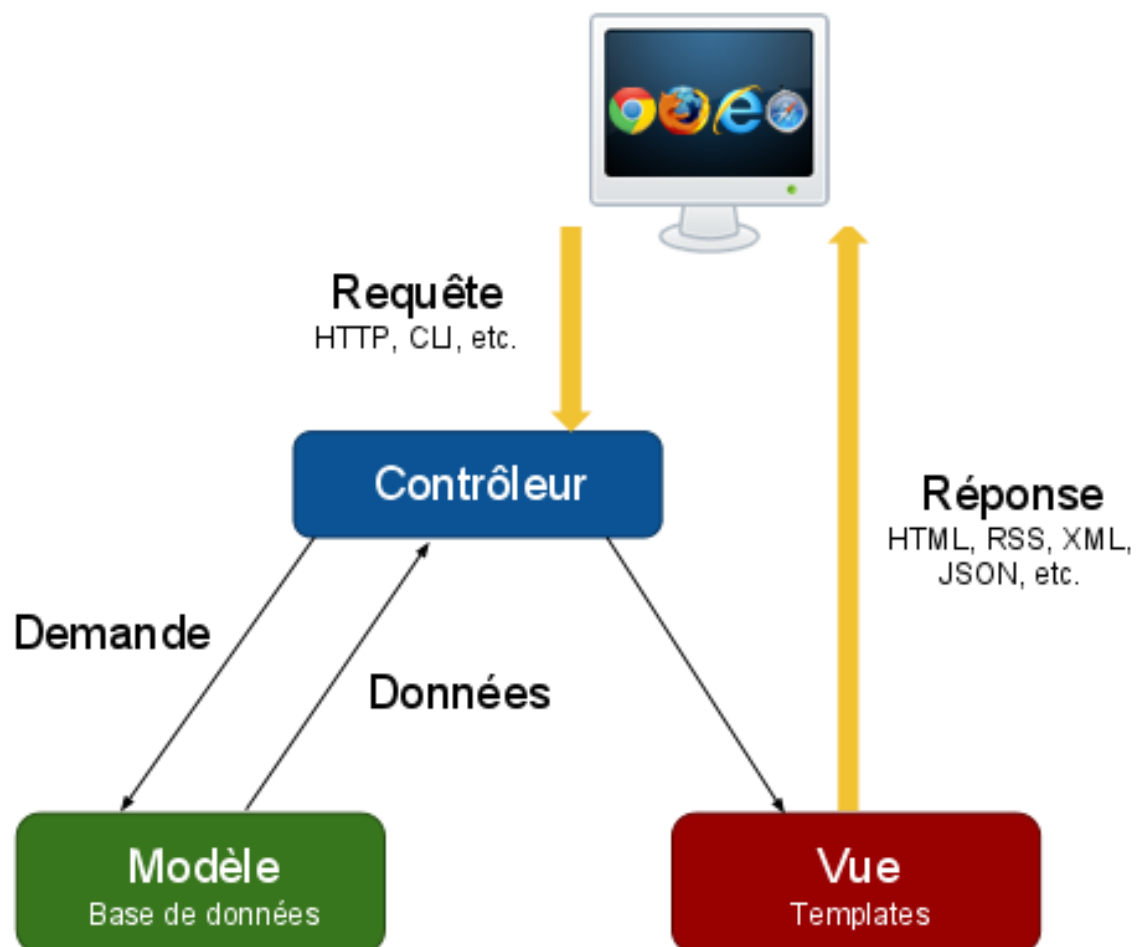
MVC signifie **Model View Controller**.

Le concept de l'architecture MVC est simple, il découpe une application en 3 parties distincte, ce qui permet d'organiser un projet, gros comme petit, de manière à s'y retrouver très facilement.

La séparation des couches est faite comme ceci :

- **Le Contrôleur** : il détermine l'action en fonction de l'URL, il est en quelque sorte le chef d'orchestre, il analyse la requête, effectue les contrôles puis va appeler le Modèle concerné.
- **Le Modèle** : il se charge de récupérer les informations en base de données suite à l'appel du Contrôleur, pour finalement lui renvoyer.
- **La Vue** : c'est elle qui va s'occuper du rendu final dans le navigateur web. Le Contrôleur récupère les données du Modèle pour les passer à la Vue concernée.

Symfony utilise cette architecture.



6.3 – Les Bundles Symfony

Un bundle est un ensemble de fichiers et de répertoires permettant d'implémenter une ou des fonctionnalités qui peuvent être utilisées dans d'autres projets.

Il existe des bundles communautaires « tout prêt », que l'on peut donc implémenter sur un projet en cours, comme FOSUserBundle, un des plus connus, qui permet la gestion complète des utilisateurs d'un site web, y compris la partie administrative.

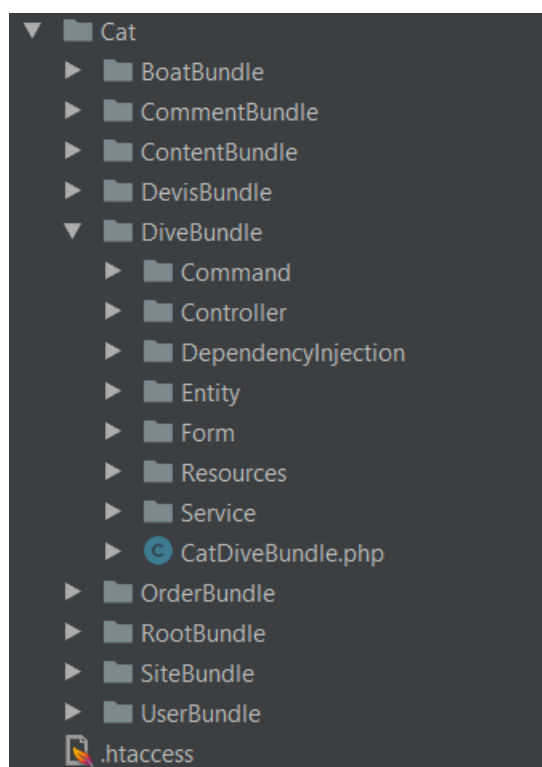
C'est d'ailleurs un bundle utilisé par SpotMyDive.

Il est bien entendu possible de créer un bundle soit même, que ce soit pour une utilisation propre à un projet en particulier, ou pour le proposer à la communauté afin d'en faire profiter le plus grand nombre !

Dans le projet SpotMyDive, un bundle principal a été créé, Cat, comprenant des sous-bundles.

NOTE : Cat aurait dû, selon la convention de nommage recommandé par l'éditeur de Symfony, s'appeler « CatBundle »

Voici le contenu du bundle Cat, ainsi que le contenu d'un de ses sous-bundles, DiveBundle :



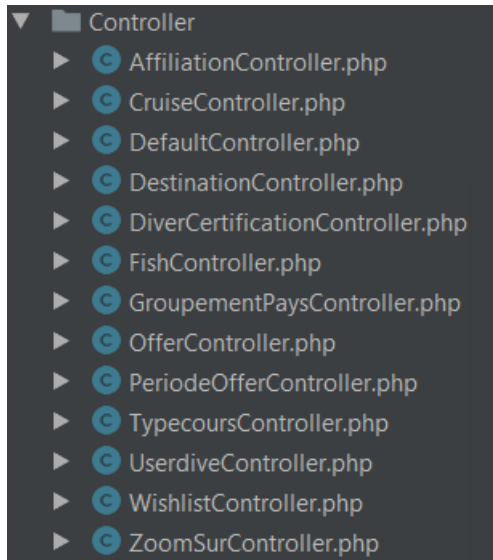
Pourquoi DiveBundle ?

Parce que c'est dans ce bundle que l'on m'a demandé de mettre en place l'Affiliation.



Nous allons à présent nous intéresser à son contenu, et plus particulièrement 4 dossiers :

- Controller
- Entity
- Form
- Resources

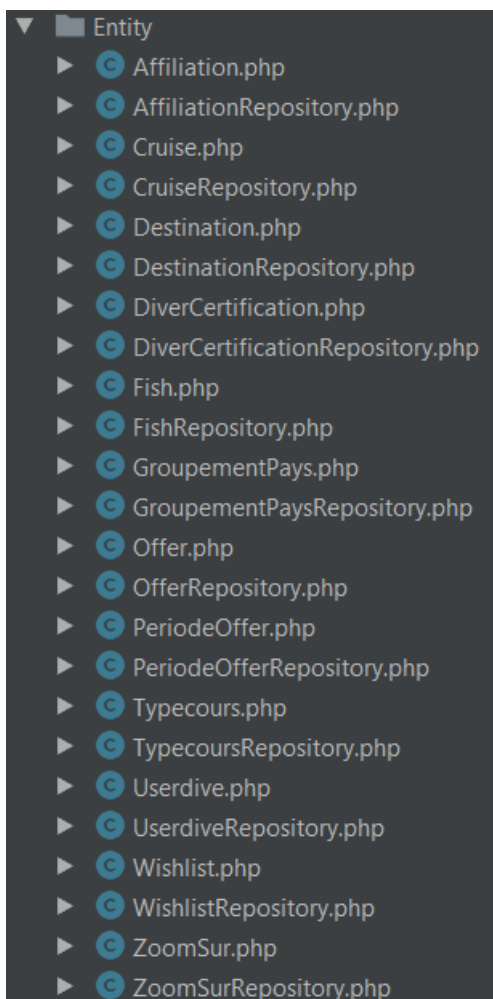


6.3.1 – Controller

Il contient tous les contrôleurs du projet.

Un Controller va permettre à une requête d'être acheminée vers une réponse. C'est tout simplement une méthode dans une classe PHP, qui va permettre d'utiliser des services, des modèles, et d'appeler la vue.

