

RAPPORT DE STAGE

Olivier HENRY
Concepteur Développeur Informatique
du 20 Juin 2016 au 19 Août 2016



"C'est toujours l'impatience de gagner qui fait perdre",
Louis XIV cité dans "L'immortel" de FOG.

SOMMAIRE

<u>Chapitre 1 - Présentation générale.....</u>	<u>5</u>
<u>1.1 - Avant-propos.....</u>	<u>6</u>
<u>1.2 - Abstract.....</u>	<u>7</u>
<u>1.3 - Remerciements.....</u>	<u>8</u>
<u>1.4 - La démarche générale.....</u>	<u>9</u>
<u>Chapitre 2 - Cahier des charges.....</u>	<u>10</u>
<u>2.1 - Présentation de la solution Adiict.....</u>	<u>11</u>
<u>2.1.1 Capture d'écran de l'accueil d'adiict.....</u>	<u>12</u>
<u>2.1.2 Capture d'écran d'un chemin de fer dans adiict.....</u>	<u>12</u>
<u>2.2 - Présentation du client.....</u>	<u>13</u>
<u>2.3 - Contexte de développement et technologies imposées.....</u>	<u>14</u>
<u>2.3.1 Au lancement d'adiict.....</u>	<u>15</u>
<u>2.4 - Cahier des charges.....</u>	<u>16</u>
<u>2.4.1 Définition des besoins fonctionnels.....</u>	<u>16</u>
<u>2.4.2 Copies d'écran de la solution ellipse utilisée par Bayard.....</u>	<u>17</u>
<u>2.4.2.1 Ellipse - Le chemin de fer</u>	<u>17</u>
<u>2.4.2.2 Ellipse - Configuration des parutions et des folios</u>	<u>17</u>
<u>2.4.2.3 Ellipse - saisie des imprimeurs.....</u>	<u>18</u>
<u>2.4.2.4 Ellipse - saisie d'un profil.....</u>	<u>18</u>
<u>Chapitre 3 - Le planning.....</u>	<u>19</u>
<u>3.1 - Prémisses.....</u>	<u>20</u>
<u>3.2 - Diagramme de GANTT.....</u>	<u>21</u>
<u>3.2.1 Front Office.....</u>	<u>21</u>
<u>3.2.2 Génération PDF et contrôle en amont.....</u>	<u>21</u>
<u>3.2.3 Chemin de fer.....</u>	<u>21</u>
<u>3.2.4 Back Office.....</u>	<u>21</u>
<u>Chapitre 4 - Analyse.....</u>	<u>22</u>
<u>4.1 - Les diagrammes de cas d'utilisation.....</u>	<u>23</u>
<u>4.1.1 - Le diagramme du projet Bayard.....</u>	<u>23</u>
<u>4.1.2 - Le diagramme «Gérer les publications».....</u>	<u>24</u>
<u>4.1.3 - Le diagramme «Gérer les titres».....</u>	<u>25</u>
<u>4.1.4 - Fiche de description textuelle d'un cas d'utilisation.....</u>	<u>26</u>
<u>4.2 - Les maquettes.....</u>	<u>28</u>
<u>4.2.1 - Gérer les titres.....</u>	<u>28</u>
<u>4.2.2 - Gérer les titres - Modifier un titre.....</u>	<u>29</u>
<u>4.2.3 - Gérer les titres - Ajouter un titre.....</u>	<u>29</u>
<u>4.2.4 - Gérer les titres - Ajouter un titre (suite).....</u>	<u>30</u>
<u>4.2.5 - Gérer les titres – Ajouter un titre(fin).....</u>	<u>30</u>
<u>4.3 - Le diagramme de navigation.....</u>	<u>31</u>
<u>4.4 - Les Diagrammes de Séquence Système.....</u>	<u>32</u>
<u>Chapitre 5 - Conception de la Base de Données.....</u>	<u>33</u>
<u>5.1 - La démarche utilisée.....</u>	<u>34</u>
<u>5.1.1 Le dictionnaire des données.....</u>	<u>35</u>
<u>5.1.2 Les règles de gestion.....</u>	<u>36</u>
<u>5.1.3 La matrice des dépendances fonctionnelles.....</u>	<u>37</u>
<u>5.1.4 Le graphe des dépendances fonctionnelles.....</u>	<u>38</u>

<u>5.2 - Le diagramme de classes.....</u>	39
5.2.1 Traduction du Graphe de dépendances fonctionnelles en diagramme de classe entité.....	39
5.2.2 Le diagramme de classes.....	40
<u>5.3 - Le modèle physique de données.....</u>	41
<u>5.4 - SQL : le LDD.....</u>	42
<u>Chapitre 6 - Conception de l'application.....</u>	48
<u>6.1 - Le diagramme de séquence détaillé (ajouter un titre).....</u>	49
6.1.1 A propos du diagramme de séquence détaillé.....	50
<u>Chapitre 7 - Développement.....</u>	51
<u>7.1 - Technologies utilisées.....</u>	52
<u>7.2 - Copies d'écrans.....</u>	53
7.2.1 Les titres.....	53
7.2.2 La recherche d'un titre.....	53
7.2.3 Modification d'un titre.....	54
7.2.4 L'ajout d'un titre – étape 1.....	54
7.2.5 Ajout d'un titre - Etape 2.....	55
7.2.6 Front-office - Créer un chemin de fer.....	55
7.2.7 Front-office – Créer un chemin de fer(suite).....	56
7.2.8 Front-office – Modifier un chemin de fer.....	56
<u>7.3 Arborescence du projet.....</u>	57
<u>7.4 - Codes statiques des écrans.....</u>	58
7.4.1 index.html.twig.....	58
7.4.2 titre.html.....	59
7.4.3 buttons.html.....	60
7.4.4 recherche.html.....	60
7.4.5 pagination.html.....	61
<u>7.5 - Codes dynamiques des écrans.....</u>	62
7.5.1 app.js.....	62
7.5.2 titreController.js.....	62
7.5.3 titreService.js.....	66
7.5.4 FormService.js.....	67
<u>7.6 - La partie back-end en php.....</u>	69
7.6.1 Connexion.php.....	69
7.6.2 Titre.php.....	69
7.6.3 IDAO.php.....	71
7.6.4 TitreDAO.php.....	72
<u>7.7 - Les controls php.....</u>	75
7.7.1 GetTitre.php.....	75
7.7.2 SaveTitre.php.....	76
7.7.3 DeleteTitre.php.....	77
<u>Chapitre 8 - Déploiement.....</u>	78
<u>8.1 - Le diagramme de déploiement.....</u>	79
<u>8.2 - Le déploiement.....</u>	80
<u>Chapitre 9 - La gestion de projet.....</u>	81
<u>9.1.1 - SUBVERSION.....</u>	82
<u>Chapitre 10 - Conclusion.....</u>	83
<u>Chapitre 11 - Annexes.....</u>	85
<u>11.1 - Correspondances Projet/Reac.....</u>	86
<u>11.2 Cahier des charges & Spécificités Techniques.....</u>	87
11.2.1 Autres documents.....	87
<u>11.3 - Outils utilisés ... pour quels objectifs ?.....</u>	88
<u>11.4 - Bibliographie et Webographie.....</u>	89

<u>11.4.1 - Bibliographie.....</u>	89
<u>11.4.2 - Webographie.....</u>	90
<u>11.4.2.1 - HTML.....</u>	90
<u>11.4.2.2 - PHP.....</u>	90
<u>11.4.2.3 - SQL.....</u>	90
<u>11.4.2.4 - AngularJS.....</u>	90
<u>11.4.2.5 - Linq.js.....</u>	90
<u>11.4.2.6 - MySQL.....</u>	90
<u>11.4.2.7 - JavaScript.....</u>	90
<u>11.4.2.8 Boostrap.....</u>	90
<u>11.5 - Glossaire/Lexique.....</u>	91
<u>Chapitre 12 - Tables.....</u>	92
<u>12.1 - Table des illustrations.....</u>	93

CHAPITRE 1 - PRÉSENTATION GÉNÉRALE

1.1 - AVANT-PROPOS

A travers ce rapport, je vais vous présenter de manière synthétique ma première expérience professionnelle au sein d'un service informatique d'ingénierie logicielle.

Lors de cette expérience, j'ai pu mettre en pratique, les compétences acquises lors de ma formation chez M2i.

Ce rapport retrace la réflexion que j'ai dû mener pour comprendre et analyser les besoins du projet qui m'a été confié par l'entreprise, et présente la démarche, les méthodologies et les outils que j'ai utilisé pour y répondre. Pour finalement aboutir à la conception et au codage de l'application.

1.2 - ABSTRACT

I will present in this internship report, the methodology I use to translate bayard company's needs, in computer language. First, with the analysis phase, followed by the design phase, leading to the development and deployment that will take place within the O2i group's adiict solution. This process uses the Unified Modeling Language (UML). This method is largely based on Pascal Buguet's courses, referent trainer during my training, as well as that advocated by Pascal Roques in his book "*UML2 Pour les bases de données*" published by Eyrolles.

.....

Je vais vous présenter dans ce rapport de stage , la méthodologie que j'ai utiliser pour traduire les besoins de l'entreprise bayard, en langage informatique. Tout d'abord, avec la phase d'analyse, suivie de la phase de conception, pour aboutir au développement et au déploiement qui aura lieu au sein de la solution adiict du groupe O2I. Ce processus utilise le langage de modélisation unifié(UML). Cette méthode est largement inspiré des cours de Pascal Buguet, formateur référent lors de ma formation, ainsi que celle prônée par Pascal Roques dans son livre « UML2 Pour les bases de données » édité par Eyrolles.

1.3 - REMERCIEMENTS

Je tiens à remercier :

- Ma compagne qui m'a supporté et soutenu tout au long de cette aventure
- Pascal BUGUET et Sébastien MALORON pour la richesse de leurs enseignements.
- Mes compères étudiants, Johlian, Alban et Dominique avec qui j'ai passé d'excellents moments durant cette formation.
- Toute l'équipe de la solution adiict et plus particulièrement François LAURETTE qui m'a permis d'effectuer ce stage dans son service.
- Arthur JOSSEAU avec qui j'ai travaillé en binôme et beaucoup appris.
- L'entreprise O2i qui souhaite m'embaucher à l'issue de mon stage pour continuer l'aventure.

1.4 - LA DÉMARCHE GÉNÉRALE

Avant de commencer à développer, notre rôle de concepteur est de traduire les besoins des utilisateurs en code d'application.

Il y a tout d'abord la phase d'analyse qui permettra de formaliser les besoins utilisateurs sous forme de diagrammes, de fiches descriptives et maquettes compréhensibles par tous les intervenants du projet.

Lui succèdera ensuite la phase de conception, qui, à l'aide d'autres diagrammes permettra de définir la structure et le comportement du système.

Pour l'analyse et la conception, j'ai adopté une démarche UML largement inspirée des méthodologies prônées par Pascal Roques dans ses ouvrages « UML2 – Modéliser une application web, Eyrolles » et « UML2 Pour les bases de données, Eyrolles » enrichies par les cours de notre formateur, Pascal Buguet, et adaptées aux fonctionnalités du projet qui m'ont été confié par l'entreprise.