Le nom du fichier du contrôleur commence toujours par le nom de la classe, et se termine par le suffixe « Controller »



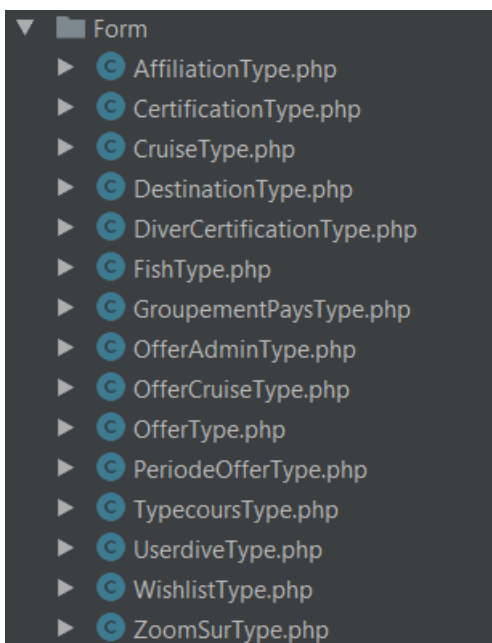
6.3.2 – Entity

Il contient toutes les entités du projet.

Une Entité est un objet qui représente une table dans la base de données. C'est un objet qui va permettre d'enregistrer une nouvelle ligne, ou d'en modifier une. Doctrine2 va se servir de l'Entity Manager pour requêter dans la base de données, soit par des méthodes préétablies (find, findBy, findAll), soit par des custom methods définies dans le Repository correspondant à l'Entité.

L'entité contient des propriétés, ainsi que des accesseurs et des mutateurs (getter/setter) correspondant le plus généralement aux champs de la table. Il contient également toutes les relations entre les entités **et qui relient** les relations entre les tables dans la base de données. (FOREIGN KEY)

Le Repository sera l'interface entre l'entité et la table qu'il représente et permet de faire du CRUD.



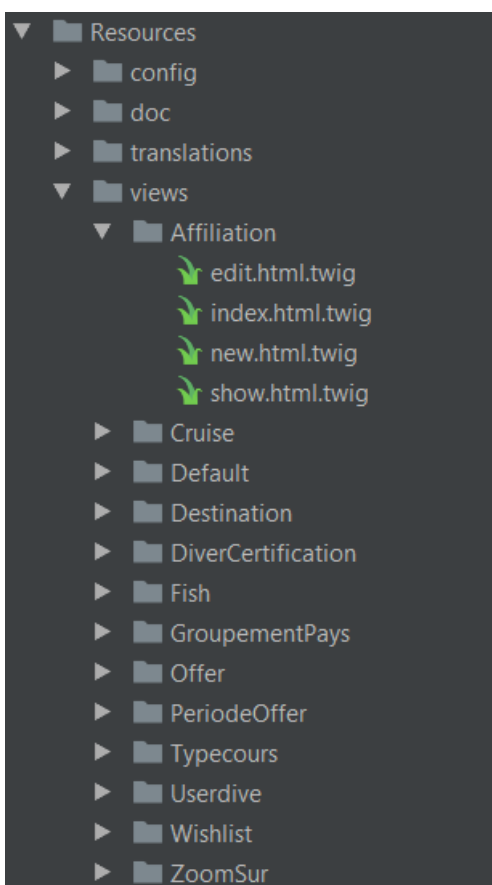
6.3.3 – Form

Il contient tous les formulaires du projet.

Symfony intègre un sous-framework appelé « Form », qui permet de générer des formulaires très complets, et cela très simplement.

Ce sous-framework permet de créer des formulaires à partir d'une entité afin d'offrir un moyen à un utilisateur de créer ou de modifier des enregistrements en base de données, il permet également de générer des formulaires dont les données saisies ne seront pas persistées en base.

Le nom du fichier d'un formulaire commence toujours par le nom de la classe, et se termine par le suffixe « Type ».



6.3.4 – Resources

Il est divisé en 4 sous dossiers :

- **config** : contient la configuration du bundle (déclaration des Routes et des Services)
- **doc** : contient la documentation du bundle
- **translations** : contient tous les fichiers de traductions
- **views** : contient tous les templates du bundle

Symfony intègre un moteur de template, appelé Twig, créé par l'éditeur de Symfony, Sensio Labs. Il peut être utilisé indépendamment de Symfony.

C'est dans ce dossier **views** que se trouvent les pages gérées avec Twig, dans une structure HTML classique. Nous pouvons y trouver, entre autres, après la génération d'un CRUD, 4 fichiers :

- **edit** qui contient des ordres **UPDATE** et **DELETE**
- **index** qui affiche un lien vers le formulaire
- **new** contient le formulaire et les ordres **INSERT**
- **show** qui contient les ordres **SELECT**

6.4 – Structure d'un Controller Symfony

Un Controller, c'est une classe PHP, avec une ou plusieurs méthodes, avec en entrée la requête, et en sortie la réponse.

Voici un extrait d'AffiliationController.php :

```
<?php

namespace Cat\DiveBundle\Controller;

use Symfony\Component\HttpFoundation\Request;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Cat\DiveBundle\Entity\Affiliation;
use Cat\DiveBundle\Form\AffiliationType;

/**
 * Affiliation controller.
 *
 */
class AffiliationController extends Controller
{
    /**
     * Lists all Affiliation entities.
     *
     */
    public function indexAction()
    {
        // Verification du ROLE de l'utilisateur
        if (!$this->get('security.context')->isGranted('ROLE_ADMIN')) {
            throw new \Exception('Vous ne disposez pas des autorisations
                                nécessaires pour effectuer cette action');
        }
        // Récupère l'EntityManager de Doctrine
        $em = $this->getDoctrine()->getManager();

        // Récupération tous les enregistrements de l'Entity Affiliation
        $entities = $em->getRepository('CatDiveBundle:Affiliation')
            ->findAll();

        // Retourne la Vue au template correspondant
        return $this->render('CatDiveBundle:Affiliation:index.html.twig',
            array(
                'entities' => $entities,
            ));
    }
}
```



```
/**
 * Creates a new Affiliation entity.
 */
public function createAction(Request $request)
{
    // Verification du ROLE de l'utilisateur
    if (!$this->get('security.context')->isGranted('ROLE_ADMIN')) {
        throw new \Exception('Vous ne disposez pas des autorisations
                               nécessaires pour effectuer cette action');
    }
    // Création d'une Entity Affiliation
    $entity = new Affiliation();
    // Récupération du Formulaire
    $form = $this->createCreateForm($entity);
    // Hydrate l'Entité avec les données postées
    $form->handleRequest($request);
    // Contrôle l'intégrité des données de l'Entité
    if ($form->isValid()) {
        $em = $this->getDoctrine()->getManager();
        $service_url = $this->container->get('catRoot.url');
        $upld = $entity->upload($service_url->getUniqFilename
                               ($entity->file));

        if($upld === -1){
            return $this->redirect(
                $this->generateUrl('superadmin_affiliation')
            );
        }
        // Si tout est correct, on persiste les données auprès de
        Doctrine
        $em->persist($entity);
        // Puis on les enregistre en BD
        $em->flush();
        // On affiche un message pour confirmer l'action
        $this->get('session')
            ->getFlashBag()
            ->add('success', $this->get('translator')
                ->trans('L\'affiliation a bien été créée'));
        // Puis on redirige vers la page show
        return $this->redirect($this->generateUrl(
            'superadmin_affiliation_show',
            array('id' => $entity->getId())));
    }

    // Si un problème est détecté, alors on affiche un message d'erreur
    $this->get('session')
        ->getFlashBag()
        ->add('error',
    $this->get('translator')
        ->trans('Un problème est survenu lors de la
                création de l\'affiliation'));

    // Création de la Vue
    return $this->render('CatDiveBundle:Affiliation:new.html.twig',
        array(
            'entity' => $entity,
            'form' => $form->createView(),
        ));
}
```


6.5 – Structure d'une Entité Symfony

L'entité n'est rien d'autre qu'une classe PHP, contenant des propriétés ainsi que les accesseurs/mutateurs.

Voici un extrait de l'entité Affiliation.php :

6.5.1 – Les Propriétés

```
<?php

namespace Cat\SiteBundle\Entity;

use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Validator\Constraints as Assert;

/**
 * Affiliation
 *
 * @ORM\Table()
 * @ORM\Entity(repositoryClass="Cat\DiveBundle\Entity\AffiliationRepository")
 */
class Affiliation
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @var integer $affiliation
     *
     * @ORM\Column(name="affiliation", type="integer")
     */
    private $affiliation;
```

Cette partie concerne les propriétés de l'entité.

Elles sont définies lors de la génération du bundle avec Doctrine2, il suffit d'entrer les propriétés définies avec le diagramme de classes pour que Doctrine2 génère le code.

Chaque propriété est précédée d'une annotation, qui est une configuration ajoutée en commentaire devant les classes, méthodes ou propriétés d'une entité.



6.5.2 – Les Accesseurs et les Mutateurs

```
/**
 * Get id
 *
 * @return integer
 */
public function getId()
{
    return $this->id;
}

/**
 * Set affiliation
 *
 * @param integer $affiliation
 * @return Affiliation
 */
public function setAffiliation($affiliation)
{
    $this->affiliation = $affiliation;

    return $this;
}

/**
 * Get affiliation
 *
 * @return integer
 */
public function getAffiliation()
{
    return $this->affiliation;
}
}
```

Cette partie concerne les **accesseurs** et les **mutateurs** de l'entité.

Plus communément appelé « **getter** » et « **setter** », ils servent à respecter le principe d'encapsulation en Programmation Orienté Objet.

Lorsque la signature de la méthode d'un objet est en « private » ou « protected », afin de limiter son accès, elle n'est accessible que dans la classe courante.

Les getter/setter sont « publics », ce qui permet de les appeler depuis une autre classe, et ainsi interagir avec les propriétés, qui sont normalement inaccessibles.

- Un **setter**, ou **mutateur**, aura pour rôle d'affecter une valeur à une propriété d'un objet privé.
- Un **getter**, ou **accesseur**, aura pour rôle de récupérer et de retourner la valeur d'une propriété d'un objet privé.