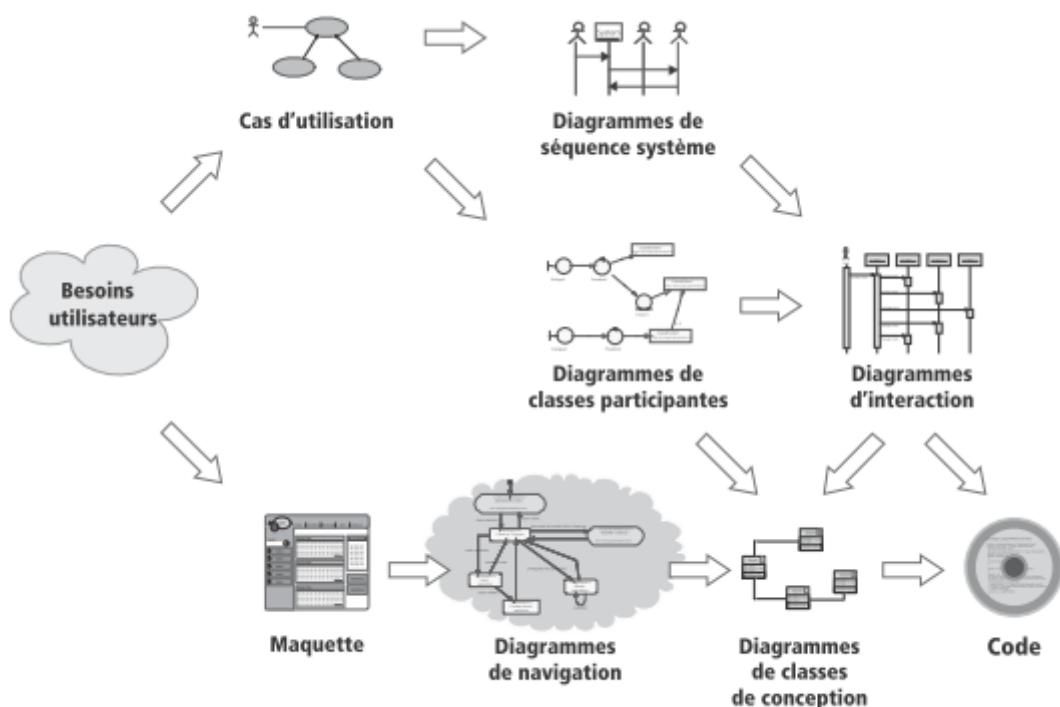


Figure 1–20 Schéma complet du processus de modélisation d'une application web

CHAPITRE 2 - CAHIER DES CHARGES

2.1 - PRÉSENTATION DE LA SOLUTION ADIICT

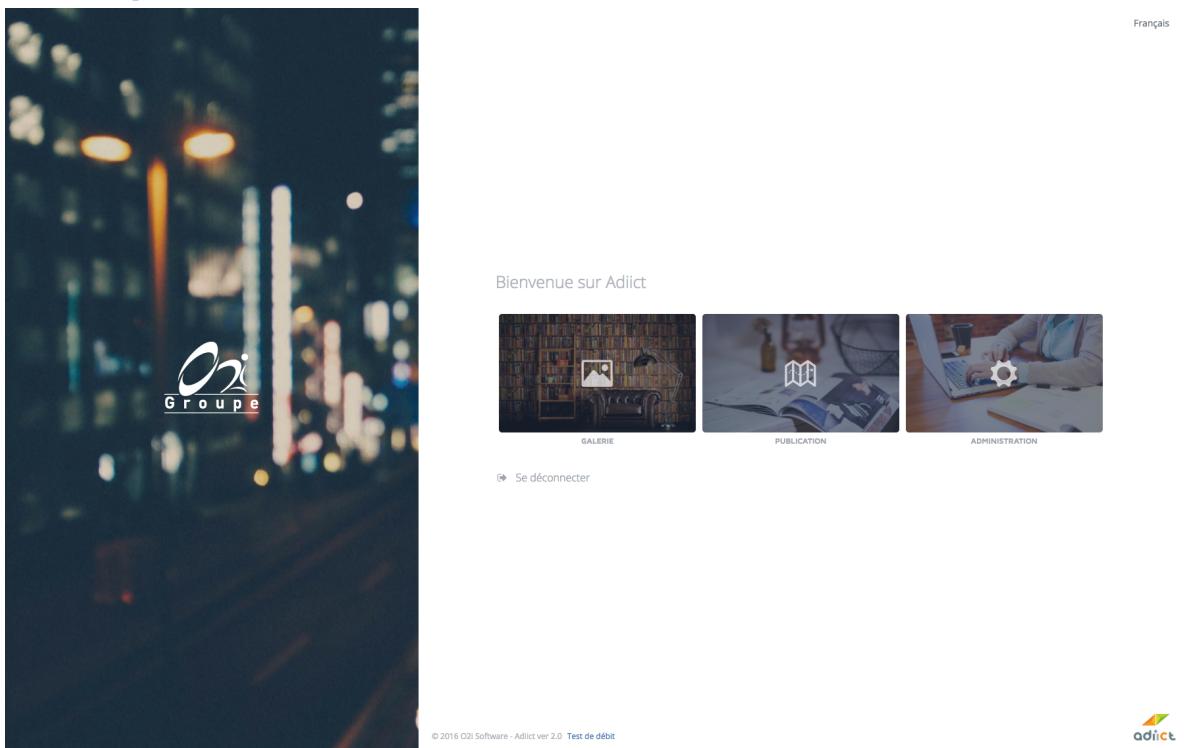
Adiict est une solution de **Digital Asset Management(DAM)** collaborative qui permet de centraliser la création, le partage, la validation et le stockage des ressources numériques pour répondre aux besoins spécifiques des métiers de la communication, de la production graphique, du marketing et de l'édition.

Adiict s'articule de 4 modules principaux autour du **Core** :

- Le module **Workflow Collaboratif** qui permet de visualiser, annoter et valider l'état d'avancement des contenus et de la mise en page des projets par les différents intervenants sur ceux-ci.
- Le module **Product Information Management(PIM)** qui permet la centralisation et la gestion des données et métadonnées des produits, l'import et l'export de celles-ci, ainsi que la génération automatique ou semi-automatique de catalogues.
- Le module **Chemin de fer** qui dépend des autres modules, permet le suivi et la production de publications(catalogues, magazines, flyers...) . Il permet notamment d'importer, exporter, manipuler, annoter, valider et gérer des pages au format pdf ou Indesign par les différents intervenants. Les fichiers de production sont gérés sur un serveur de fichier.
- Le module **Web To Print** qui permet de générer des sites de commandes de documents personnalisables(cartes de visites, papiers à entêtes, etc...) et d'en assurer la validation et le suivi.

Initialement, O2i se positionnait en tant qu'éditeur de logiciel avec la solution Adiict, puis s'est convertie au fil du temps en société de service en ingénierie informatique(SSII) pour adapter cette solution aux exigences plus spécifiques de ses clients.

2.1.1 Capture d'écran de l'accueil d'adiict



2.1.2 Capture d'écran d'un chemin de fer dans adiict

A screenshot of the Adiict software interface showing a document structure. The top navigation bar includes "PUBLICATION", "FICHIER", "EDITION", "FLIPBOOK", and "AIDE". On the right, a sidebar titled "Chemin de fer" has tabs for "DÉTAILS" (selected) and "RUBRIQUES". A message says "Sélectionnez un folio pour en afficher les détails." Below the sidebar is a grid of 29 numbered boxes (1 to 29) representing document pages. Each page box contains a small document icon. At the bottom of the grid, there is a button labeled "AFFICHER LE CHUTIER".

2.2 - PRÉSENTATION DU CLIENT

Le groupe Bayard est un groupe international spécialisé dans la presse, l'édition et le numérique au travers de ses différentes filiales éditrices:

- Bayard
- Bayard jeunesse
- Bayard Canada
- Milan
- Twenty Third
- Novalis

Le groupe Bayard publie 150 magazines pour 36 millions de lecteurs dans le monde, et affiche 4800 livres à son catalogue avec 700 nouvelles références par an.

Le groupe Bayard, c'est également 150 sites web à travers le monde.

2.3 - CONTEXTE DE DÉVELOPPEMENT ET TECHNOLOGIES IMPOSÉES

Initialement développée en PHP 4 et les composants d'interface développés en grande partie en JavaScript avec la librairie Dojo Toolkit, la solution Adiict se retrouve actuellement avec des problèmes de performances et de maintenabilité.

Le problème principal est le nombre de dépendances très conséquent que la librairie Dojo charge au lancement de l'application.

Une équipe de développement est donc actuellement en train d'assurer la migration du core et de ses modules en PHP 5, en se basant sur le framework CodeIgniter 2, et le framework AngularJS pour remplacer Dojo Toolkit.

Dans ce contexte, et ne sachant pas encore si le projet Bayard sera intégré à l'ancienne ou la nouvelle application, le responsable technique, Mr François Laurette, nous a confié la tache, Arthur Josseau, développeur confirmé nouvellement employé, et moi, de développer une partie du projet en dehors d'Adiict.

Cette partie qui correspond à l'arborescence de classement des parutions(Chemins de fer) devra donc être développée en PHP 5 pour la partie serveur et avec HTML5 , CSS3 et AngularJS pour la partie client. Elle sera, plus tard, intégrée à la solution Adiict.

2.3.1 Au lancement d'adiict

Comme on peut le constater, adiict utilise énormément de dépendances, ce qui allonge considérablement les temps de chargement. Cette capture d'écran a été prise sur localhost, je vous laisse donc imaginer le résultat en environnement de production.



2.4 - CAHIER DES CHARGES

Le cahier des charges a été rédigé par la chef de projet, et est disponible dans les annexes sous la rubrique Cahier des charges & Spécificités Techniques.

Tout au long de la phase d'analyse et de conception , j'ai pu m'appuyer sur ce document pour définir plus précisément les besoins fonctionnels et techniques. Il m'a fallut tout de même obtenir des renseignements complémentaires à l'oral, lors des réunions et discussions avec les membres de l'équipe adjoint. En toute honnêteté, il m'a fallut du temps avant de saisir toute la substance du projet.

La chef de projet, Aurore Bertrand, s'assure de faire la jonction entre l'équipe de développement et le client.

2.4.1 Définition des besoins fonctionnels

Cette définition des besoins fonctionnels correspond à la partie du projet sur lequel je travaille actuellement. Elle correspond à la première partie du projet.

Le groupe Bayard utilise actuellement un plugin de Adobe InDesign dénommé ellipse qui permet d'ajouter des informations métier à la fin du processus rédactionnel des chemins de fer. Cela implique de ressaisir manuellement pour chaque folio, de chaque chemin de fer, ces informations de façon répétitive avec un risque d'erreur et donc, une perte de productivité.

Le groupe Bayard veut donc:

- Pouvoir, à l'avance, définir dans le back-office, les informations métier. Ces informations seront gérables par le groupe publication et l'administrateur.
- Pouvoir, dans le même back-office, définir une arborescence débouchant sur une Publication(un titre) à laquelle sera liée toutes les informations de la partie fixe du document de cartographie fonctionnelle « Bayard_organigramme dev_V2.pdf » fourni.
- Pouvoir créer un chemin de fer en sélectionnant facultativement, l'arborescence précédemment définie.
- Ajouter également lors de la création du chemin de fer, les informations définies dans la partie variable du document cité précédemment.

2.4.2 Copies d'écran de la solution ellipse utilisée par Bayard

Pour mieux appréhender le projet, ci-joint, quelques copies d'écran de la solution actuellement utilisée par le groupe Bayard.

2.4.2.1 Ellipse – Le chemin de fer

Folio1	Folio2	Folio3	Folio4	Imprimeurs
001	002	091	092	Maury Phosphore
003	026	087	090	Maury Phosphore
027	046	047	006	Maury Phosphore
a				Bayard

2.4.2.2 Ellipse – Configuration des parutions et des folios

Nom	Profil	Imprimeurs	Format_	Format_
Couverture	X-1a_ICV2_3	Maury Phosphore	220	271
Cahier 1	X-1a_PSOLV	Maury Phosphore	220	271
Cahier 2	X-1a_PSOVL	Maury Phosphore	220	271
Cahier 3	X-1a_PSOLV	Bayard	220	271
Cahier 4	X-1a_ICV2_3	Bayard	220	271
Cahier 5	X-1a_ICV2_3	Bayard	220	271

2.4.2.3 Ellipse - saisie des imprimeurs

The screenshot shows the Ellipse software interface. In the center, a modal dialog box titled "Saisie d'un compte imprimeur" (Printer account creation) is open. It contains fields for "Nom" (Name) set to "Maury Phosphore", "Serveur" (Server) set to "fp.maury-imprimeur.fr", "Chemin" (Path) set to "/Phosphore/traiter", "Arborescence" (Arborescence) set to "Vrac", "Login" set to "bayard", and "Password" set to "Po5Domil6". Below these fields are buttons for "Supprimer" (Delete), "Effacer" (Clear), and "Enregistrer" (Save). At the bottom of the dialog is an orange "Test FTP" button. To the left of the dialog, a list of printers is displayed, with "Maury Phosphore" highlighted. The main interface shows various tabs like Magazine, Parution, Secteur, Gamma, Rubrique, Profil, Imprimeur, Utilisateur, Planning, Flux, and Digital.

2.4.2.4 Ellipse – saisie d'un profil

The screenshot shows the Ellipse software interface. A modal dialog box titled "Formulaire Profil" (Profile form) is open. It displays a table with two rows. The first row has "Nom" (Name) set to "Couverture" and "Profil" (Profile) set to "X-1a_ICV2_300". The second row has "Nom" (Name) set to "Intérieur" and "Profil" (Profile) set to "X-1a_PSOLVC_280". Below the table are buttons for "Supprimer" (Delete), "Effacer" (Clear), and "Enregistrer" (Save). To the left of the dialog, a list of profiles is shown, with "Couverture" and "Intérieur" listed. The main interface shows various tabs like Magazine, Parution, Secteur, Gamma, Rubrique, Profil, Imprimeur, Utilisateur, Planning, Flux, and Digital.

CHAPITRE 3 - LE PLANNING

3.1 - PRÉMISSES

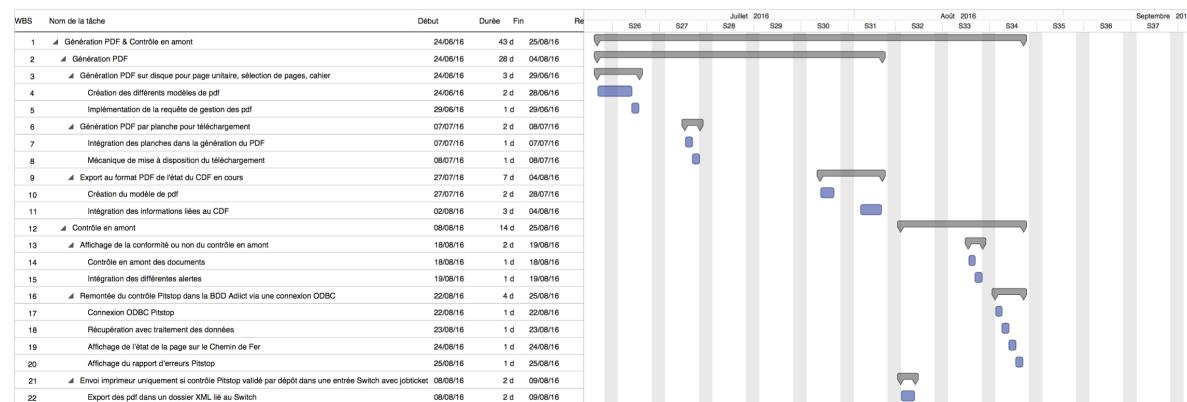
Les diagrammes de Gantt m'ont été généreusement fourni par la chef de projet, ils planifient et synthétisent la successions des taches du projet. Certaines taches peuvent dépendre d'un ou plusieurs taches précédente(s).

3.2 - DIAGRAMME DE GANTT

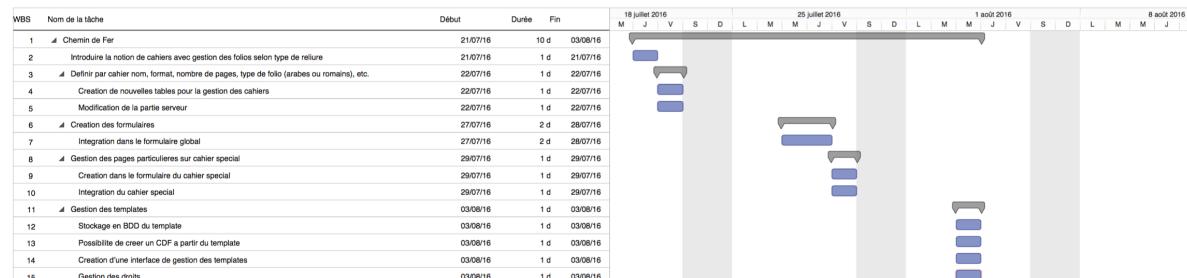
3.2.1 Front Office



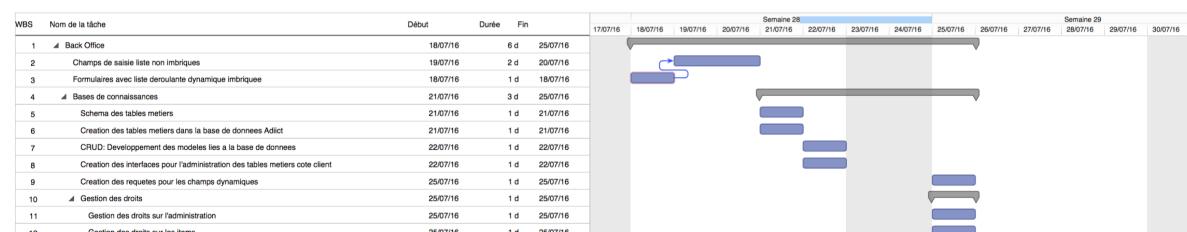
3.2.2 Génération PDF et contrôle en amont



3.2.3 Chemin de fer



3.2.4 Back Office



CHAPITRE 4 - ANALYSE

4.1 - LES DIAGRAMMES DE CAS D'UTILISATION

Les diagrammes de cas d'utilisation permettent d'illustrer les tâches fondamentales attendues du système d'un point de vue de leurs utilisateurs. Un diagramme de cas d'utilisation est un ensemble de cas d'utilisation, il permet de définir un processus métier.