Leurs noms doivent, par convention, être préfixés du mot « set » pour le mutateur, et du mot « get » pour l'accesseur, le mot suivant devant être « camélisé », c'est-à-dire commencer par une majuscule, pour plus de lisibilité.

Ce qui donne, dans notre exemple, « setAffiliation », et « getAffiliation ».

6.6 – Structure d'un formulaire « Form »

```
<?php

namespace Cat\DiveBundle\Form;

use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolverInterface;

/**
 * Class AffiliationType
 * @package Cat\DiveBundle\Form
 */
class AffiliationType extends AbstractType
{
    /**
     * @param FormBuilderInterface $builder
     * @param array $options
     */
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('name','text', array('label' => 'Nom de l\'Affiliation'))
            ->add('file','file', array('label' => 'Logo (jpg, png, gif)',
                                   'required' => false))
            ->add('affiliation_type','choice', array('choices' => array(
                                                         '1' => '1 - Plongée',
                                                         '2' => '2 - Apnée' )))
        ;
    }

    /**
     * @param OptionsResolverInterface $resolver
     */
    public function setDefaultOptions(OptionsResolverInterface $resolver)
    {
        $resolver->setDefaults(array(
            'data_class' => 'Cat\DiveBundle\Entity\Affiliation'
        ));
    }

    /**
     * @return string
     */
    public function getName()
    {
        return 'cat_divebundle_affiliation';
    }
}
```

Voici le contenu d'un fichier Form. C'est une classe qui étend d'une autre classe, `AbstractType`, qui est un composant de Form.

Elle est composée d'une fonction « `buildForm` », qui va nous permettre de construire le squelette de notre formulaire, en définissant les champs dont nous aurons besoin. C'est le moteur du formulaire, il se concentre sur le traitement des données, ainsi que sur sa mise en forme future dans le rendu de notre vue.



6.7 – Structure d'une page Twig

```
{% extends '::admin.html.twig' %}

{% block content %}
<div class="wrapper wrapper-content animated fadeInRight">
  <div class="row">
    <div class="col-md-12">
      <div class="ibox float-e-margins">
        <div class="ibox-title">
          <h1>{% trans %}Création d'une affiliation{% endtrans %}</h1>

          {{ form_start(form) }}
          {{ form_row(form.name, { 'attr': { 'class': 'form-control' } }) }}
          {{ form_row(form.affiliation_type, { 'attr':
                                                    { 'class': 'form-control' } }) }}

          {{ form_row(form.file) }}

          <div class="hidden">
            {{ form_rest(form) }}
          </div>
          <br />
          <ul class="record_actions">
            <li>
              <button type="submit" class="btn btn-primary">
                {% trans %}Créer affiliation{% endtrans %}
              </button>
            </li>

            <li>
              <a class="btn btn-primary"
                href="{{ path('superadmin_affiliation') }}">
                <i class="fa fa-th-list"></i>
                {% trans %}Retour à la liste{% endtrans %}
              </a>
            </li>
          </ul>
        </div>
      </div>
    </div>
  </div>
</div>
{% endblock %}
```

Twig est le moteur de template par défaut dans Symfony.

Un moteur de template permet de séparer l'interface graphique, de la logique d'une page web.

Le code ci-dessus représente le formulaire d'ajout d'une affiliation.

L'objectif principal de Twig est de réduire le nombre de caractère dit « non nécessaire ». Il offre un pseudo-langage propre, avec une syntaxe et une utilisation très simple, pour un rendu très puissant.

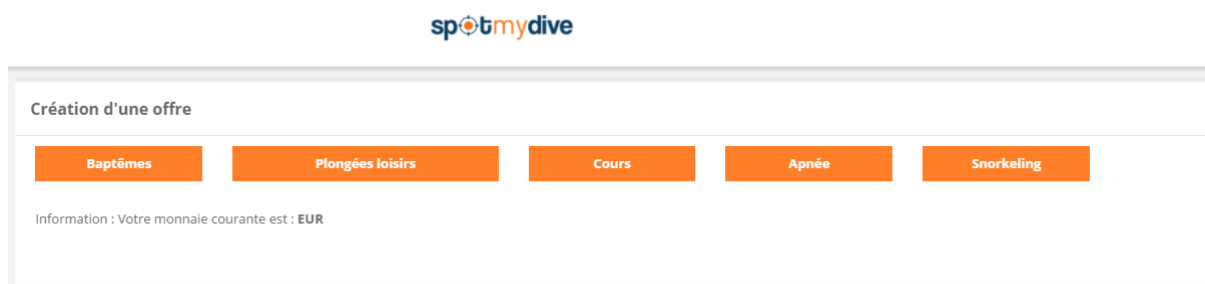
Le langage de Twig est transformé en PHP avant d'être exécuté pour finalement afficher le contenu, il est ensuite mis en cache pour une plus grande rapidité lors d'un second affichage.

6.8 – jQuery

J'ai dû mettre en place les formulaires liés aux affiliations dans une barre de navigation, ou « navbar », gérée avec jQuery.

Le principe est simple : Charger l'intégralité du formulaire, le cacher, puis en afficher seulement une partie. Ce procédé permet d'avoir une fluidité d'affichage lorsque l'utilisateur va cliquer sur l'un des boutons de la navbar.

Voici le rendu final, j'ai dû modifier Cours, et ajouter Apnée et Snorkeling.



spotmydive

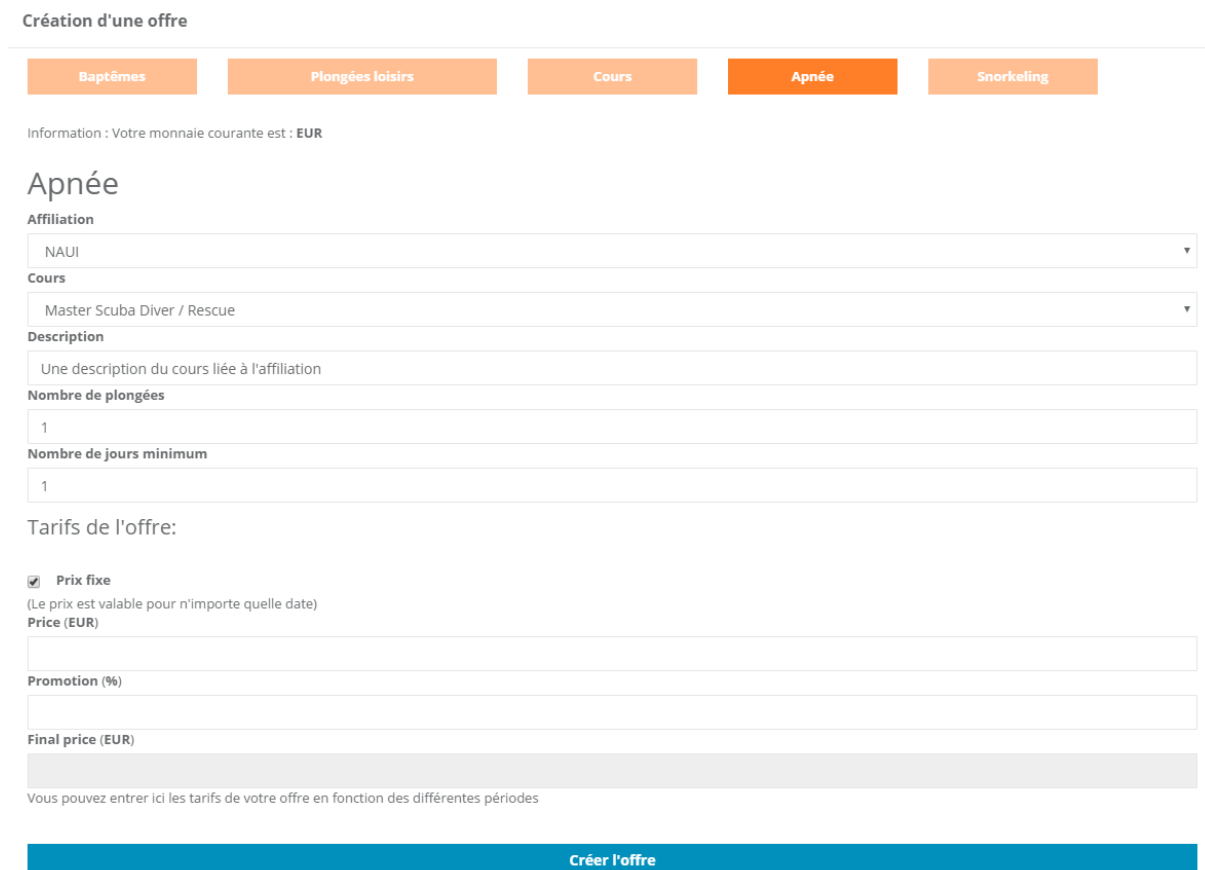
Création d'une offre

Baptêmes Plongées loisirs Cours Apnée Snorkeling

Information : Votre monnaie courante est : EUR

Lorsque l'utilisateur, dans ce cas-ci, l'administrateur/propriétaire d'un centre de plongée, clique sur l'un des boutons du menu, le formulaire correspondant est récupéré puis affiché juste en dessous.

Voici un exemple avec le formulaire Apnée :



Création d'une offre

Baptêmes Plongées loisirs Cours Apnée Snorkeling

Information : Votre monnaie courante est : EUR

Apnée

Affiliation

NAUI ▼

Cours

Master Scuba Diver / Rescue ▼

Description

Une description du cours liée à l'affiliation

Nombre de plongées

1

Nombre de jours minimum

1

Tarifs de l'offre:

☒ **Prix fixe**
(Le prix est valable pour n'importe quelle date)

Price (EUR)

Promotion (%)

Final price (EUR)

Vous pouvez entrer ici les tarifs de votre offre en fonction des différentes périodes

Créer l'offre



Quelques extraits du code jQuery de la gestion des onglets de la navbar :

```
// On cache tous les formulaires au chargement de la page
$(".hid_baptême").hide();
$(".hid_loisirs").hide();
$(".hid_cours").hide();
$(".hid_apnea").hide();
$(".hid_snorkeling").hide();
```

Ici, nous allons donc cacher les formulaires pour ne faire apparaître que les boutons de la navbar, avec `.hide()`;

C'est une fonction jQuery très puissante qui permet de changer le « statut » d'une partie de la page instantanément.

```
// Le formulaire se charge sur l'onglet Apnée
{% if check == 5 %}
    $(".hid_apnea").show();
    $(".apnea").addClass("btn-primary2");
{% endif %}
```

Si le « check » correspond à la 5^{ème} valeur, alors on charge et on affiche le formulaire.

```
// Affichage de l'onglet Apnée
$(".apnea").bind("click", function () {
    $(".loisirs").removeClass("btn-primary2");
    $(".cours").removeClass("btn-primary2");
    $(".baptême").removeClass("btn-primary2");
    $(".apnea").addClass("btn-primary2");
    $(".snorkeling").removeClass("btn-primary2");
    $(".hid_apnea").show(); // C'est cette ligne qui affiche l'onglet
    $(".hid_cours").hide();
    $(".hid_baptême").hide();
    $(".hid_loisirs").hide();
    $(".hid_snorkeling").hide();
});
```

Ensuite, les informations sont traitées dans le formulaire Twig.

6.9 – Autres Langages

6.9.1 – Bootstrap

Bootstrap est un framework **CSS** (Cascading Style Sheets), créé par et pour **Twitter** en 2011. Remarquant le potentiel qu'avait eu cette solution en interne, les ingénieurs en charge du projet, ont décidé de le publier comme projet open-source.

Il est devenu, en quelques mois seulement, l'outil incontournable pour tout développeur front-end, il est depuis le framework CSS le plus populaire.

Mais Bootstrap, qu'est-ce que c'est ?

C'est tout simplement une grande bibliothèque de plusieurs fonctions, qui permettent de mettre en forme le design d'un site web, intégrant le « responsive design », c'est-à-dire la capacité du site à s'afficher correctement sur toutes les plateformes existantes, ordinateurs, tablettes, smartphones...

C'est un ensemble standardisé qui contient des codes HTML, CSS, et pleins d'éléments interactifs comme des boutons, des formulaires, des barres de progression, et même des extensions JavaScript via jQuery.

Tout cela permet d'avoir une apparence visuelle uniforme pour tous les éléments d'une page web.

6.9.2 – HTML / CSS

Les bases de la construction et de la mise en forme d'une page web.

HTML, « HyperText Markup Language », ou Langage Signalétique HyperTexte, est le format par défaut conçu pour représenter une page web. C'est un langage de balisage, qui permet d'écrire des documents « hypertexte », c'est-à-dire permettant la mise en relation de plusieurs documents grâce à des « hyperliens », ou liens hypertexte.

C'est lui qui va permettre la construction d'une page web.

CSS, « Cascading Style Sheets », ou Feuille de Style de Cascade, est un langage qui va s'occuper de la présentation d'un document HTML, ou même XML.

C'est lui qui va ordonner la mise en forme d'un cadre, la taille d'une image, ou encore la couleur d'un lien hypertexte.

Toutes les instructions de mise en forme se trouvent dans un même fichier, facilement modifiable, qui répercutera les changements sur toutes les pages concernées.

Chapitre 7 – Déploiement de l'Application

7.1 – Déploiement

Le diagramme de déploiement est réalisé en fin de parcours.

Il correspond à la description de l'environnement d'exécution du système et de la façon dont les composants y sont installés.

Un DPL représente l'architecture physique des matériels (**nœuds**) qui composent un système ainsi que la répartition des **composants** et des **artéfacts** (fichiers exécutables et data) sur les matériels et les liens entre les différents nœuds (le plus souvent des liaisons réseaux).

Les nœuds sont représentés par des cubes.

Chaque nœud ou processeur assure un ensemble de traitements représentés par des composants.

La nature de l'équipement peut être précisée au moyen d'un **stéréotype**.

Les liens sont représentés par des arcs.

Ils peuvent être surmontés d'un stéréotype pour préciser le protocole utilisé.

Des **multiplicités** peuvent préciser les associations entre nœuds.

Les nœuds qui correspondent à des processeurs (pas des dispositifs) portent le nom des exécutables qu'ils hébergent.

Chez SpotMyDive, le déploiement se fait en 2 étapes.

Tout d'abord, le code est récupéré avec **GIT** par l'**administrateur du système**, puis il est mis sur un serveur de **préproduction**, ce qui permet de tester les nouvelles fonctionnalités en conditions réelles sans affecter le serveur de production, et ainsi détecter les éventuels bugs qui n'auraient pas été vus en local, pour les corriger.

Une fois que tout est vérifié sur la **préprod**, l'administrateur envoie le tout sur le **serveur de production**.

Voici le rendu final de cette mise en place, avec quelques screenshots des étapes de la création d'une Affiliation par un Administrateur du site, l'ajout d'une offre liée à une Affiliation par un Centre de Plongée, ainsi que le rendu final des offres sur la page Réservations.

Listing des Affiliations : Vue de l'Administrateur du site web

FR

SpotMyDive

Lukas

Tableau de bord

Commandes (0)

Offres

Affiliations

Types de cours

Sites

Pays

Utilisateurs (0)

Commentaires (0)

Poissons

Photos (0)

Zoom Sur

10


records per page

Nom

Logo Affiliation

Actions

AIDA




Voir

Modifier

Supprimer

CMAS




Voir

Modifier

Supprimer

FRESSM




Voir

Modifier

Supprimer

FIAS Italy



Voir

Modifier

Supprimer

FR

SpotMyDive

Lukas

Tableau de bord

Commandes (0)

Offres

Affiliations

Types de cours

Sites

Pays

Utilisateurs (0)

Commentaires (0)


Poissons

Photos (0)

Zoom Sur

PADI

PADI TecRec




Voir

Modifier

Supprimer

SDI




Voir

Modifier

Supprimer

SSI




Voir

Modifier

Supprimer

TDI



Voir

Modifier

Supprimer

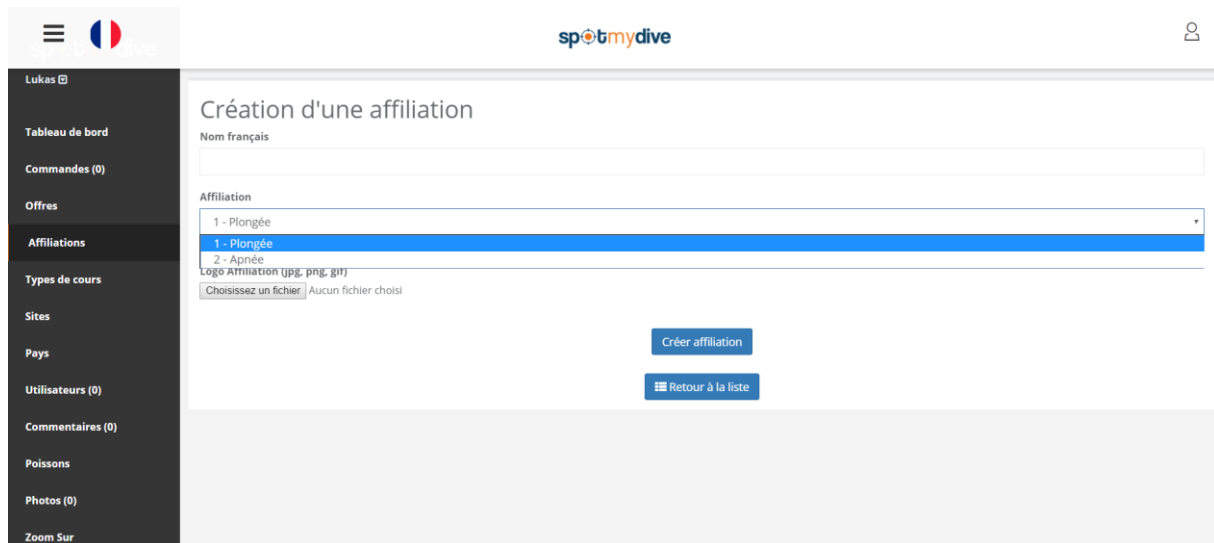
Previous

1

Next

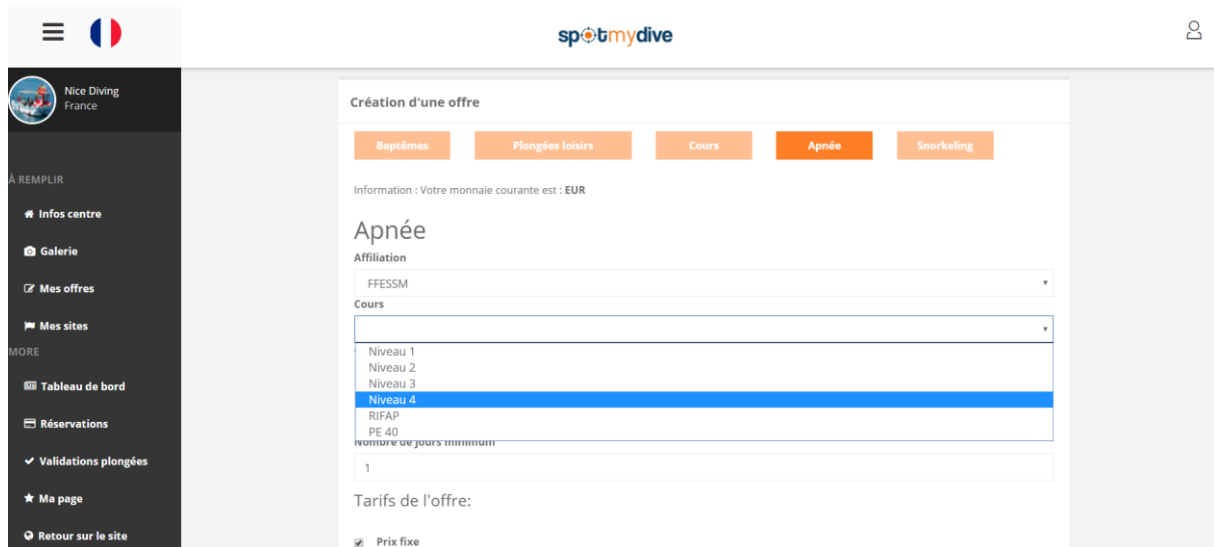
Ajouter une affiliation

Création d'une Affiliation : Vue de l'Administrateur du site web



The screenshot shows the 'Création d'une affiliation' page. On the left is a dark sidebar with a menu: 'Lukas', 'Tableau de bord', 'Commandes (0)', 'Offres', 'Affiliations' (highlighted), 'Types de cours', 'Sites', 'Pays', 'Utilisateurs (0)', 'Commentaires (0)', 'Poissons', 'Photos (0)', and 'Zoom Sur'. The main content area has the title 'Création d'une affiliation'. It contains a 'Nom français' text field, an 'Affiliation' dropdown menu with options '1 - Plongée' and '2 - Apnée' (the second option is selected), a 'Logo Affiliation (jpg, png, gif)' section with a 'Choisissez un fichier' button and the text 'Aucun fichier choisi', a 'Créer affiliation' button, and a 'Retour à la liste' button.