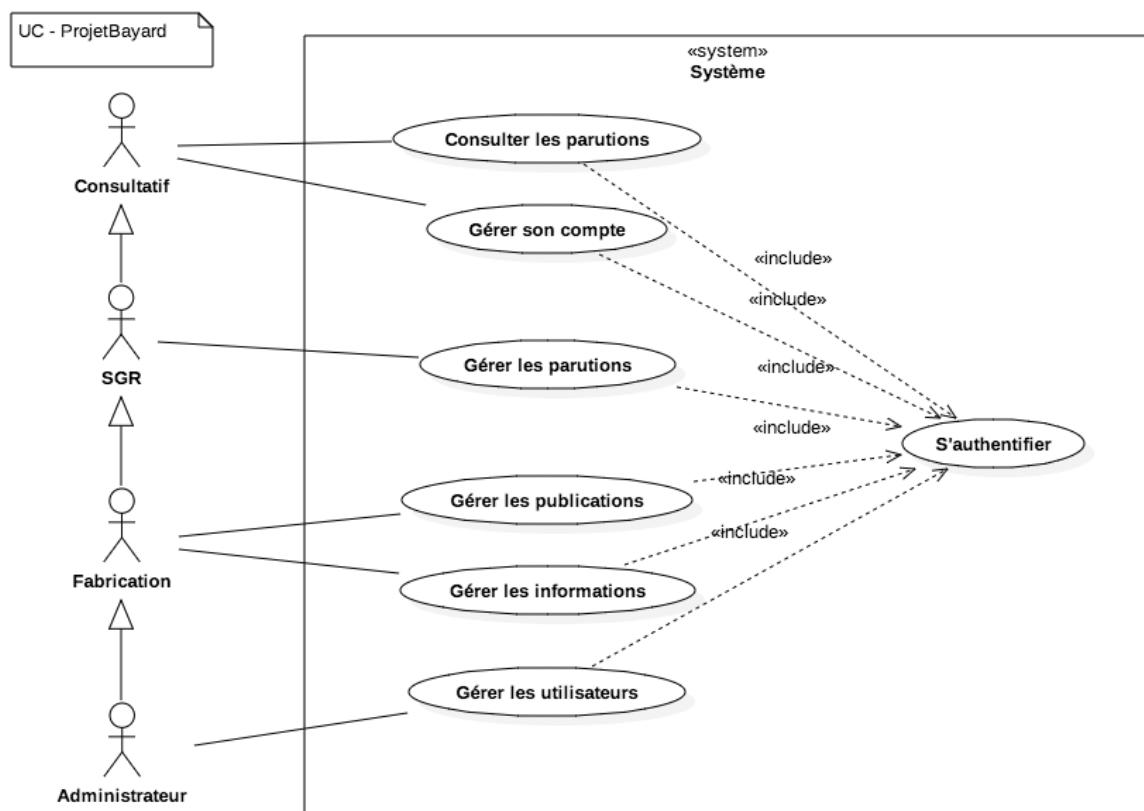
Un cas d'utilisation permet de modéliser un **besoin** utilisateur. Il permet de formaliser les **interactions** des **acteurs** avec le **système**.

A partir du document des spécificités techniques fourni par l'entreprise. J'ai dû tout d'abord définir les acteurs qui sont les représentants des différents utilisateurs ou rôles qui interagissent avec le système.

J'en ai déduit, qu'il y avait des acteurs avec les rôles:

- Consultatif
- SGR(Secrétaire généraux de rédaction, secrétaires de rédaction)
- Fabrication(Graphistes, Maquettistes)
- Administrateur

4.1.1 - Le diagramme du projet Bayard

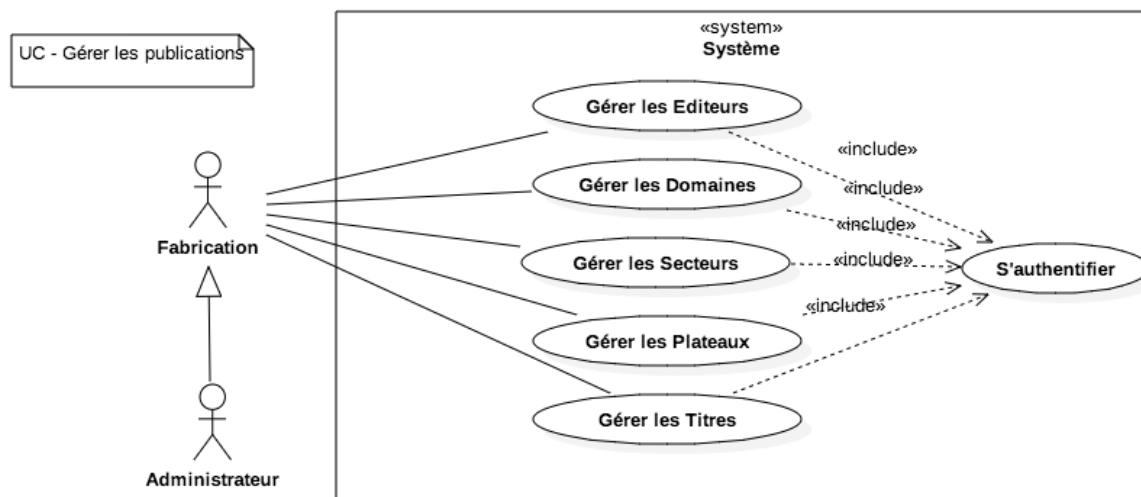


4.1.2 - Le diagramme «Gérer les publications»

Gérer les publications est un diagramme de cas d'utilisation qui permet de décomposer le Cas d'utilisation « **Gérer les publications** » du diagramme de cas d'utilisation du Projet Bayard. Le diagramme Projet Bayard est une représentation plus macroscopique du système.

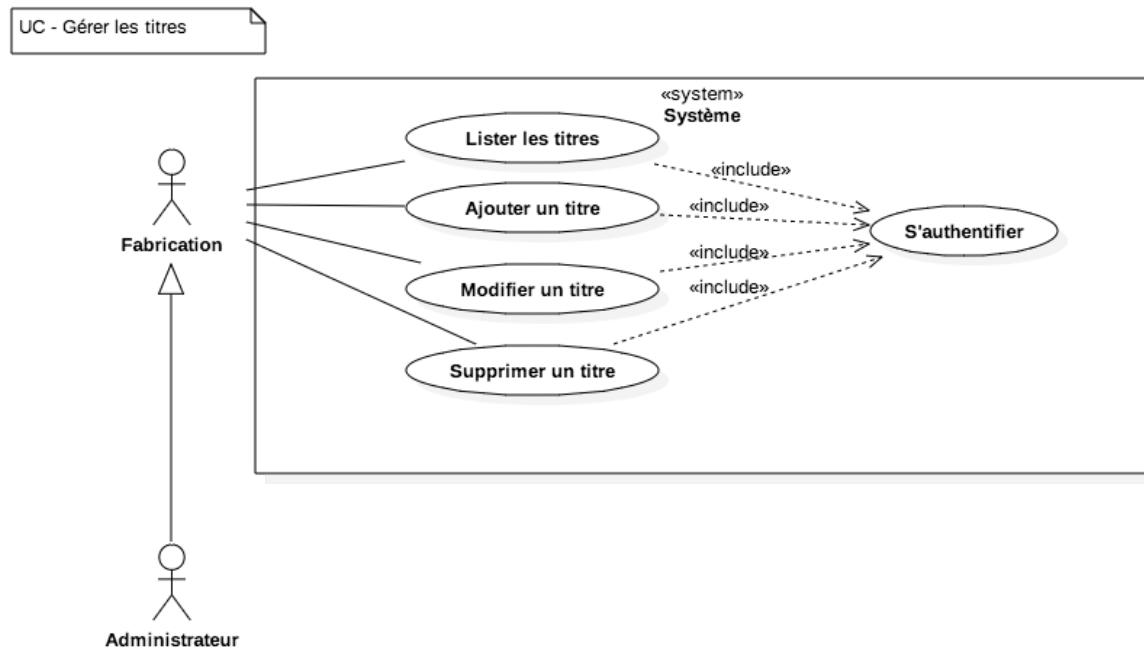
On notera la relation d'inclusion avec son **stéréotype <<include>>** qui permet de préciser qu'un cas d'utilisation en utilise un autre avec une méthode externe.

Dans notre cas, nous ne nous occuperons pas du système d'authentification qui existe déjà au sein de la solution adjointe.



4.1.3 - Le diagramme «Gérer les titres»

Me rendant compte que l'écriture des fiches de description textuelles et des diagrammes de séquence système seraient encore relativement complexes avec le cas d'utilisation gérer les titres, j'ai choisi de le décomposer encore avec un troisième niveau.



4.1.4 - Fiche de description textuelle d'un cas d'utilisation

Pour chaque cas d'utilisation, j'ai rédigé ensuite une fiche textuelle qui présente le scénario nominal, ses scénariis d'erreur ainsi que ses scénariis alternatifs et leurs erreurs(si besoin).

Ci dessous la fiche de description textuelle du cas d'utilisation « Ajouter un titre »

Etapes	Description
Identification du CU	<p>Titre : Ajouter un titre Résumé : Remplir le formulaire titre en fonction de l'arborescence présélectionnée Acteurs : Fabrication, Administrateur Date de création : 03/07/2016 Date de dernière modification : 03/07/2016 Version : 1.0 Auteur : Olivier HENRY</p>
Pré-conditions	<p>Boite de dialogue disponible Au moins une arborescence (Éditeur → domaine → secteur → plateau) existe Connexion à la Base de donnée disponible.</p>
Scenario nominal	<p>L'utilisateur clique sur le bouton Ajouter requête asynchrone(JavaScript) vers le serveur pour récupérer(JSON) la liste des éditeurs. une boite de dialogue s'affiche avec une liste déroulante pour choisir un éditeur. L'utilisateur sélectionne un éditeur. requête asynchrone(JavaScript) vers le serveur pour récupérer(JSON) la liste des domaines correspondants à l'éditeur. la liste déroulante pour choisir un domaine s'affiche. l'utilisateur sélectionne un domaine. requête asynchrone(JavaScript) vers le serveur pour récupérer(JSON) la liste des secteurs correspondants au domaine. la liste déroulante pour choisir un secteur s'affiche. l'utilisateur sélectionne un secteur. requête asynchrone(JavaScript) vers le serveur pour récupérer(JSON) la liste des plateaux correspondants au secteur. la liste déroulante pour choisir un plateau s'affiche. l'utilisateur sélectionne un plateau. Affichage des autres champs du formulaire Requêtes asynchrones pour récupérer les types et les partenaires afin de remplir leurs liste déroulantes respectives. L'utilisateur rempli le formulaire et clique sur Sauvegarder Contrôle des saisies côté client en JavaScript requête asynchrone vers le serveur pour y envoyer les données en JSON Traitement de la requête côté serveur et contrôle des données Insertion des données dans la BD</p>

	Fermeture automatique de la boite de dialogue et ajout du titre dans la « datagrid »
Scenarii d'erreurs du cas nominal	Échec de l'envoi ou de la récupération des données lors des requêtes asynchrones. Échec d'insertion dans la base de données. Affichage d'un message d'erreur.
Scenarii alternatifs	L'utilisateur n'a pas correctement rempli tous les champs du formulaire, affichage d'un message pour le lui indiquer. L'utilisateur clique sur annuler ou ferme la boite de dialogue. Cette dernière se ferme.
Post-conditions	Les données ont été correctement persistées et ajoutées à la grille de données(datagrid)
Exigences non fonctionnelles	Requêtes asynchrones rapides < 1 secondes
Besoins d'IHM	Boite de dialogue

4.2 - LES MAQUETTES

Avec l'outil Balsamiq Mockups et en me basant sur les diagrammes de cas d'utilisation, j'ai pu créer les maquettes qui présentent la future interface de gestion.