Création d'une Offre liée à une Affiliation : Vue d'un Centre de Plongée - Apnée



The screenshot shows the 'Création d'une offre' page for a user named 'Nice Diving France'. The left sidebar has a menu: 'Nice Diving France' (with a profile picture), 'À REMPLIR', 'Infos centre', 'Galerie', 'Mes offres', 'Mes sites', 'MORE', 'Tableau de bord', 'Réservations', 'Validations plongées', 'Ma page', and 'Retour sur le site'. The main content area has the title 'Création d'une offre' and five tabs: 'Baptêmes', 'Plongées loisirs', 'Cours', 'Apnée' (selected), and 'Snorkeling'. Below the tabs, it says 'Information : Votre monnaie courante est : EUR'. The form is titled 'Apnée' and includes an 'Affiliation' dropdown set to 'FFESSM', a 'Cours' dropdown set to 'Niveau 4', a 'RIFAP' field set to 'PE 40', and a 'nombre de jours minimum' field set to '1'. At the bottom, under 'Tarifs de l'offre:', there is a checked checkbox for 'Prix fixe'.



Création d'une Offre liée à une Affiliation : Vue d'un Centre de Plongée - Cours

Nice Diving
France

À REMPLIR

Infos centre

Galerie

Mes offres

Mes sites

MORE

Tableau de bord

Réservations

Validations plongées

Ma page

Retour sur le site

spotmydive

Création d'une offre

Baptêmes

Plongées loisirs

Cours

Apnée

Snorkeling

Information : Votre monnaie courante est : EUR

Cours

Affiliation

SSI

Cours

Open Water Diver

Advanced Adventurer

Stress & Rescue

OW Instructor

Dive Guide

Divemaster

AOW Instructor

1

Tarifs de l'offre:

Prix fixe

Visualisation du Front : Vue de l'utilisateur du site web

f

Reservation

t

Informations

i

La carte des sites

o

Meilleurs spots

p

Meilleurs centres

Ecrivez un avis

RESERVATIONS

BAPTEMES

LOISIRS

COURS

APNEE

SNORKELING

Cours de plongée PADI

EFR Détails de l'offre ^

0 plongées - 1 jour

Équipement inclus

200 €

+ AJOUTER

Description

Ce qui est inclus ?

Indispensable pour passer son niveau Rescue diver, la formation Emergency First réponse est l'équivalent du passage d'un brevet de secourisme adapté à la pratique de la plongée sous-marine. L'EFR vous forme à la manière d'administrer les premiers secours ou encore de réaliser une réanimation cardio-pulmonaire. Requis • Aucun âge minimum n'est requis, • Formation ouverte aux non-plongeurs

Open Water Diver Détails de l'offre v

4 plongées - 3 jours

Équipement inclus

480 €

+ AJOUTER

Advanced OW Diver Détails de l'offre v

5 plongées - 3 jours

Équipement inclus

365 €

+ AJOUTER

Offline

iver Détails de l'offre v

s - 3 jours

405 €

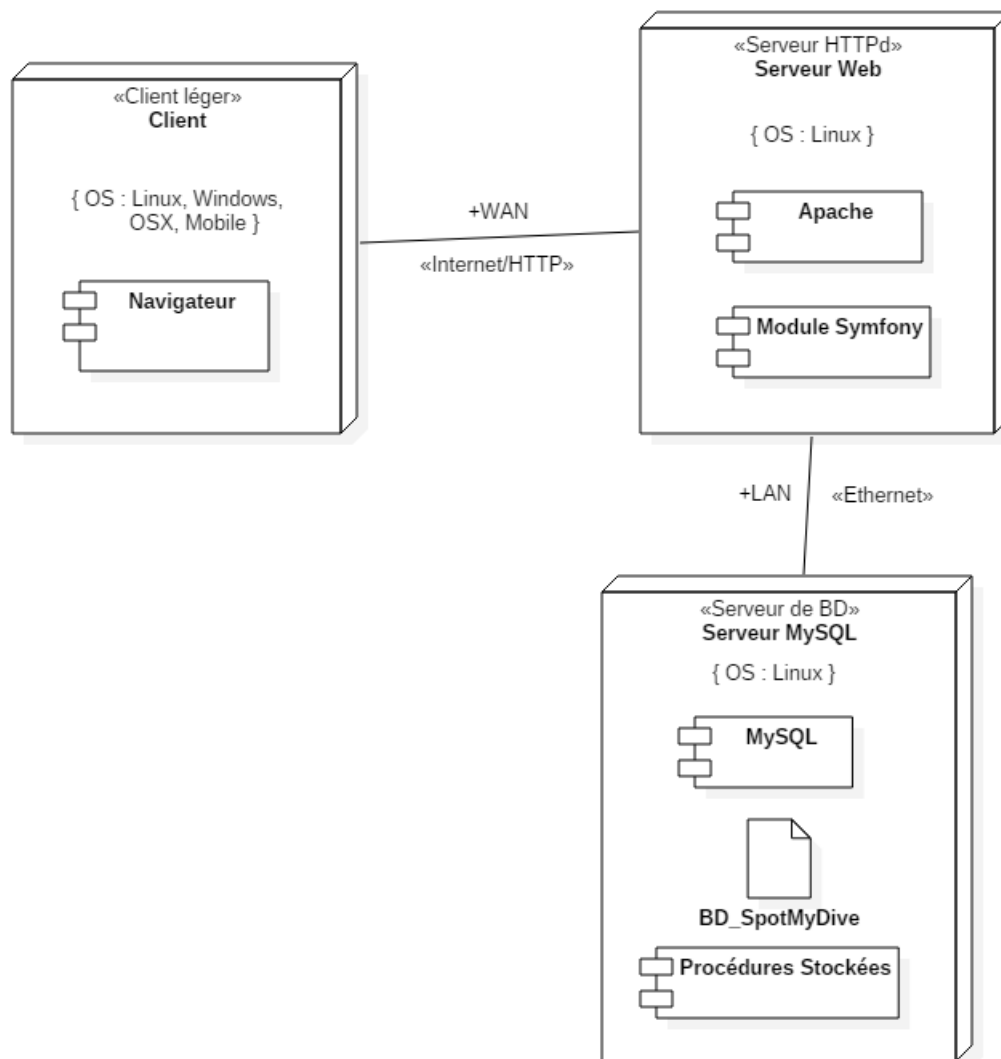
+ AJOUTER

Présenté par Benoît CATILLON

Page - 60 -

7.2 – Diagramme de Déploiement

Diagramme de Déploiement
Affiliation



Chapitre 8 – Gestion de Projet

8.1 – GIT



GIT est un logiciel créé par **Linus Torvalds**, également créateur du noyau Linux. C'est d'ailleurs pour le développement de ce dernier que GIT a été programmé.

Il s'agit d'un logiciel de gestion de versions multi-users et décentralisé, pour la création d'applications en équipe, qui permet de garder un historique de toutes les modifications faites sur le code, pour faire des comparaisons ou revenir à un code antérieur, par exemple.

Chaque fois que des modifications sont faites sur le projet, nous pouvons les enregistrer via un « commit », accompagné d'un message pour décrire la modification.

Un commit correspond à une version du code, avec un identifiant unique.