4.2.1 - Gérer les titres

Editeur	Domaine	Secteur	Plateau	Nom	Code	Action
Bayard	Presse	Jeunesse	Lecture	Je Bouquine	JBQN	Modifier Supprimer
Bayard	Presse	Jeunesse	Lecture	Je Bouquine Hors Série	JBQH	Modifier Supprimer
Bayard	Presse	Jeunesse	Lecture	Je Bouquine Supplément	JBQS	Modifier Supprimer
Bayard	Presse	Jeunesse	Lecture	J'aime Lire	JLIN	Modifier Supprimer
Bayard	Presse	Jeunesse	Lecture	J'aime Lire Hors Série	JLIH	Modifier Supprimer
Bayard	Presse	Jeunesse	Lecture	J'aime Lire Supplément	JLIS	Modifier Supprimer
Bayard	Presse	Jeunesse	Lecture	Mes premiers J'aime lire	MPRN	Modifier Supprimer
Bayard	Presse	Jeunesse	Petite Enfance	Pomme d'api	PAPN	Modifier Supprimer
Bayard	Presse	Jeunesse	Petite Enfance	Pomme d'api supplément	PAPS	Modifier Supprimer
Bayard	Presse	Jeunesse	Petite Enfance	Pomme d'api Hors Série	PAPH	Modifier Supprimer

Début Précédent 1 2 Suivant Fin

4.2.2 - Gérer les titres - Modifier un titre

A Web Page

Editeur	Domaine	Secteur	Plateau	Titre	Code	Hauteur	Largeur	Type	Partenaire
Bayard	Presse	Jeunesse	Lecture	Le Bouquine	JBQN	190	245	Mensuel	ADL

Action

Code	Modifier	Supprimer
JBQN	Modifier	Supprimer
JBQH	Modifier	Supprimer
JBQS	Modifier	Supprimer
JLIN	Modifier	Supprimer
JLIH	Modifier	Supprimer
JLIS	Modifier	Supprimer
MPRN	Modifier	Supprimer
PAPN	Modifier	Supprimer
PAPS	Modifier	Supprimer
PAPH	Modifier	Supprimer

4.2.3 - Gérer les titres - Ajouter un titre

A Web Page

Editeur	Domaine	Secteur	Plateau	Nom
Bayard	Presse	Jeunesse	Lecture	Je Bouquine
Bayard	Presse	Jeunesse	Lecture	Je Bouquine Hors Série

Action

Code	Modifier	Supprimer
JBQN	Modifier	Supprimer
JBQH	Modifier	Supprimer
JBQS	Modifier	Supprimer
JLIN	Modifier	Supprimer
JLIH	Modifier	Supprimer
JLIS	Modifier	Supprimer
MPRN	Modifier	Supprimer
PAPN	Modifier	Supprimer
PAPS	Modifier	Supprimer
PAPH	Modifier	Supprimer

4.2.4 - Gérer les titres - Ajouter un titre (suite)

A Web Page

Code	Action
JBQN	Modifier Supprimer
JBQH	Modifier Supprimer
JBQS	Modifier Supprimer
JLIN	Modifier Supprimer
JLIH	Modifier Supprimer
JLIS	Modifier Supprimer
MPRN	Modifier Supprimer
PAPN	Modifier Supprimer
PAPS	Modifier Supprimer
PAPH	Modifier Supprimer

4.2.5 - Gérer les titres – Ajouter un titre(fin)

A Web Page

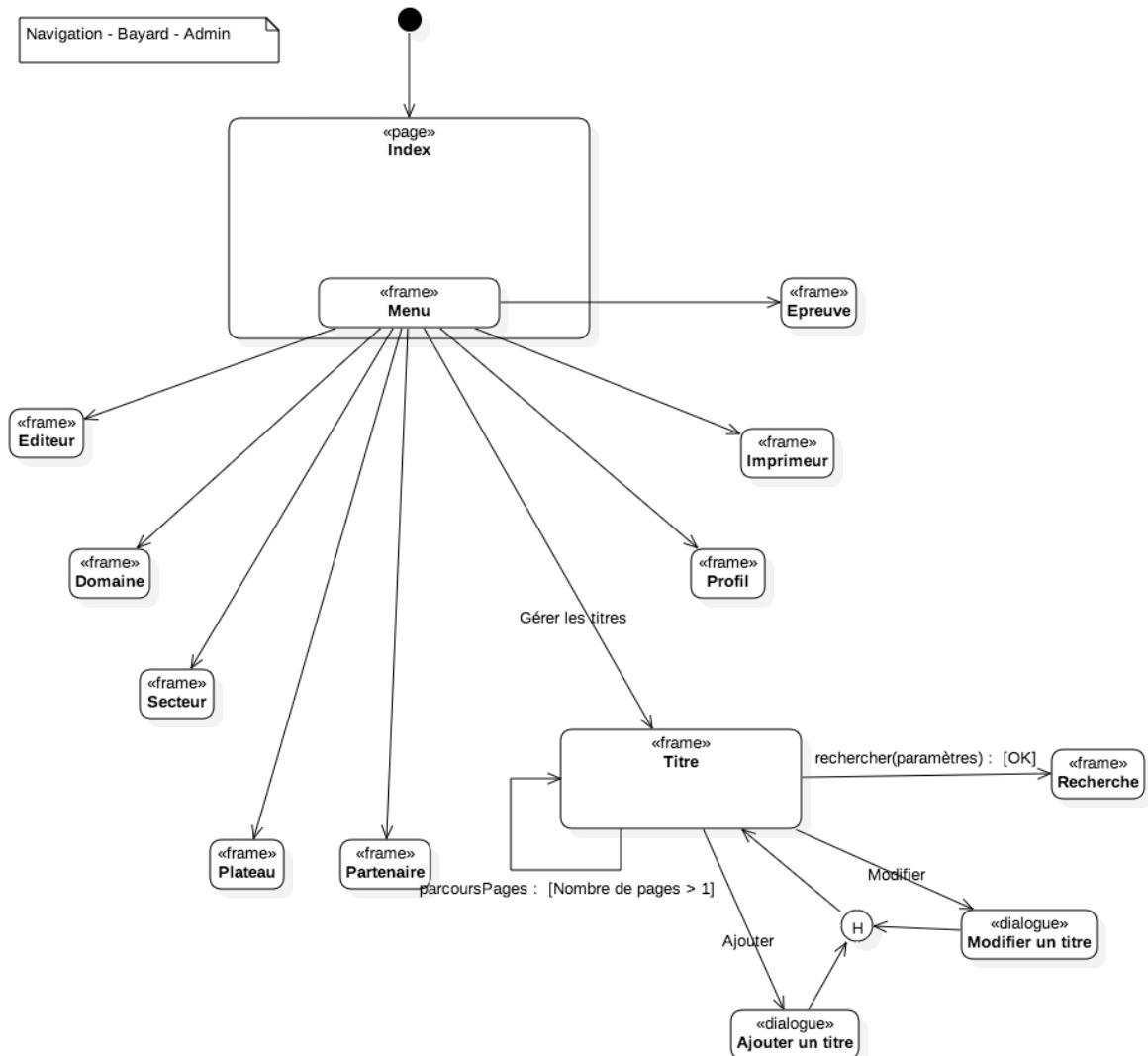
Code	Action
JBQN	Modifier Supprimer
JBQH	Modifier Supprimer
JBQS	Modifier Supprimer
JLIN	Modifier Supprimer
JLIH	Modifier Supprimer
JLIS	Modifier Supprimer
MPRN	Modifier Supprimer
PAPN	Modifier Supprimer
PAPS	Modifier Supprimer
PAPH	Modifier Supprimer

4.3 - LE DIAGRAMME DE NAVIGATION

Le diagramme de navigation, permet, en s'a aidant des maquettes, de représenter la cinématique de l'application, la navigation entre les différentes pages ou frames de celle-ci. Il est d'une grande aide pour nous aider à définir nos futurs controllers et views.

Dans notre cas, les liens de navigations correspondront à des controllers et les boutons à des méthodes au sein de ces controllers. Comme nous allons développer une Single Page Application, nous n'avons donc qu'un seul élément avec le stéréotype «page», le reste sera représenté par des «frame» ou des «dialogue».

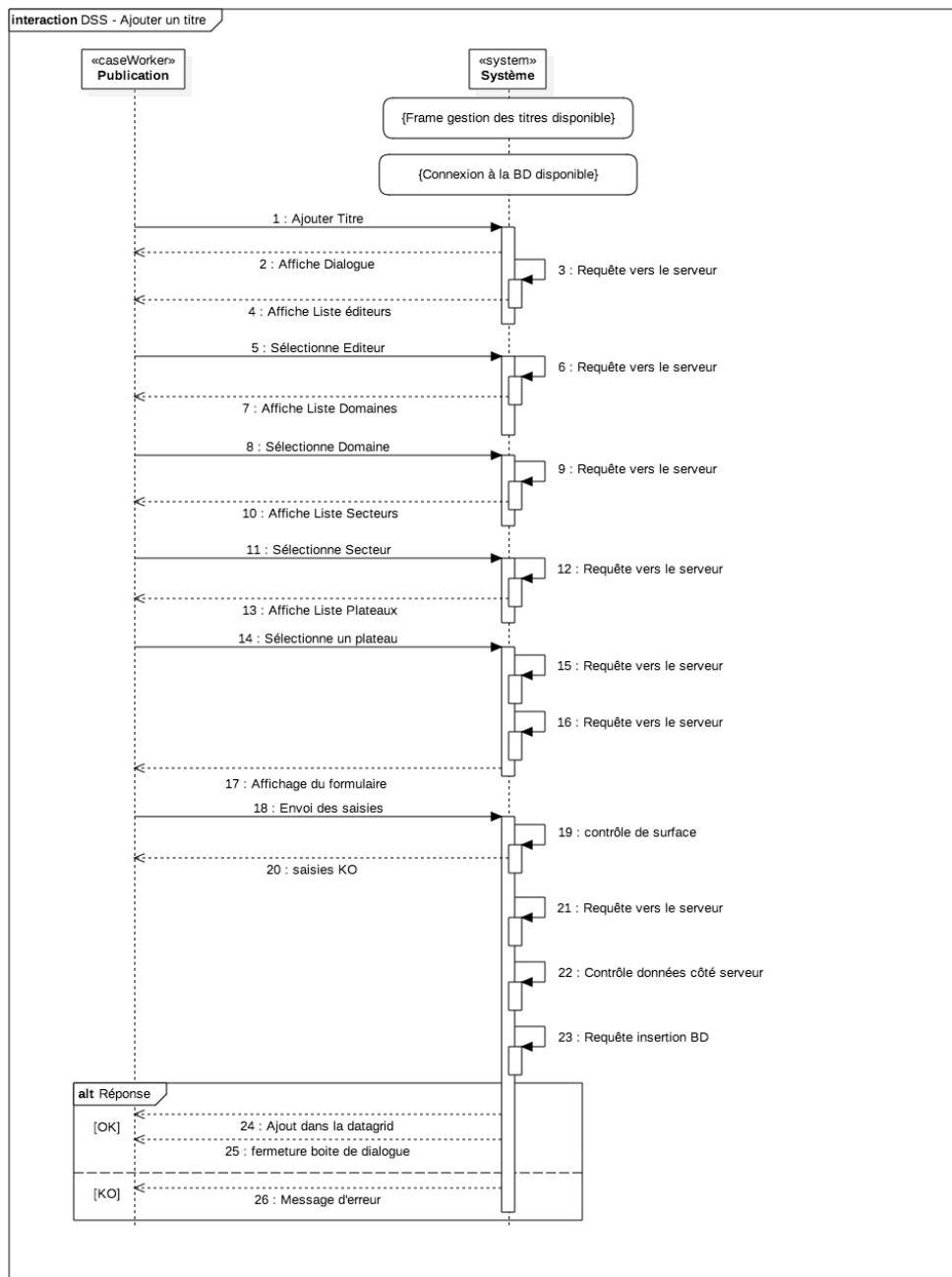
Toutes les frames dépendantes de la «frame» Menu adopteront un comportement identique, pour simplifier la lecture, je n'ai volontairement représenté que la gestion des titres.



4.4 - LES DIAGRAMMES DE SÉQUENCE SYSTÈME

Les diagrammes de séquences système permettent de modéliser les messages transmis entre les acteurs et le système.

On conçoit un diagramme de séquence système pour chacun des scénarios d'un cas d'utilisation(nominal, erreur, alternatif...).



CHAPITRE 5 - CONCEPTION DE LA BASE DE DONNÉES

5.1 - LA DÉMARCHE UTILISÉE

Pour concevoir et normaliser la base de données, j'ai analysé les documents que j'avais à ma disposition pour utiliser la **méthode des dépendances fonctionnelles** dont la démarche consiste à:

1. Créer un dictionnaire de données à partir des documents existants
2. Créer une matrice des dépendances fonctionnelles
3. Modéliser un graphe de dépendances fonctionnelles
4. Traduire ce graphe en diagramme de classes Entité
5. Traduire ce diagramme de classe en Modèle Logique de données(MLD)
6. Convertir le modèle logique en modèle physique de données(MPD)

Les informations n'étant pas complètes, j'ai dû utiliser la **méthode Chen**(analyse du discours) en interrogeant le directeur technique et la chef de projet, pour récolter ces dernières.

La **normalisation** consiste à rendre cohérentes les données par rapport à la réalité, en évitant les anomalies de mise à jour et la redondance d'information.

L'objectif est d'obtenir un modèle de données normalisé qui respecte la forme normale de Boyce Codd(**BCNF**) dont tous les attributs:

- Contiennent des données atomiques(1NF)
- Contiennent une valeur scalaire(1NF)
- Contiennent des valeurs non répétitives(1NF)
- Contiennent des valeurs constantes dans le temps(1NF)
- Non clef ne dépendent pas à d'une partie de la clef (2NF)
- Non clef ne dépendent pas d'un ou plusieurs attributs non clef(3NF)
- Non clef ne sont pas source de dépendance fonctionnelle vers une partie de la clef(BCNF)

5.1.1 Le dictionnaire des données

Pour définir le **dictionnaire des données**, j'ai analysé le document Bayard_organigramme que vous pourrez trouver dans les annexes. Les informations étant incomplètes, j'ai obtenu d'autres informations oralement. Notamment, que Profil, Epreuve et Imprimeur, seront associés aux différents cahiers eux même contenus dans un chemin de fer.

La partie fixe correspond aux données liées à un titre.

La partie variable correspond à celles contenues par un chemin de fer.

La partie automatique correspond à la génération d'un fichier xml qui sera traitée beaucoup plus tard dans le projet.

The diagram illustrates the structure of a title record (Titre) and its associated parts:

- Partie fixe (Fixed Part):**
 - Libellé Titre:** Je Bouquine (highlighted in red)
 - Code Titre:** JBQN (highlighted in blue)
 - Format:**
 - Hauteur (en mm): 190 (highlighted in yellow)
 - Largeur (en mm): 245 (highlighted in yellow)
 - Type:**
 - Quotidien
 - Hebdomadaire
 - Quinzomadaire
 - Mensuel (highlighted in yellow)
 - Bimestriel
 - Trimestriel
- Partie variable (Variable Part):**
 - Partenaires:**
 - ADL (highlighted in yellow)
 - EDD (highlighted in red)
 - Numéro Cdf:**
 - Numéro: 388 (highlighted in yellow)
 - Daté: JUIN 2016 (highlighted in yellow)
 - Date de mise en vente: 20 avril 2016 (highlighted in yellow)
 - Date de départ impression: 20 mars 2016 (highlighted in yellow)
- Partie automatique (Automatic Part):**
 - Xml:**
 - issueld: JBQN (highlighted in yellow)
 - issueNumber: 0388 (highlighted in yellow)
 - publicationDate: 2016-04-20T00:00:00 (highlighted in yellow)

Annotations in red and blue highlight specific fields and their types:

- nomTitre** and **codeTitre** point to the Libellé Titre and Code Titre fields.
- hauteurTitre** and **largeurTitre** point to the Hauteur and Largeur fields in the Format section.
- nomTypeTitre** points to the Type field in the Format section.
- dateMiseVente** and **dateDepartImpression** point to the Date de mise en vente and Date de départ impression fields in the Numéro Cdf section.
- titreSecondaire** points to the Xml section.
- nomPartenaire** points to the ADL entry in the Partenaires section.
- numeroCdf** points to the 388 entry in the Numéro Cdf section.
- dateCdf** points to the JUIN 2016 entry in the Date de départ impression field of the Numéro Cdf section.