Ce logiciel s'utilise principalement dans une console, en ligne de commande, très simplement grâce à quelques instructions très simples, dont voici quelques exemples :

GIT INIT	Initialise GIT sur le projet et crée un nouveau dépôt
GIT CLONE	Permet de créer une copie du dépôt local ou de récupérer un projet sur un serveur distant
GIT ADD	Ajoute à l'index les fichiers versionnés et modifiés
GIT COMMIT	Stock l'index avec un message qui décrit les changements
GIT PUSH	Envoi le commit sur le dépôt du serveur distant
GIT PULL	Permet de récupérer la dernière version du serveur distant
GIT LOG	Historique complet de toute l'activité sur le dépôt

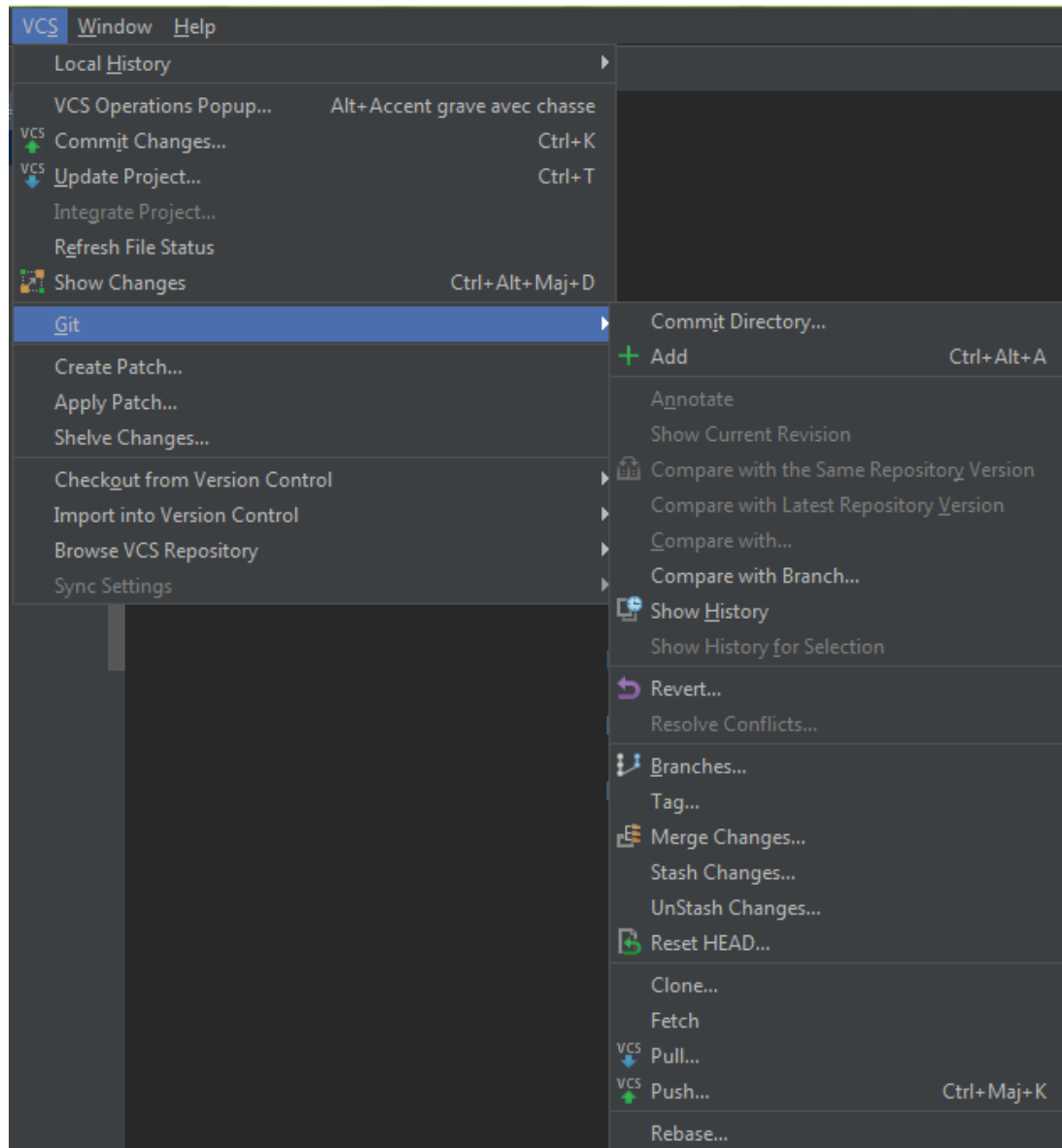
SpotMyDive utilise GIT sur un serveur interne à l'entreprise qui gère le projet.



8.2 – Gestion de GIT avec PHPStorm

PHPStorm permet la gestion de GIT, directement implémentée dans l'interface et configurée avec le serveur distant, sans passer par la console et ses lignes de commande.

Voici l'interface :



Nous pouvons très facilement versionner le code, l'enregistrer, le partager...

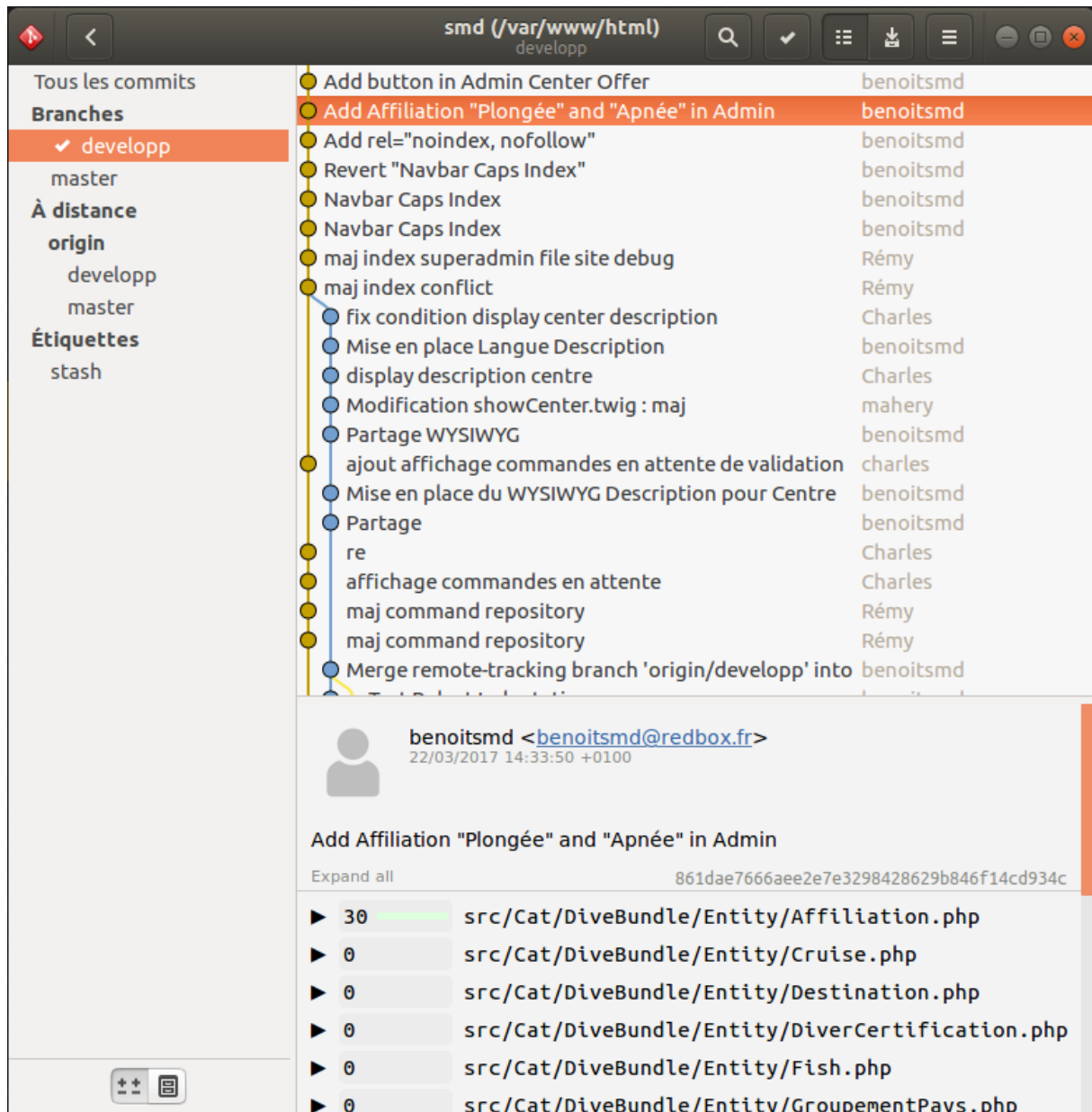
Toute la gestion est prévue par PHPStorm, il n'est plus nécessaire de passer par la console et ses lignes de commandes.



8.3 – Gestion de GIT avec GitG

Le développeur s'occupant du projet m'a conseillé d'installer GitG, c'est un logiciel qui permet de consulter très facilement un répertoire, ou repository, GIT.

PHPStorm offre quelque chose de similaire, mais GitG peut-être plus clair dans certains cas, ils sont donc complémentaires.



Chapitre 9 – Sécurité

9.1 – La Sécurité avec Symfony

Symfony propose de nombreuses possibilités de sécurisation de l'application, et ce, nativement.

Les échanges entre l'utilisateur et le serveur sont source de risques qu'il est important de connaître et de maîtriser. Il faut être prudent avec le contenu et l'origine des données reçues ou envoyées par l'utilisateur ou le serveur, afin d'éviter les attaques de type injection SQL, exploitation de failles XSS...

Une règle d'or en programmation : Ne **JAMAIS** faire confiance à l'utilisateur !

Des sécurités doivent donc être mises en place à différents niveaux de l'application.

Un fichier nommé « security.yml » est généré par défaut lors de la création d'une application Symfony, il décrit les règles d'authentification et d'autorisation pour toute l'application. Il se trouve dans le répertoire « app/config/ », avec tous les fichiers de configuration.

9.2 – Failles XSS, CSRF, et injections SQL

Voici les principales failles de sécurité d'une application web :

Les failles XSS, ou Cross Site Scripting, ont pour but d'injecter du code interprétable par le navigateur, comme du HTML ou du JavaScript, côté client.

Il existe de nombreuses possibilités d'utilisation de cette faille : vol de cookies, modification de la structure HTML, redirection vers un autre site...

Les failles CSRF, ou Cross Site Request Forgeries, exploitent les utilisateurs en les rendant complices de l'attaque. C'est une faille spécifique aux applications web, car elles exploitent principalement le navigateur de l'utilisateur.

Le principe est d'inciter un internaute à se rendre sur une page qui va exécuter une requête à son insu. Si cet utilisateur a des droits spécifiques sur l'application, les répercussions peuvent être catastrophiques.

Les injections SQL, également appelées SQLi, consistent à modifier la requête SQL en cours, en y « injectant » du code SQL qui n'était pas prévu dans la requête initiale. Il existe plusieurs méthodes qui permettent ces injections, c'est pourquoi il faut être très vigilant.

Doctrine2 permet de se protéger de ces failles de sécurité, nativement, et très simplement, même s'il ne fait pas tout le travail, et qu'il ne protège que des requêtes « classiques ».