5.1.2 Les règles de gestion

Ce sont des **règles métier** qui permettront d'ajouter des **contraintes** et **vérifications** lors des **insertions, modifications** ou **ajout** dans la base de donnée. Elles permettront notamment de **définir les dépendances fonctionnelles**, puis plus tard, les **multiplicités** et **contraintes d'intégrité référentielle** en fonction du comportement attendu par Bayard Presse.

En interrogeant mes différents interlocuteurs, j'ai pu définir :

- Un éditeur contient 0 ou plusieurs domaines
- Un domaine appartient à 1 éditeur et contient 0 ou plusieurs secteurs
- Un secteur appartient à 1 domaine et contient 0 ou plusieurs plateaux
- Un plateau appartient à 1 secteur et contient 0 ou plusieurs titres
- Un titre appartient à 1 plateau et contient 0 ou plusieurs chemin de fer
- Un titre contient également 1 type(Périodicité) et 1 partenaire
- Un type appartient à 0 ou plusieurs titres
- Un partenaire appartient à 0 ou plusieurs titres
- Un chemin de fer appartient à 0 ou 1 titre, contient 0 ou 1 titre secondaire, et contient 0 ou plusieurs cahiers
- Un cahier appartient à 1 chemin de fer, et contient 0 ou 1 profil, 0 ou 1 Epreuve, 0 ou 1 imprimeur
- Un profil appartient à 0 ou plusieurs cahiers
- Une épreuve appartient à 0 ou plusieurs cahiers
- un imprimeur appartient à 0 ou plusieurs cahiers et contient un type de dépôt
- un Type de dépôt appartient à 0 ou plusieurs imprimeur

5.1.3 La matrice des dépendances fonctionnelles

Après avoir analysé et nettoyé les éventuels doublons dans mon dictionnaire de données et déterminé les règles de gestion, j'y ai ajouté des identifiants qui seront plus tard les clefs primaires de mes tables.

Dans la matrice des dépendances fonctionnelles, j'ai recherché les dépendances fonctionnelles :

Il y a **dépendance fonctionnelle** entre deux attributs lorsque un attribut permet de définir, **une et une seule valeur d'un autre attribut**.

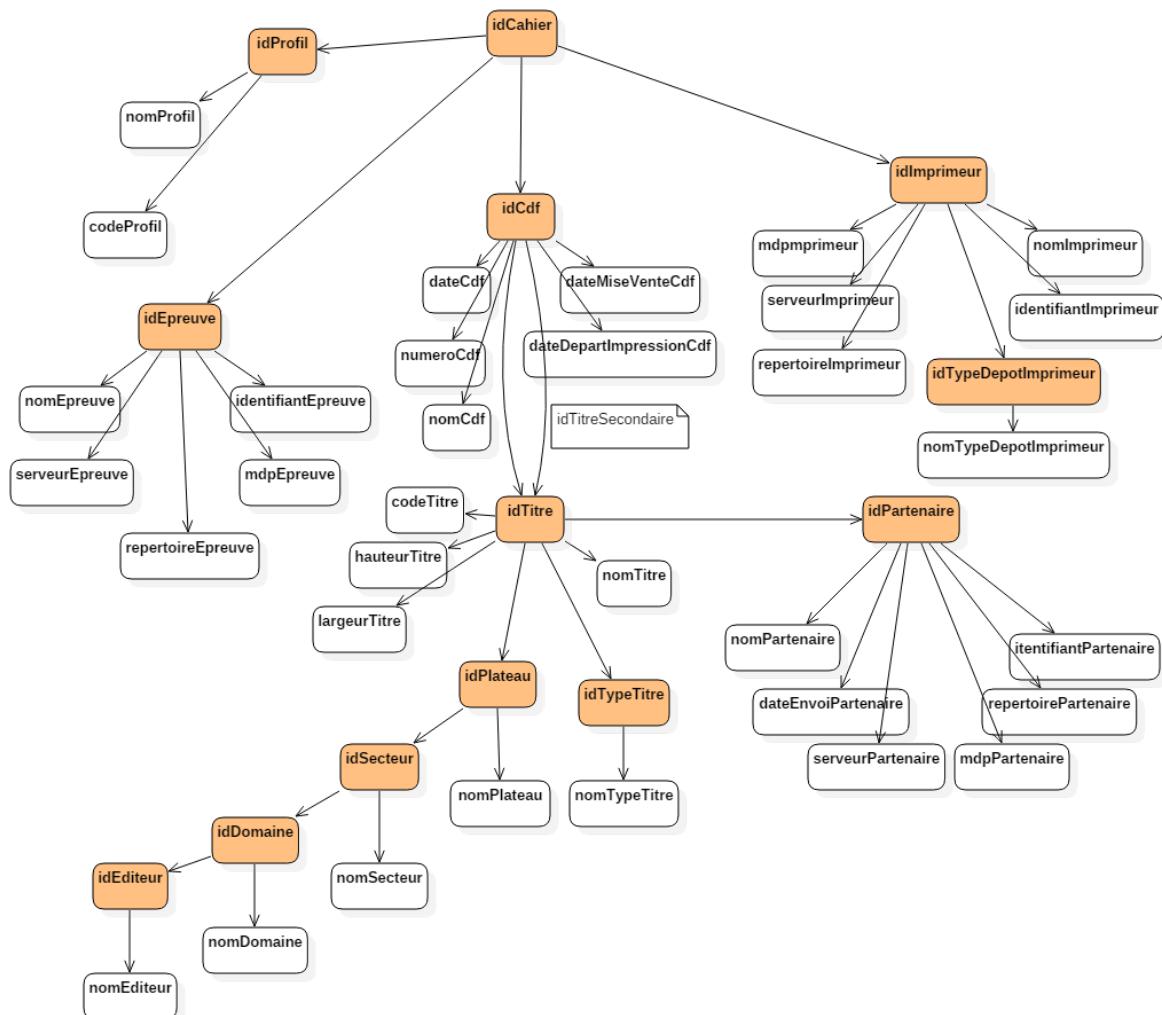
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44				
idTitre	x	x	x	x	x	x	x						x																																			
nomTitre		x																																														
hauteurTitre	2		x																																													
largeurTitre	3			x																																												
idTypeTitre	4				x	x																																										
nomTypeTitre	5					x																																										
codeTitre	6						x																																									
idPartenaire	7							x	x	x	x	x	x	x																																		
nomPartenaire	8							x																																								
serveurPartenaire	9								x																																							
identifiantPartenaire	10								x																																							
mdpPartenaire	11									x																																						
dateEnvoiPartenaire*	12									x																																						
repertoirePartenaire	13										x																																					
idPlateau	14										x	x	x																																			
nomPlateau	15										x																																					
idSecteur	16											x	x	x																																		
nomSecteur	17											x																																				
idDomaine	18											x	x	x																																		
nomDomaine	19											x																																				
idEditeur	20											x	x																																			
nomEditeur	21											x																																				
idProfil	22												x	x	x																																	
nomProfil	23												x																																			
codeProfil	24												x																																			
idImprimeur	25												x																																			
nomImprimeur	26												x																																			
serveurImprimeur	27												x																																			
identifiantImprimeur	28												x																																			
mdpImprimeur	29												x																																			
idTypeDepotImprim*	30												x																																			
nomTypeDepotImp*	31												x																																			
repertoireImprimeur	32												x																																			
idEpreuve	33												x																																			
nomEpreuve	34												x																																			
serveurEpreuve	35												x																																			
identifiantEpreuve	36												x																																			
mdpEpreuve	37												x																																			
repertoireEpreuve	38												x																																			
idCdf	39												x																																			
nomCdf	40												x																																			
dateCdf	41												x																																			
dateDepartImpress*	42												x																																			
dateMiseVente	43												x																																			
idTitreSecondaire	44												x																																			

5.1.4 Le graphe des dépendances fonctionnelles

J'ai ensuite modélisé le graphe des dépendances fonctionnelles qui me servira à créer mon diagramme de classes. J'ai coloré le sommet des arbres, qui représentent les futurs identifiants de classe, alors que les arbres eux-même représentent une classe.

J'ai utilisé quelques règles de transformation pour le passage de la matrice au graphe de dépendances fonctionnelles :

1. Un attribut source de dépendance fonctionnelle est placé en haut d'un arbre
 2. une dépendance fonctionnelle est représenté par une flèche
 3. On assure une couverture minimale en éliminant les éventuelles dépendances qui sont obtenues par transitivité
 4. Si il y a des attributs orphelins, il faut leur chercher une source multiple de dépendances fonctionnelles.



5.2 - LE DIAGRAMME DE CLASSES

Pour établir le **Modèle conceptuel**, j'ai choisi d'utiliser un **diagramme de classes entités** qui est un **modèle statique normalisé par UML** pour représenter les classes entités, leurs **associations** et **multiplicités**.

Une classe est une représentation **abstraite** d'un objet, réel ou virtuel, elle contient un nom, des attributs et des opérations.

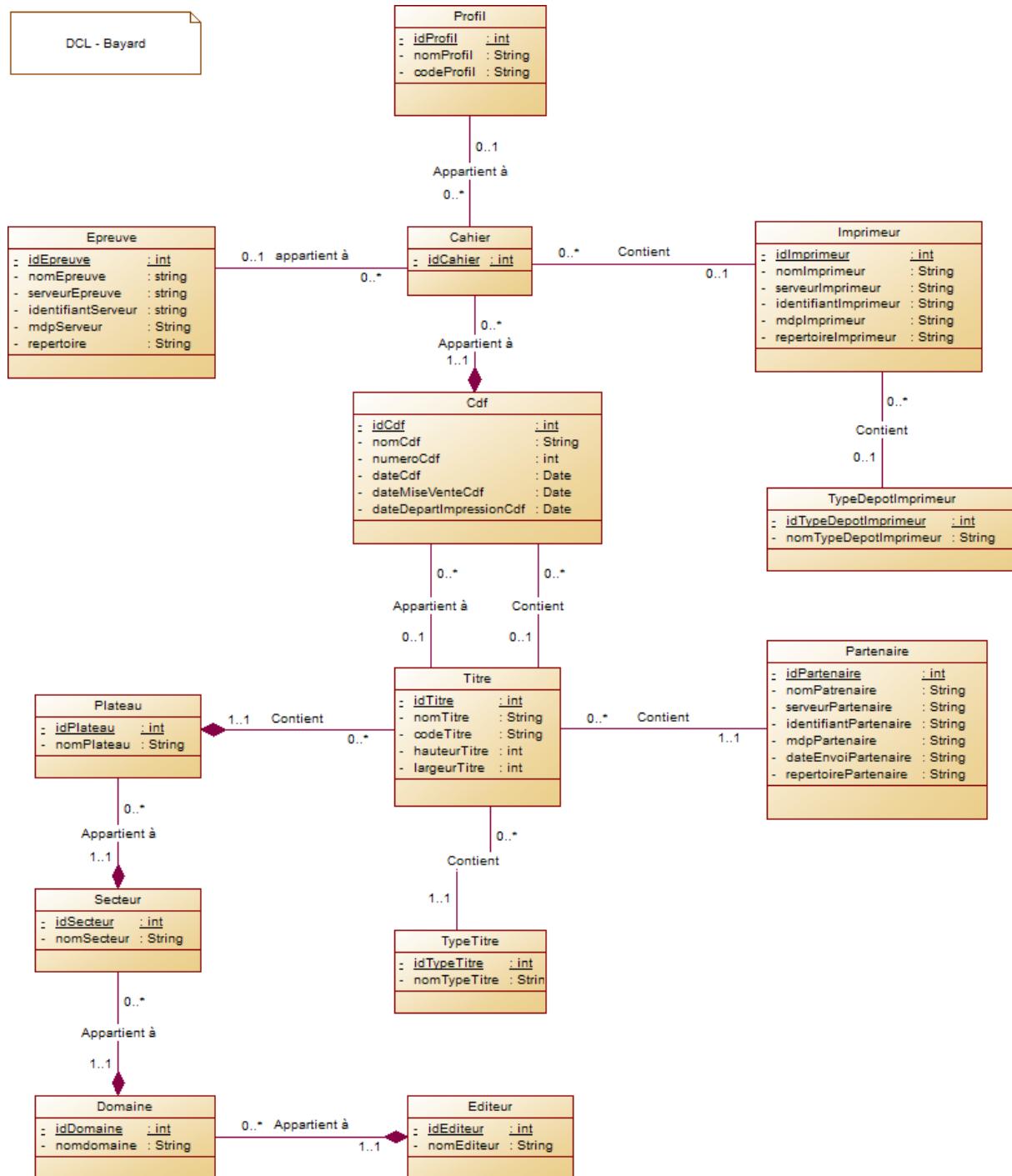
Les classes Entités, selon le modèle **Entity Control Boundary(ECB)** d'Ivar Jacobson, représentent le plus souvent les données qui seront persistantes.

5.2.1 Traduction du Graphe de dépendances fonctionnelles en diagramme de classe entité

Pour traduire le graphe des dépendances fonctionnelles en diagramme de classe, j'ai dû appliquer les règles suivantes :

- Un arbre représente une classe
- Le sommet d'un arbre est un attribut identifiant de la classe
- Une feuille terminale d'un arbre est un attribut non-identifiant de la classe
- Une dépendance fonctionnelle inter sommet est une association inter classe avec une multiplicité maxi de 1 sur l'un de ses cotés(Père-Fils)
- Un attribut avec plusieurs sommets représente une classe avec deux associations
- Dans mon projet je n'ai malheureusement pas d'association inter-classes avec une multiplicité de *.* , je ne vous représente donc pas de concaténation, qui représenterait une classe association, si celle-ci avait un attribut dépendant.