Chapitre 10 – Conclusion

10.1 – La Formation

J'ai toujours été passionné par le monde de l'informatique, et particulièrement le développement, que j'ai appris en autodidacte il y a plusieurs années.

Ce choix de formation n'est donc pas anodin, je sais que mon épanouissement dans la vie professionnelle ne pourra se faire que par ce biais.

Le côté intensif de la formation a révélé en moi un passionné qui ne compte pas les heures données à la compréhension et l'apprentissage général.

Les professeurs ont tout fait pour nous faciliter les choses au maximum, mais nous avons une très grande part de travail à réaliser en dehors des heures allouées à la formation.

Une formation extrêmement riche en connaissances, qui m'a permis d'en apprendre bien plus que je ne l'aurais espéré, dans un temps pourtant très court.

10.2 – Le Stage

Ce stage chez SpotMyDive me conforte définitivement dans ce choix.

Le contexte professionnel n'était pourtant pas favorable à un aboutissement du projet dans les temps, mais les recherches ainsi que les échanges avec mon binôme m'ont permis d'arriver à mes fins, même si cela reste perfectible.

L'entreprise m'a donné carte blanche pour adapter l'application comme je la voyais, ce qui m'a permis de faire un énorme travail de réflexion pour répondre au mieux à leurs attentes.

Le retour de mes responsables est excellent, ils n'espéraient pas voir le projet fini dans les temps, sachant que cette application n'était pas la seule mission qu'ils m'ont confié.

Cette première expérience dans ce monde est, je pense, un grand succès, pour moi comme pour SpotMyDive.

10.3 – Projet Futur

SpotMyDive me propose de continuer l'aventure à leur côté.

Une opportunité inespérée que je ne pensais pas accessible immédiatement.

La richesse de leur projet me donne envie de continuer sur cette lancée, la mise en place d'une offre « Croisière » en tant que nouveau modèle économique est un énorme travail qui me permettrait de continuer mon apprentissage de Symfony avec des bases solides.

Chapitre 11 – Annexes



11.1 – Correspondance Projet / REAC

ACTIVITÉ	COMPÉTENCE	CORRESPONDANCE
Développer des composants d'interface	Maquetter une application	UML : DCU, Maquettes Mockup, DNAV, DAC, DSEQ, DSD, DPL
	Développer des composants d'accès aux données	Serveur PHP
	Développer des pages web en lien avec une base de données	HTML5, CSS3, Twig, jQuery, PHP
Développer la persistance des données	Concevoir une base de données	Règles de Gestion UML (DCL, MPD) avec StarUML Schéma de la Base de Données avec phpMyAdmin
	Mettre en place une base de données	Création avec Doctrine2
	Développer des composants dans le langage d'une base de données	Gestion de la persistance avec une Entité (Doctrine2)
	Utiliser l'anglais dans l'activité professionnelle en informatique	Développement
Développer une application n-tiers	Concevoir une application	Démarche UML
	Collaborer à la gestion d'un projet informatique	Gestion de projet avec la méthode AGILE Trello
	Développer des composants métier	Classes « Entités »
	Construire une application organiser en couches	MVC
	Préparer et exécuter le déploiement d'une application	Diagramme de Déploiement (UML) Déploiement Mise en place de la BD Mise en place des fichiers Twig, CSS et jQuery Mise en place des fichiers PHP

11.2 – Les outils utilisés

OUTIL	OBJECTIF	COMMENTAIRE
PHPStorm	IDE : Rédaction du code Gestion de GIT	Un outil parfait pour gagner du temps sur bien des points,
SGBDR	MySQL	Rien à ajouter, vraiment très puissant.
phpMyAdmin	Gestion de la BD	Un indispensable que je connais depuis quelques années, et que j'ai pu voir évoluer. Je ne peux pas m'en passer !
GitG	Gestion de GIT	Un très bon complément à ce que propose PHPStorm.
Trello	Gestion de projet	Une très bonne application en ligne qui permet de créer un tableau Scrum, utilisé par l'entreprise.
StarUML	Modélisation UML	OS : Linux
PowerDesigner	Modélisation UML	OS : Windows
Balsamiq Mockup	Outil de maquettage	OS : Windows



11.3 – Bibliographie et Webographie

11.3.1 – Bibliographie

- UML2, Modéliser une application Web, Pascal ROQUES Ed. EYROLLES
- UML2 par la pratique, Pascal ROQUES, Ed. EYROLLES
- UML 2 en action, Pascal ROQUES, Ed. EYROLLES
- UML 2, Pratique de la modélisation, Benoit CHARROUX...Ed.
- Support de cours UML, Pascal BUGUET

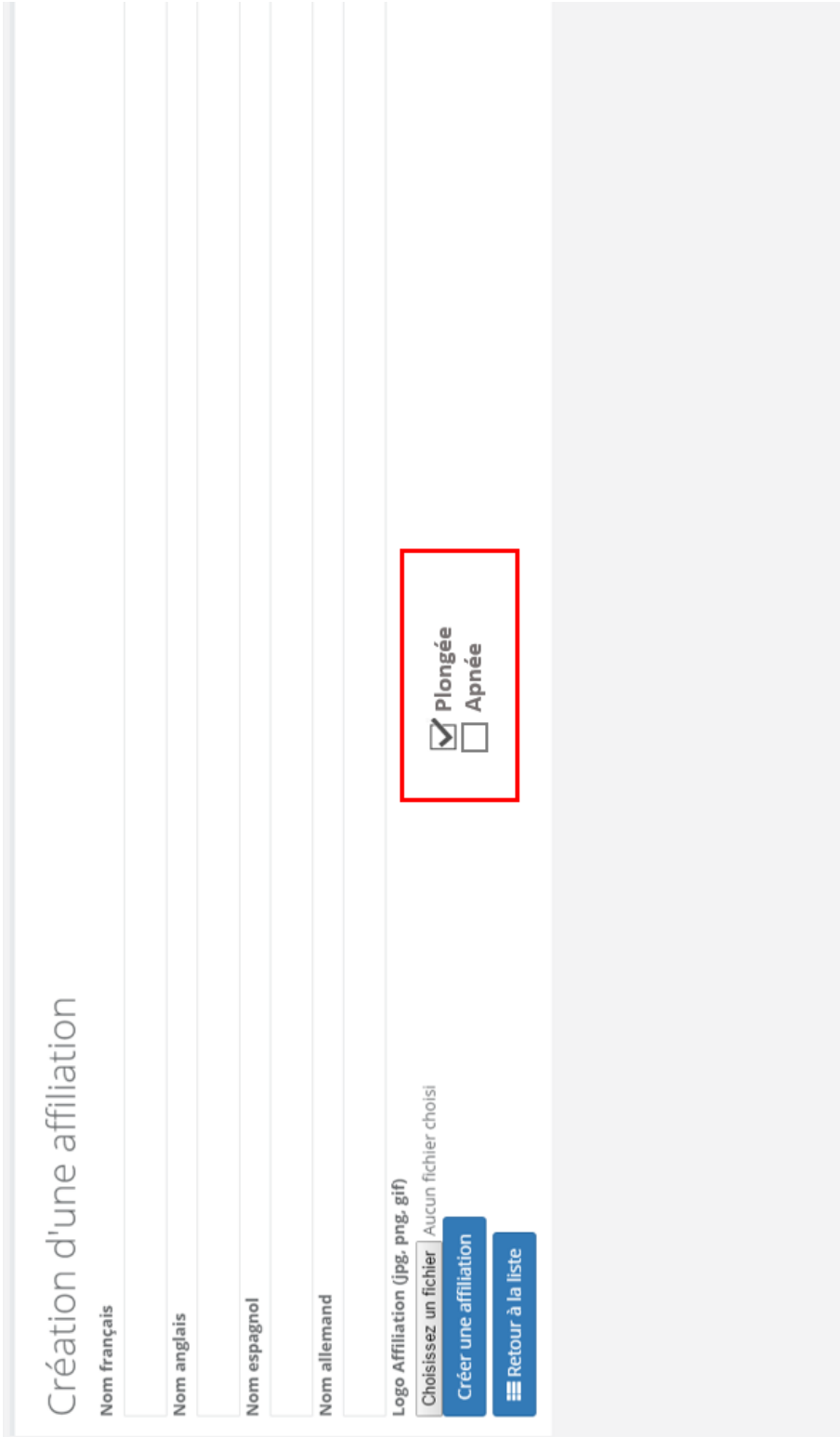
- Support de cours PHP, SQL/MySQL, Symfony, Javascript, des différents intervenants M2i

11.3.2 – Webographie

- <http://php.net>
- <http://openclassrooms.com>
- <http://symfony.com>
- <http://w3schools.com>
- <http://elephorm.com>

11.4 – Cahier des Charges

Document fournis par l'entreprise, montage basé sur des formulaires existants.
Quelques erreurs ont été corrigées par la suite.



Création d'une affiliation

Nom français

Nom anglais

Nom espagnol

Nom allemand

Logo Affiliation (jpg, png, gif)

Choisissez un fichier Aucun fichier choisi

☒ Plongée ☐ Apnée

Créer une affiliation

Retour à la liste

Création d'une offre

Baptême de plongée

Déjà certifiés Plongée loisirs

Cours de plongée

Apnée

Snorkeling

Votre monnaie couranteEUR

Baptême

Nom de l'offre

Description

Nombre de plongées

1

Nombre de jours minimum

1

Tarifs:

☒ Prix fixe
(Le prix est valable pour n'importe quelle date)
Prix (EUR)

Promotion (%)

Final price (EUR)

Vous pouvez entrer ici les tarifs de votre offre en fonction des différentes périodes

on.