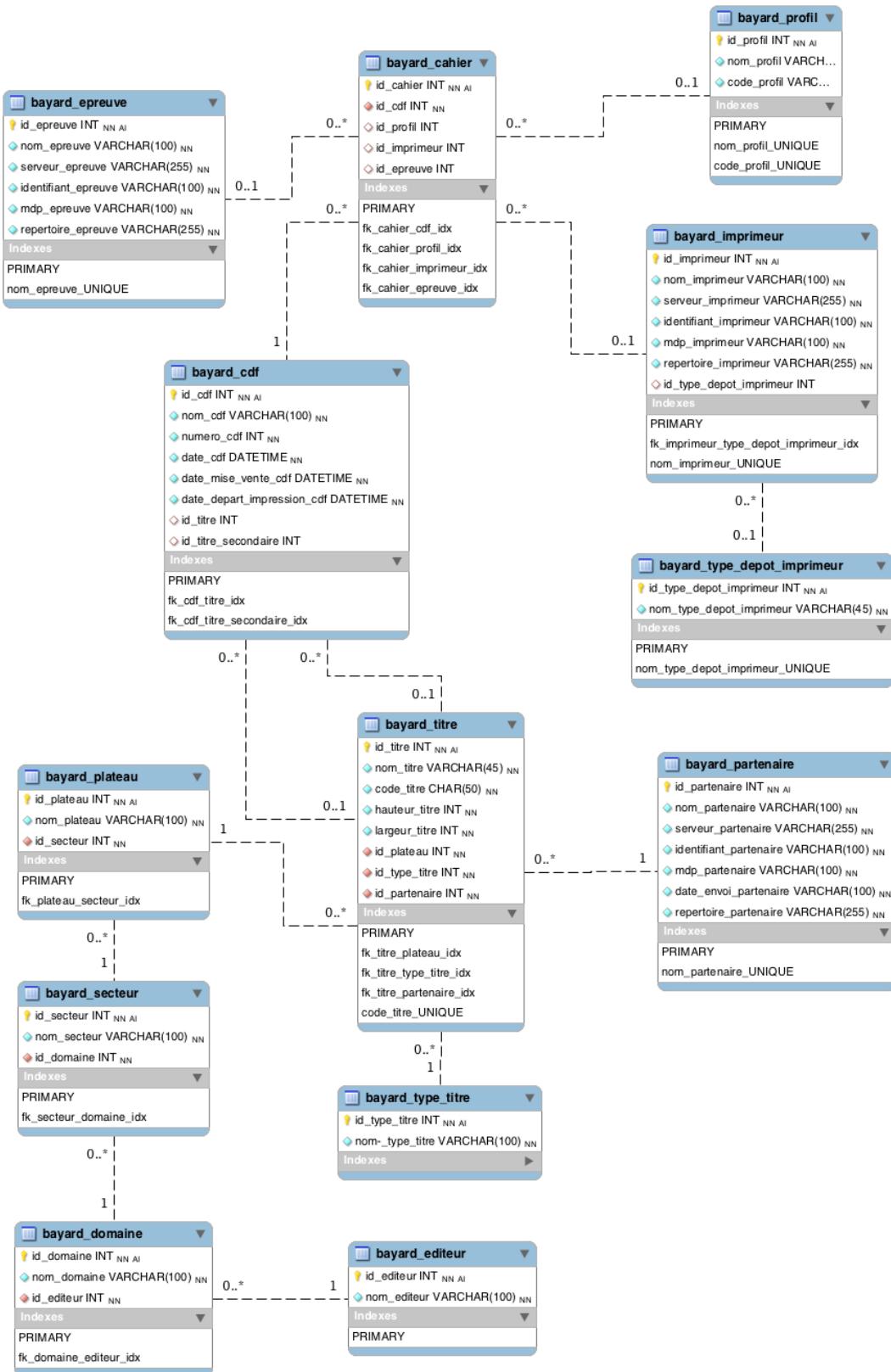
5.2.2 Le diagramme de classes



On notera les associations de composition représentées par un losange plein, La classe composite qui se trouve du côté du losange, contient le composant c'est une agrégation forte, la destruction du composite entraîne la destruction du composant. Nous pourrons ajouter des DELETE CASCADE lors de la création du Modèle physique de données.

5.3 - LE MODÈLE PHYSIQUE DE DONNÉES

Pour élaborer le modèle physique de données, qui représente le schéma de la base de données, j'ai utilisé MySQL WorkBench.



5.4 - SQL : LE LDD

Le Structured Query Language est un langage informatique normalisé permettant d'exploiter les bases de données relationnelles. Il se compose de 4 sous-langages dont le LDD(Langage de définition des données) qui sert principalement à manipuler la structure des données.

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----
-- Schema dev
--

CREATE SCHEMA IF NOT EXISTS `xplan` DEFAULT CHARACTER SET utf8 ;

USE `dev` ;

-- -----
-- Table `bayard_editeur`
--

DROP TABLE IF EXISTS `bayard_editeur` ;

CREATE TABLE IF NOT EXISTS `bayard_editeur` (
  `id_editeur` INT NOT NULL AUTO_INCREMENT,
  `nom_editeur` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`id_editeur`)
)
ENGINE = InnoDB;

-- -----
-- Table `bayard_domaine`
--

DROP TABLE IF EXISTS `bayard_domaine` ;

CREATE TABLE IF NOT EXISTS `bayard_domaine` (
  `id_domaine` INT NOT NULL AUTO_INCREMENT,
  `nom_domaine` VARCHAR(100) NOT NULL,
  `id_editeur` INT NOT NULL,
  PRIMARY KEY (`id_domaine`),
  INDEX `fk_domaine_editeur_idx` (`id_editeur` ASC),
  CONSTRAINT `fk_domaine_editeur`
    FOREIGN KEY (`id_editeur`)
    REFERENCES `bayard_editeur` (`id_editeur`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
```

```

ENGINE = InnoDB;

-- -----
-- Table `bayard_secteur`
-- -----
DROP TABLE IF EXISTS `bayard_secteur` ;

CREATE TABLE IF NOT EXISTS `bayard_secteur` (
  `id_secteur` INT NOT NULL AUTO_INCREMENT,
  `nom_secteur` VARCHAR(100) NOT NULL,
  `id_domaine` INT NOT NULL,
  PRIMARY KEY (`id_secteur`),
  INDEX `fk_secteur_domaine_idx` (`id_domaine` ASC),
  CONSTRAINT `fk_bayard_secteur_bayard_domaine1`
    FOREIGN KEY (`id_domaine`)
    REFERENCES `bayard_domaine` (`id_domaine`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `bayard_plateau`
-- -----
DROP TABLE IF EXISTS `bayard_plateau` ;

CREATE TABLE IF NOT EXISTS `bayard_plateau` (
  `id_plateau` INT NOT NULL AUTO_INCREMENT,
  `nom_plateau` VARCHAR(100) NOT NULL,
  `id_secteur` INT NOT NULL,
  PRIMARY KEY (`id_plateau`),
  INDEX `fk_plateau_secteur_idx` (`id_secteur` ASC),
  CONSTRAINT `fk_plateau_secteur`
    FOREIGN KEY (`id_secteur`)
    REFERENCES `bayard_secteur` (`id_secteur`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `bayard_type_titre`
-- -----
DROP TABLE IF EXISTS `bayard_type_titre` ;

CREATE TABLE IF NOT EXISTS `bayard_type_titre` (
  `id_type_titre` INT NOT NULL AUTO_INCREMENT,
  `nom_type_titre` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`id_type_titre`))
ENGINE = InnoDB;

```

```

-- -----
-- Table `bayard_partenaire`
-- -----
DROP TABLE IF EXISTS `bayard_partenaire` ;

CREATE TABLE IF NOT EXISTS `bayard_partenaire` (
  `id_partenaire` INT NOT NULL AUTO_INCREMENT,
  `nom_partenaire` VARCHAR(100) NOT NULL,
  `serveur_partenaire` VARCHAR(255) NOT NULL,
  `identifiant_partenaire` VARCHAR(100) NOT NULL,
  `mdp_partenaire` VARCHAR(100) NOT NULL,
  `date_envoi_partenaire` VARCHAR(100) NOT NULL,
  `repertoire_partenaire` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id_partenaire`),
  UNIQUE INDEX `nom_partenaire_UNIQUE` (`nom_partenaire` ASC)
ENGINE = InnoDB;

-- -----
-- Table `bayard_titre`
-- -----
DROP TABLE IF EXISTS `bayard_titre` ;

CREATE TABLE IF NOT EXISTS `bayard_titre` (
  `id_titre` INT NOT NULL AUTO_INCREMENT,
  `nom_titre` VARCHAR(45) NOT NULL,
  `code_titre` CHAR(50) NOT NULL,
  `hauteur_titre` INT NOT NULL,
  `largeur_titre` INT NOT NULL,
  `id_plateau` INT NOT NULL,
  `id_type_titre` INT NOT NULL,
  `id_partenaire` INT NOT NULL,
  PRIMARY KEY (`id_titre`),
  INDEX `fk_titre_plateau_idx` (`id_plateau` ASC),
  INDEX `fk_titre_type_titre_idx` (`id_type_titre` ASC),
  INDEX `fk_titre_partenaire_idx` (`id_partenaire` ASC),
  UNIQUE INDEX `code_titre_UNIQUE` (`code_titre` ASC),
  CONSTRAINT `fk_titre_plateau`
    FOREIGN KEY (`id_plateau`)
    REFERENCES `bayard_plateau` (`id_plateau`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_titre_type_titre`
    FOREIGN KEY (`id_type_titre`)
    REFERENCES `bayard_type_titre` (`id_type_titre`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_titre_partenaire`
    FOREIGN KEY (`id_partenaire`)
    REFERENCES `bayard_partenaire` (`id_partenaire`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)

```

```

ENGINE = InnoDB;

-- -----
-- Table `bayard_cdf`
-- -----
DROP TABLE IF EXISTS `bayard_cdf` ;

CREATE TABLE IF NOT EXISTS `bayard_cdf` (
  `id_cdf` INT NOT NULL AUTO_INCREMENT,
  `nom_cdf` VARCHAR(100) NOT NULL,
  `numero_cdf` INT NOT NULL,
  `date_cdf` DATETIME NOT NULL,
  `date_mise_vente_cdf` DATETIME NOT NULL,
  `date_depart_impression_cdf` DATETIME NOT NULL,
  `id_titre` INT NULL,
  `id_titre_secondaire` INT NULL,
  PRIMARY KEY (`id_cdf`),
  INDEX `fk_cdf_titre_idx` (`id_titre` ASC),
  INDEX `fk_cdf_titre_secondaire_idx` (`id_titre_secondaire` ASC),
  CONSTRAINT `fk_cdf_titre`
    FOREIGN KEY (`id_titre`)
    REFERENCES `bayard_titre` (`id_titre`)
    ON DELETE SET NULL
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_cdf_titre_secondaire`
    FOREIGN KEY (`id_titre_secondaire`)
    REFERENCES `bayard_titre` (`id_titre`)
    ON DELETE SET NULL
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `bayard_profil`
-- -----
DROP TABLE IF EXISTS `bayard_profil` ;

CREATE TABLE IF NOT EXISTS `bayard_profil` (
  `id_profil` INT NOT NULL AUTO_INCREMENT,
  `nom_profil` VARCHAR(100) NOT NULL,
  `code_profil` VARCHAR(200) NOT NULL,
  PRIMARY KEY (`id_profil`),
  UNIQUE INDEX `nom_profil_UNIQUE` (`nom_profil` ASC),
  UNIQUE INDEX `code_profil_UNIQUE` (`code_profil` ASC))
ENGINE = InnoDB;

-- -----
-- Table `bayard_type_depot_imprimeur`
-- -----
DROP TABLE IF EXISTS `bayard_type_depot_imprimeur` ;

CREATE TABLE IF NOT EXISTS `bayard_type_depot_imprimeur` (
  `id_type_depot_imprimeur` INT NOT NULL AUTO_INCREMENT,

```

```

`nom_type_depot_imprimeur` VARCHAR(45) NOT NULL,
PRIMARY KEY (`id_type_depot_imprimeur`),
UNIQUE INDEX `nom_type_depot_imprimeur_UNIQUE`(`nom_type_depot_imprimeur` ASC)
ENGINE = InnoDB;

-- -----
-- Table `bayard_imprimeur`
-- -----
DROP TABLE IF EXISTS `bayard_imprimeur` ;

CREATE TABLE IF NOT EXISTS `bayard_imprimeur` (
  `id_imprimeur` INT NOT NULL AUTO_INCREMENT,
  `nom_imprimeur` VARCHAR(100) NOT NULL,
  `serveur_imprimeur` VARCHAR(255) NOT NULL,
  `identifiant_imprimeur` VARCHAR(100) NOT NULL,
  `mdp_imprimeur` VARCHAR(100) NOT NULL,
  `repertoire_imprimeur` VARCHAR(255) NOT NULL,
  `id_type_depot_imprimeur` INT NULL,
  PRIMARY KEY (`id_imprimeur`),
  INDEX `fk_imprimeur_type_depot_imprimeur_idx`(`id_type_depot_imprimeur` ASC),
  UNIQUE INDEX `nom_imprimeur_UNIQUE`(`nom_imprimeur` ASC),
  CONSTRAINT `fk_imprimeur_type_depot_imprimeur`
    FOREIGN KEY (`id_type_depot_imprimeur`)
    REFERENCES `bayard_type_depot_imprimeur`(`id_type_depot_imprimeur`)
    ON DELETE SET NULL
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `bayard_epreuve`
-- -----
DROP TABLE IF EXISTS `bayard_epreuve` ;

CREATE TABLE IF NOT EXISTS `bayard_epreuve` (
  `id_epreuve` INT NOT NULL AUTO_INCREMENT,
  `nom_epreuve` VARCHAR(100) NOT NULL,
  `serveur_epreuve` VARCHAR(255) NOT NULL,
  `identifiant_epreuve` VARCHAR(100) NOT NULL,
  `mdp_epreuve` VARCHAR(100) NOT NULL,
  `repertoire_epreuve` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id_epreuve`),
  UNIQUE INDEX `nom_epreuve_UNIQUE`(`nom_epreuve` ASC))
ENGINE = InnoDB;

```

```
-- -----
-- Table `bayard_cahier`
-- -----
DROP TABLE IF EXISTS `bayard_cahier` ;

CREATE TABLE IF NOT EXISTS `bayard_cahier` (
  `id_cahier` INT NOT NULL AUTO_INCREMENT,
  `id_cdf` INT NOT NULL,
  `id_profil` INT NULL,
  `id_imprimeur` INT NULL,
  `id_epreuve` INT NULL,
  PRIMARY KEY (`id_cahier`),
  INDEX `fk_cahier_cdf_idx` (`id_cdf` ASC),
  INDEX `fk_cahier_profil_idx` (`id_profil` ASC),
  INDEX `fk_cahier_imprimeur_idx` (`id_imprimeur` ASC),
  INDEX `fk_cahier_epreuve_idx` (`id_epreuve` ASC),
  CONSTRAINT `fk_cahier_cdf`
    FOREIGN KEY (`id_cdf`)
    REFERENCES `bayard_cdf` (`id_cdf`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_bayard_cahier_profil`
    FOREIGN KEY (`id_profil`)
    REFERENCES `bayard_profil` (`id_profil`)
    ON DELETE SET NULL
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_cahier_imprimeur`
    FOREIGN KEY (`id_imprimeur`)
    REFERENCES `bayard_imprimeur` (`id_imprimeur`)
    ON DELETE SET NULL
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_cahier_epreuve`
    FOREIGN KEY (`id_epreuve`)
    REFERENCES `bayard_epreuve` (`id_epreuve`)
    ON DELETE SET NULL
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

CHAPITRE 6 - CONCEPTION DE L'APPLICATION

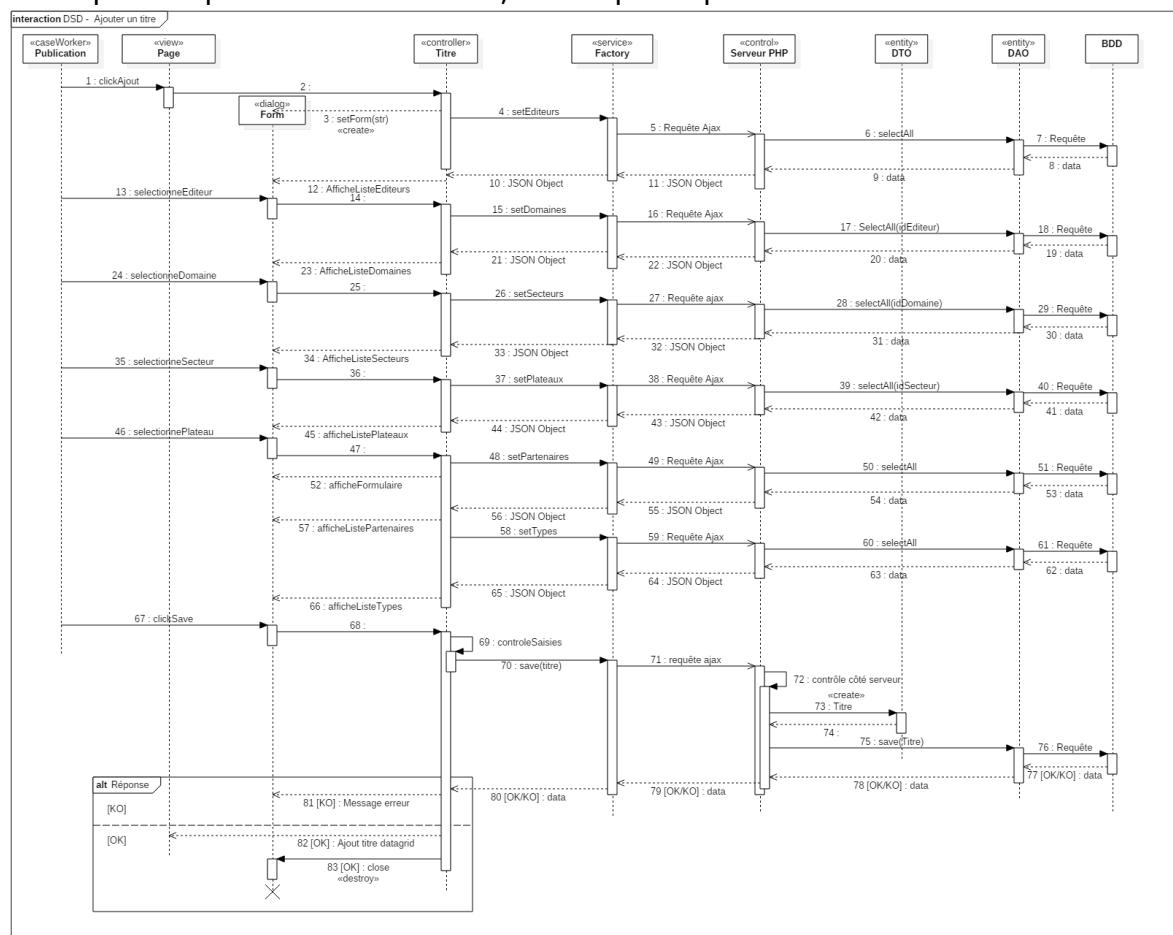
6.1 - LE DIAGRAMME DE SÉQUENCE DÉTAILLÉ (AJOUTER UN TITRE)

Voici le diagramme de séquence détaillé pour l'ajout d'un titre, ce diagramme intervient durant la phase de conception, juste avant de commencer à coder l'application, il formalise les actions utilisateurs et les interactions entre les différentes couches du système, par l'intermédiaire de stimulus dont les noms correspondent aux message des langages informatique.

On peut noter sur ce diagramme, qu'il y a plusieurs stéréotypes, notamment :

- **view**, qui correspond à la couche statique de l'écran, celle qui sera visible directement par l'utilisateur dans son navigateur
- **dialog** qui est là pour représenter une boîte de dialogue
- **controller** qui correspond dans notre cas à la couche dynamique de l'écran, notamment le contrôleur AngularJS
- **service**, qui est utilisé dans le cadre d'angular correspondrait ici à un data access object, il permet ici de communiquer avec le serveur.
- **control** qui se trouve sur le serveur qui récupèrera les requêtes et renverra une réponse
- **entity**, qui représente le Data transfert object(DTO) et le Data access object(DAO).

Le temps est représenté verticalement, tandis que l'espace horizontalement.



6.1.1 A propos du diagramme de séquence détaillé

Comme nous avons pu le constater, j'utilise une partie du pattern **ECB(Entities Controls Boundaries)** sur le serveur, à savoir la couche **controls** et la couche **entities**.

Tandis qu'en front-end, j'utilise un **MVC(Model-view-controller)** et plus particulièrement un **MVVM(Model-View View-Model)**.

Le **MVVM** est globalement comme un MVC, sauf que la communication entre le modèle et la vue est bidirectionnel.

Avec AngularJS, à chaque fois qu'un contrôleur est instancié, un objet **\$scope** est créé, qui sera lié à des éléments de la vue et la mettra à jour.

J'ai volontairement « simplifié » ce diagramme de séquence détaillé, j'aurai pu ajouter des controllers, services, controls, et entities pour chaque requête effectuée pour obtenir autre chose qu'un objet titre(par ex : Editeur, Domaine, Secteur, Plateau, etc..) , mais cela aurait été très difficile à lire.

CHAPITRE 7 - DÉVELOPPEMENT

7.1 - TECHNOLOGIES UTILISÉES

Comme nous l'avions vu précédemment, j'ai utilisé :

- AngularJS 1.5.7 pour le MVVM côté client.
- PHP 5.6 côté serveur.
- HTML 5, CSS3 pour la partie statique côté client.
- J'ai également utilisé la librairie javascript linq.js qui apporte une façon élégante(je trouve) de faire des boucles en une seule ligne.
- MySQL 5.6 pour le système de gestion de base de données relationnelle(SGBR)

Sur certains des codes que vous trouverez dans les pages suivantes, notamment la page index, il apparaît du **twig**, qui est un langage de **template**. Nous venons juste d'intégrer cette partie du projet au nouveau cœur développé sur le framework Codeigniter, et nous avons malheureusement été obligés de convertir cette page qui était initialement une simple page Html pour pouvoir profiter du système de routing du framework, et par la même occasion, de ses droits d'accès.

7.2 - COPIES D'ÉCRANS

7.2.1 Les titres

EDITEUR	DOmaine	SECTEUR	PLATEAU	NOM	CODE	ACTION	
						Modifier	Supprimer
Bayard	Presse	jeunesse	Langues	J'aime Lire	JLN	Modifier	Supprimer
Bayard	Presse	jeunesse	Lecture	J'aime Lire Hors Série	JLH	Modifier	Supprimer
Bayard	Presse	jeunesse	Lecture	Je Bouquine	JBQN	Modifier	Supprimer
Bayard	Presse	jeunesse	Lecture	Je Bouquine Hors Série	JBQH	Modifier	Supprimer
Bayard	Presse	jeunesse	Lecture	Je bouquine Supplément	JBQS	Modifier	Supprimer

Début | Précédent | 1 | Suivant | Fin

7.2.2 La recherche d'un titre

EDITEUR	DOmaine	SECTEUR	PLATEAU	NOM	CODE	ACTION	
						Modifier	Supprimer
Bayard	Presse	jeunesse	Langues	J'aime Lire	JLN	Modifier	Supprimer
Bayard	Presse	jeunesse	Lecture	J'aime Lire Hors Série	JLH	Modifier	Supprimer

Début | Précédent | 1 | Suivant | Fin

Tout afficher | x

7.2.3 Modification d'un titre

The screenshot shows a software application window titled "Adict". The main area displays a list of titles categorized by editor (Bayard), domain (Presse), and sector (Jeunesse). One title, "J'aime lire", is selected. A modal dialog box titled "Edition d'un titre" is open, allowing modification of various parameters such as Editor, Domaine, Sector, Plateau, Title, and Type. At the bottom of the dialog box are "Annuler" and "Sauvegarder" buttons.

7.2.4 L'ajout d'un titre – étape 1

The screenshot shows a software application window titled "Adict". The main area displays a list of titles categorized by editor (Bayard), domain (Presse), and sector (Jeunesse). A new title, "Edition", is being added. A modal dialog box titled "Edition d'un titre" is open, showing the "Title" field populated with "Edition". Other fields like Editor, Domaine, Sector, Plateau, Profile, Partner, Printer, Proof, and Type are also visible. At the bottom of the dialog box are "Annuler" and "Sauvegarder" buttons.

7.2.5 Ajout d'un titre - Etape 2

The screenshot shows a web-based application interface for managing titles. At the top, there's a navigation bar with tabs like 'Editeur', 'Domaine', 'Secteur', etc. Below the navigation, a table lists several entries under the 'BAYARD' category. One entry is selected, and a modal dialog box titled 'Edition d'un titre' is open over it. This dialog box contains input fields for 'Titre' (set to 'Choisir un Titre'), 'Numéro', 'Date de parution', 'Date de mise en vente', 'Date départ impression', 'Titre secondaire' (set to 'Choisir un Titre secondaire'), and other publishing details like 'Dossier de destination' (set to 'Bayard'), 'Label', 'Prefix', 'Folio de départ', 'Folio de fin', 'Bouclage', 'Rubrique', and 'Rubrique 1'. At the bottom of the dialog are 'Annuler' and 'Sauvegarder' buttons.

7.2.6 Front-office - Créer un chemin de fer

This screenshot shows the 'Publication' section of the Adict application. On the left, there's a sidebar with 'Dossiers' and 'PRODUCTION' sections containing 'Bayard' and 'test' items. The main area displays a table with columns: LABEL, PRÉFIXE, PAGES, BOUCLAGE, DATE DE CRÉATION, and CHEMIN. A single row is listed with the label 'Aucun chemin de fer'. A modal dialog box titled 'Créer un nouveau chemin de fer' is open over this row. This dialog box contains fields for 'Titre' (set to 'Choisir un Titre'), 'Numéro', 'Date de parution', 'Date de mise en vente', 'Date départ impression', 'Titre secondaire' (set to 'Choisir un Titre secondaire'), and other publishing details like 'Dossier de destination' (set to 'Bayard'), 'Label', 'Prefix', 'Folio de départ', 'Folio de fin', 'Bouclage', 'Rubrique', and 'Rubrique 1'. At the bottom of the dialog are 'ANNULER' and 'Créer' buttons.

7.2.7 Front-office – Créer un chemin de fer(suite)

7.2.8 Front-office – Modifier un chemin de fer

7.3 ARBORESCENCE DU PROJET

Pour séparer les différentes couches de l'application, j'ai créé une arborescence :

Le dossier API qui contiendra tous les scripts interprétés côté serveur :

- **CONTROLS** : les contrôleurs
- **DAO**: Les Data Access Objects
- **Entities** : Les Data Transfert Objects
- **UTILS** : Les utilitaires (connexion ...)

Le dossier **Views** qui contiendra principalement les **partials** qui serviront au **routing** d'AngularJS.

Et , le dossier js, qui contiendra tous les scripts voués à être interprétés sur le navigateur côté client :

- **controller** : les contrôleurs
- **directive** : les directives
- **library** : les scripts autre(linq.js, angular.js ...)
- **service** : Les factory
- **app.js** qui est le script qui définit le **module** angularJS et ses routes

A la racine, on trouve également **index.twig**, qui est le point d'entrée de l'application Single Page



7.4 - CODES STATIQUES DES ÉCRANS

Les codes statiques des écrans sont, dans le cas de notre application, développés en HTML 5 avec CSS 3.0, je n'ai pas eu besoin d'ajouter mes propres fichiers CSS, nous avons utilisé ceux déjà présents au sein d'adiict. Notamment, un bootstrap modifié par le web designer de l'équipe.

7.4.1 index.html.twig

La page index.html, a donc été convertie en twig tardivement, WEBROOT est une variable javascript globale accessible dans adiict.