14:44 14/03/2017

Création d'une offre

Baptême de plongée

Déjà certifiés Plongée loisirs

Cours de plongée

Apnée

Snorkeling

Votre monnaie couranteEUR

Plongée loisir

Nom de l'offre

Description

Nombre de plongées

1

Nombre de jours minimum

1

Tarifs:

☒ Prix fixe
(Le prix est valable pour n'importe quelle date)

Prix (EUR)

Promotion (%)

Final price (EUR)

Vous pouvez entrer ici les tarifs de votre offre en fonction des différentes périodes

Création d'une offre

Baptême de plongée

Déjà certifiés Plongée loisirs

Cours de plongée

Apnée

Snorkeling

Cours

Affiliation

Description

Nombre de plongées

1

Nombre de jours minimum

1

Tarifs:

☒ Prix fixe

(Le prix est valable pour n'importe quelle date)

Prix (EUR)

Promotion (%)

Final price (EUR)

Vous pouvez entrer ici les tarifs de votre offre en fonction des différentes périodes

on.

14:44

14/03/2017

Création d'une offre

Baptême de plongée

Déjà certifiés Plongée loisirs

Cours de plongée

Apnée

Snorkeling

Apnée

Affiliation

Nom de l'offre

Description

Nombre de plongées

1

Nombre de jours minimum

1

Tarifs:

☒ Prix fixe
(Le prix est valable pour n'importe quelle date)

Prix (EUR)

Promotion (%)

Final price (EUR)

Création d'une offre

Baptême de plongée

Déjà certifiés Plongée loisirs

Cours de plongée

Apnée

Snorkeling

Votre monnaie couranteEUR

Snorkeling

Nom de l'offre

Description

Nombre de plongées

1

Nombre de jours minimum

1

Tarifs:

☒ Prix fixe
(Le prix est valable pour n'importe quelle date)

Prix (EUR)

Promotion (%)

Final price (EUR)

on

↓

📄

📶

🔋

14:44

14/03/2017

RESERVATION

Reservation

Informations

Les sites

Meilleurs centres

Meilleurs spots

Ecrivez un avis



Baptême de plongée

**Déjà certifiés
Plongée loisirs**

Apnée

Snorkeling

Cours de plongée PADI

Open Water Diver
3 plongées - 3 jours
Équipement inclus

\$258

+ AJOUTER

[View details](#)

RESERVATION

- Reservation**
- Informations
- Les sites
- Meilleurs centres
- Meilleurs spots
- Ecrivez un avis



- Baptême de plongée
- Déjà certifiés Plongée loisirs
- Cours de plongée
- Apnée
- Snorkeling

Cours de plongée PADI

Open Water Diver
3 plongées - 3 jours
Équipement inclus

258 \$

Hide details ▲

+ AJOUTER

Ce qui est inclus ?

Description

Tout premier niveau PADI, la formation Open Water Diver vous permet de découvrir l'ensemble des techniques et la théorie nécessaire pour plonger et découvrir le monde sous-marin en toute sécurité. Il permet une autonomie en binôme jusqu'à 18m. En France, la formation PADI Open Water Diver équivaut au niveau 1 de plongée FFESSM/CMAS et permet de plonger dans la zone des 20m accompagné par un guide de palanquée.

Requis

Au moins 10 ans

Selon les pays, un certificat médical de non contre-indication à la plongée sous-marine ou une décharge de responsabilité



11.5 – Code complet de création de la BD

```
CREATE DATABASE IF NOT EXISTS `spotmydive` /*!40100 DEFAULT CHARACTER SET
utf8 COLLATE utf8_unicode_ci */;
USE `spotmydive`;
-- MySQL dump 10.13  Distrib 5.7.17, for Win64 (x86 64)
--
-- Host: localhost    Database: spotmydive
--
-- Server version  5.7.14

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `affiliation`
--

DROP TABLE IF EXISTS `affiliation`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `affiliation` (
  `id_affiliation` int(11) NOT NULL AUTO_INCREMENT,
  `offer_id` int(11) NOT NULL,
  `name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `affiliation_type` int(11) NOT NULL,
  `description` longtext COLLATE utf8_unicode_ci,
  `path` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`id_affiliation`),
  KEY `offer_id` (`offer_id`),
  CONSTRAINT `affiliation_ibfk_1`
  FOREIGN KEY (`offer_id`)
  REFERENCES `offer` (`id_offer`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `affiliation`
--

LOCK TABLES `affiliation` WRITE;
/*!40000 ALTER TABLE `affiliation` DISABLE KEYS */;
/*!40000 ALTER TABLE `affiliation` ENABLE KEYS */;
UNLOCK TABLES;
```



```
--
-- Table structure for table `leisure_type`
--

DROP TABLE IF EXISTS `leisure_type`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `leisure_type` (
  `id_leisure_type` int(11) NOT NULL AUTO_INCREMENT,
  `affiliation_id` int(11) NOT NULL,
  `name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `description` longtext COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`id_leisure_type`),
  KEY `affiliation_id` (`affiliation_id`),
  CONSTRAINT `leisure_type_ibfk_1`
  FOREIGN KEY (`affiliation_id`)
  REFERENCES `affiliation` (`id_affiliation`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `leisure_type`
--

LOCK TABLES `leisure_type` WRITE;
/*!40000 ALTER TABLE `leisure_type` DISABLE KEYS */;
/*!40000 ALTER TABLE `leisure_type` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `offer`
--

DROP TABLE IF EXISTS `offer`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `offer` (
  `id_offer` int(11) NOT NULL AUTO_INCREMENT,
  `affiliation_id` int(11) NOT NULL,
  `name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `description` longtext COLLATE utf8_unicode_ci NOT NULL,
  `leisure` tinyint(1) NOT NULL,
  `lesson` tinyint(1) NOT NULL,
  `firstdive` tinyint(1) NOT NULL,
  `apnea` tinyint(1) NOT NULL,
  `snorkeling` tinyint(1) NOT NULL,
  `price` double NOT NULL,
  `date_begin` datetime NOT NULL,
  `date_end` datetime NOT NULL,
  PRIMARY KEY (`id_offer`),
  UNIQUE KEY `affiliation_id` (`affiliation_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
```



```
--
-- Dumping data for table `offer`
--

LOCK TABLES `offer` WRITE;
/*!40000 ALTER TABLE `offer` DISABLE KEYS */;
/*!40000 ALTER TABLE `offer` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `user`
--

DROP TABLE IF EXISTS `user`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `user` (
  `id_user` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  `first_name` varchar(60) COLLATE utf8_unicode_ci NOT NULL,
  `last_name` varchar(60) COLLATE utf8_unicode_ci NOT NULL,
  `birthday` date NOT NULL,
  `email` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `password` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `salt` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `address` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `zip_code` varchar(10) COLLATE utf8_unicode_ci NOT NULL,
  `country` varchar(100) COLLATE utf8_unicode_ci NOT NULL,
  `phone` varchar(30) COLLATE utf8_unicode_ci NOT NULL,
  `roles` int(11) NOT NULL,
  PRIMARY KEY (`id_user`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `user`
--

LOCK TABLES `user` WRITE;
/*!40000 ALTER TABLE `user` DISABLE KEYS */;
/*!40000 ALTER TABLE `user` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `user_affiliation`
--

DROP TABLE IF EXISTS `user_affiliation`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `user_affiliation` (
  `id_user_affiliation` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) NOT NULL,
  `affiliation_id` int(11) NOT NULL,
  PRIMARY KEY (`id_user_affiliation`),
  KEY `user_id` (`user_id`),
  KEY `affiliation_id` (`affiliation_id`),
  CONSTRAINT `user_affiliation_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user` (`id_user`),
  CONSTRAINT `user_affiliation_ibfk_2` FOREIGN KEY (`affiliation_id`) REFERENCES `affiliation` (`id_affiliation`)
)
```



```

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `user_affiliation`
--

LOCK TABLES `user_affiliation` WRITE;
/*!40000 ALTER TABLE `user_affiliation` DISABLE KEYS */;
/*!40000 ALTER TABLE `user_affiliation` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `user_auth`
--

DROP TABLE IF EXISTS `user_auth`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `user_auth` (
  `id_user_auth` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) NOT NULL,
  `nick_name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `profile_picture` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `token` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`id_user_auth`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `user_auth_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user`
  (`id_user`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `user_auth`
--

LOCK TABLES `user_auth` WRITE;
/*!40000 ALTER TABLE `user_auth` DISABLE KEYS */;
/*!40000 ALTER TABLE `user_auth` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `user_diver`
--

DROP TABLE IF EXISTS `user_diver`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `user_diver` (
  `id_user_diver` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) NOT NULL,
  `affiliation_id` int(11) NOT NULL,
  `first_dive` date NOT NULL,
  `duration` time NOT NULL,
  `depth_max` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `comment` text COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`id_user_diver`),
  UNIQUE KEY `user_id` (`user_id`),
  UNIQUE KEY `affiliation_id` (`affiliation_id`),
  CONSTRAINT `user_diver_ibfk_1` FOREIGN KEY (`affiliation_id`) REFERENCES
`affiliation` (`id_affiliation`),

```



```

    CONSTRAINT `user_diver_ibfk_2` FOREIGN KEY (`user_id`) REFERENCES `user`
    (`id_user`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
/*!40101 SET character set client = @saved_cs_client */;

--
-- Dumping data for table `user_diver`
--

LOCK TABLES `user_diver` WRITE;
/*!40000 ALTER TABLE `user_diver` DISABLE KEYS */;
/*!40000 ALTER TABLE `user_diver` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

```



11.6 – Glossaire / Lexique

MOT CLE	DESCRIPTION
CRUD	C : create (INSERT) R : read (SELECT) U : update (UPDATE) D : delete (DELETE)
UML	Unified M odeling L anguage
MVC	M odel V iew C ontroller
BD	B ase de D onnées
SQL	S tructured Q uery L anguage
SGBDR	S ystème de G estion de B ase de D onnées R elationnelle
ORM	O bject- R elational M apping
IHM	Interface H omme M achine
DCU	Diagramme de C as d' U tilisation
DAC	Diagramme d' A ctivité
DNAV	Diagramme de N avigation
DCL	Diagramme de C lasse
DSS	Diagramme de S équence S ystème
DSD	Diagramme de S équence D étaillé
DPL	Diagramme de D épLoiement