On notera la présence de **ng-app**, qui permet de préciser à AngularJS, dans quelle balise sera la portée de l'application. Egalement, la balise **ng-view**, dans laquelle apparaîtront des **inclusions** de fichiers html grâce à AngularJS.

```
<div ng-app="ngApp">

    <nav class="text text-center" >
        <ul >
            <li style="display: inline-block"><a class="btn btn-active " href="#/editeur">Editeur</a></li>
            <li style="display: inline-block"><a class="btn btn-active " href="#/domaine">Domaine</a></li>
            <li style="display: inline-block"><a class="btn btn-active " href="#/secteur">Secteur</a></li>
            <li style="display: inline-block"><a class="btn btn-active " href="#/plateau">Plateau</a></li>
                <li style="display: inline-block"><a class="btn btn-active " href="#/titre">Titre</a></li>
                <li style="display: inline-block"><a class="btn btn-active " href="#/profil">Profil</a></li>
            <li style="display: inline-block"><a class="btn btn-active " href="#/partenaire">Partenaire</a></li>
                <li style="display: inline-block"><a class="btn btn-active " href="#/imprimeur">Imprimeur</a></li>
                <li style="display: inline-block"><a class="btn btn-active " href="#/epreuve">Epreuve</a></li>
                    <li style="display: inline-block"><a class="btn btn-active " href="#/type">Type</a></li>
            <li style="display: inline-block"><a class="btn btn-active " href="#/type">Type</a></li>
        </ul>
    </nav>

    <div ng-view></div>

    {#      LIBRARY  #-}
    <script src="{{ WEBROOT }}asset/lib/angular.js"></script>
    <script src="{{ WEBROOT }}asset/lib/angular-route.1.5.7.js"></script>
    <script src="{{ WEBROOT }}asset/lib/ling.hack.min.js"></script>

    {#      APP   #-}
    <script src="{{ WEBROOT }}asset/plug/bayard/js/app.js"></script>

    {#      SERVICES  #-}
    <script src="{{ WEBROOT }}asset/plug/bayard/js/service/FormService.js"></script>
    <script src="{{ WEBROOT }}asset/plug/bayard/js/service/countService.js"></script>
    <script src="{{ WEBROOT }}asset/plug/bayard/js/service/PaginatorService.js"></script>
    <script src="{{ WEBROOT }}asset/plug/bayard/js/service/typeService.js"></script>
    <script src="{{ WEBROOT }}asset/plug/bayard/js/service/editeurService.js"></script>
    <script src="{{ WEBROOT }}asset/plug/bayard/js/service/domaineService.js"></script>
    <script src="{{ WEBROOT }}asset/plug/bayard/js/service/secteurService.js"></script>
    <script src="{{ WEBROOT }}asset/plug/bayard/js/service/plateauService.js"></script>
    <script src="{{ WEBROOT }}asset/plug/bayard/js/service/imprimeurService.js"></script>
    <script src="{{ WEBROOT }}asset/plug/bayard/js/service/epreuveService.js"></script>
    <script src="{{ WEBROOT }}asset/plug/bayard/js/service/partenaireService.js"></script>
    <script src="{{ WEBROOT }}asset/plug/bayard/js/service/profilService.js"></script>
    <script src="{{ WEBROOT }}asset/plug/bayard/js/service/titreService.js"></script>
```

```

{#      CONTROLLERS  #}
<script src="{{ WEBROOT }}asset/plug/bayard/js/controller/editeurController.js"></script>
<script src="{{ WEBROOT }}asset/plug/bayard/js/controller/domaineController.js"></script>
<script src="{{ WEBROOT }}asset/plug/bayard/js/controller/secteurController.js"></script>
<script src="{{ WEBROOT }}asset/plug/bayard/js/controller/plateauController.js"></script>
<script src="{{ WEBROOT }}asset/plug/bayard/js/controller/imprimeurController.js"></script>
<script src="{{ WEBROOT }}asset/plug/bayard/js/controller/epreuveController.js"></script>
<script src="{{ WEBROOT }}asset/plug/bayard/js/controller/partenaireController.js"></script>
<script src="{{ WEBROOT }}asset/plug/bayard/js/controller/profilController.js"></script>
<script src="{{ WEBROOT }}asset/plug/bayard/js/controller/titreController.js"></script>
<script src="{{ WEBROOT }}asset/plug/bayard/js/controller/typeController.js"></script>

{#      DIRECTIVES  #}
<script src="{{ WEBROOT }}asset/plug/bayard/js/directive/pagination.js"></script>
<script src="{{ WEBROOT }}asset/plug/bayard/js/directive/recherche.js"></script>
<script src="{{ WEBROOT }}asset/plug/bayard/js/directive/buttons.js"></script>
<script src="{{ WEBROOT }}asset/plug/bayard/js/directive/test.js"></script>
</div>

```

7.4.2 titre.html

Ce fichier correspond à la page titre qui sera incluse à l'index, on peut constater la balise **ng-controller** qui précise la portée d'un contrôleur. On aperçoit une **directive <recherche></recherche>**. Une directive permet d'exporter un bloc de code répétitif dans une fichier externe, et de définir un élément html ou un attribut. **Ng-click** permet d'appeler une méthode du contrôleur. **Ng-bind** et **ng-model** permettent tous les deux de faire du **data-binding**, le premier de façon **unidirectionnelle**(Model->View) tandis que le second est **bidirectionnel**.

```

<div ng-controller="titreController">

<recherche></recherche>





```

```

</div>
<div class="form-group" ng-show="current.domaine !== null && current.editeur !== null">
  <label>Secteur :</label>

  <select class="form-control" ng-options="secteur as secteur.label for secteur in secteurs track by secteur.id" ng-model="current.secteur" ng-change="setSecteur(current.secteur)"></select>

</div>
<div class="form-group" ng-show="current.secteur !== null && current.editeur !== null && current.domaine !== null && current.secteur !== null">
  <label>Plateau :</label>

  <select class="form-control" ng-options="plateau as plateau.label for plateau in plateaux track by plateau.id" ng-model="current.plateau" ng-change="setPlateau(current.plateau)"></select>

</div>
<div ng-show="current.plateau !== null && current.editeur !== null && current.domaine !== null && current.plateau !== null && current.secteur !== null">
  <div class="form-group">
    <label>Titre :</label>
    <input class="form-control" type="text" value="" ng-model="current.label">
  </div>
  <div class="form-group">
    <label>Code :</label>
    <input class="form-control" type="text" value="" ng-model="current.key">
  </div>
  <div class="form-group">
    <label>Hauteur :</label>
    <input class="form-control" type="text" value="" ng-model="current.hauteur">
  </div>
  <div class="form-group">
    <label>Largeur :</label>
    <input class="form-control" type="text" value="" ng-model="current.largeur">
  </div>
  <div class="form-group">

    <label>Types :</label>

    <select class="form-control" ng-options="t as t.label for t in types track by t.id" ng-model="current.type"></select>
  </div>
  <div class="form-group">

    <label>Partenaires :</label>

    <select class="form-control" ng-options="part as part.label for part in partenaires track by part.id" ng-model="current.partenaire"></select>
  </div>
  <buttons></buttons>

</div>
</div>
<pagination></pagination>
</div>

```

7.4.3 buttons.html

C'est un exemple de code utilisé par une directive.

```

<div class="form-group col col-12">
  <button ng-click="cancel()" class="btn btn-default">Annuler</button>
  <button ng-click="save()" class="btn btn-green">Sauvegarder</button>
</div>

```

7.4.4 recherche.html

```

<div class="form-inline text-center float-left">
  <div class="form-group " style="width: 100%">

    <label for="search">Recherche :</label>
    <input class="form-control" type="text" name="search" value="" ng-model="search" ng-keyup="fire($event)"/>
    <button ng-click="refresh()" class="btn btn-default"><i class="fa fa-search"></i></button>

  </div>
</div>
<button ng-click="add()" class="btn btn-green pull-right"><i class="fa fa-plus"></i> Nouveau</button>

```

7.4.5 pagination.html

```
<!--PAGINATION-->
<div class="text text-center" style="position: absolute; bottom: 30px ; width: 100%">
  <a href="" ng-click="debut()" class="btn btn-grey">Début</a>
  <a href="" ng-click="precedent()" class="btn btn-grey">Précédent</a>
  <a href="" ng-click="setPage(t)" ng-repeat="t in pages track by $index" class="btn btn-grey">{{t}}</a>
  <a href="" ng-click="suivant()" class="btn btn-grey">Suivant</a>
  <a href="" ng-click="fin()" class="btn btn-grey">Fin</a>
</div>
```

7.5 - CODES DYNAMIQUES DES ÉCRANS

Ce sont les scripts javascript. Ceux-ci sont interprétés par le navigateur du client, ce qui permet d'apporter de l'interactivité et de la réactivité à notre application. On parle d'**application riche**. L'avantage d'une application riche, par rapport à une application lourde, c'est qu'il n'y a pas besoin de déployer l'application sur n-postes.

7.5.1 app.js

Le fichier app.js est celui qui définit l'application, son point d'entrée côté navigateur, ici nous lui **injectons, une dépendance**, qui est le système de **routing** d'angularJS.

Ensuite, lors de la configuration du module, nous lui précisons les routes, qui serviront à inclure les « **partials** », les fichiers qui apparaîtront dans le **ng-view**.

```
//Déclaration du module et injection des dépendances angular
var app = angular.module('ngApp', [ 'ngRoute']);

//Configuration des routes du module
app.config(function ($routeProvider) {
  $routeProvider
    .when('/', {templateUrl: App.WEBROOT + 'asset/plug/bayard/Boundaries/home.html'})
    .when('/editeur', {templateUrl: App.WEBROOT + 'asset/plug/bayard/Views/partials/editeur.html'})
    .when('/domaine', {templateUrl: App.WEBROOT + 'asset/plug/bayard/Views/partials/domaine.html'})
    .when('/secteur', {templateUrl: App.WEBROOT + 'asset/plug/bayard/Views/partials/secteur.html'})
    .when('/plateau', {templateUrl: App.WEBROOT + 'asset/plug/bayard/Views/partials/plateau.html'})
    .when('/titre', {templateUrl: App.WEBROOT + 'asset/plug/bayard/Views/partials/titre.html'})
    .when('/profil', {templateUrl: App.WEBROOT + 'asset/plug/bayard/Views/partials/profil.html'})
    .when('/partenaire', {templateUrl: App.WEBROOT + 'asset/plug/bayard/Views/partials/partenaire.html'})
    .when('/imprimeur', {templateUrl: App.WEBROOT + 'asset/plug/bayard/Views/partials/imprimeur.html'})
    .when('/epreuve', {templateUrl: App.WEBROOT + 'asset/plug/bayard/Views/partials/epreuve.html'})
    .when('/type', {templateUrl: App.WEBROOT + 'asset/plug/bayard/Views/partials/type.html'})

    .otherwise({redirectTo: '/'});
});
```

7.5.2 titreController.js

Dans un autre fichier, nous déclarons un **contrôleur**, ici titreController. Celui-ci nous permettra de gérer la vue partielle titre.html. On voit, l'injection des différentes dépendances, notamment les **services(Factory)** et le **\$scope**.

Le \$scope est un élément très important, c'est la couche **Model** de notre MVVM. Pour chaque controller, un \$scope est automatiquement créé par le framework. D'après les benchmark, on peut utiliser jusqu'à 200 \$scopes dans une **Single Page Application(SPA)**, ce qui laisse la place à de jolies perspectives.

```
//déclaration du controller et injection des dépendances
app.controller('titreController', function ($scope, $filter, titreFactory, secteurFactory, editeurFactory,
domaineFactory, plateauFactory, partenaireFactory, typeFactory, PaginatorService, FormFactory) {
  FormFactory.reset();

  //déclaration des attributs du $scope
  $scope.show = false;
  $scope.showDelete = false;
  $scope.showSecteur = false;
  $scope.showDomaine = false;
  $scope.secteurSelected = null;
  $scope.searchTitre = null;
  $scope.resultatTitre = null;
  $scope.partenaires = {};
```

```

$scope.types = {};
$scope.pages = [];

$scope.current = {};
$scope.search = '';

$scope.orderBy = 'label';
$scope.sens = "ASC";

$scope.setOrder = function (str) {
    $scope.sens = $scope.sens === "ASC" ? "DESC" : "ASC";
    $scope.orderBy = str + " " + $scope.sens;
    $scope.refresh();
};

//gestion de la touche <enter> sur le champs recherche
$scope.fire = function (event) {
    if (event.keyCode === 13) {
        $scope.refresh();
    }
};

//maj de la pagination
PaginatorService.refresh = function () {
    $scope.refresh();
};

//initialisation de la pagination
PaginatorService.init("bayard_titre");

//maj tableaux des titres
$scope.refresh = function () {

    $scope.titres = titreFactory.getTitres(PaginatorService.getDebut(), PaginatorService.getFin(),
    $scope.search, 0, $scope.orderBy).then(function (data) {
        $scope.pages = PaginatorService.addPaginator(PaginatorService.currentPage);
        $scope.titres = data;
    });
};

//pagination: précédent
$scope.precedent = function () {
    PaginatorService.precedent();
    $scope.refresh();
};

//pagination : suivant
$scope.suivant = function () {
    PaginatorService.suivant();
    $scope.refresh();
};

//pagination: première page
$scope.debut = function () {

    PaginatorService.debut();
    $scope.refresh();
};

//pagination : dernière page
$scope.fin = function () {

    PaginatorService.fin();
    $scope.refresh();
};

//pagination : choix d'une page
$scope.setPage = function (index) {
    PaginatorService.currentPage = index;
    $scope.refresh();
};

//recherche d'un titre
$scope.rechercherTitre = function () {

    $scope.resultatTitre = titreFactory.getTitres(1000, 0, $scope.searchTitre, 0, 't.label').then(function
    (data) {
        $scope.resultatTitre = data;
    });
};

```

```

//listes des partenaires
$scope.partenaires = partenaireFactory.getPartenaires(1000, 0, '', 'label').then(function (data) {
  $scope.partenaires = data;
});

//liste des éditeurs
$scope.editeurs = editeurFactory.getEditeurs(1000, 0, '', 'label').then(function (data) {
  $scope.editeurs = data;
});

//liste des types
$scope.types = typeFactory.getTypes(1000, 0, '', 'label').then(function (data) {
  $scope.types = data;
});

//selection d'un éditeur dans la liste déroulante
$scope.setEditeur = function (editeur) {

  $scope.domaines = [];
  $scope.current.editeur = editeur;
  $scope.current.domaine = null;
  $scope.current.secteur = null;
  $scope.loadDomaine(editeur.id);

};

//sélection d'un domaine dans la liste déroulante
$scope.setDomaine = function (domaine) {

  $scope.secteurs = [];
  if (domaine === null)
    return;
  $scope.current.label_secteur = null;
  $scope.current.secteur = null;
  $scope.current.domaine = domaine;
  $scope.loadSecteur(domaine.id);
};

//récupération des domaines en fonction de l' id d'un éditeur
$scope.loadDomaine = function (id) {

  $scope.domaines = domaineFactory.getDomaines(1000, 0, '', id, 'd.label').then(function (data) {
    $scope.domaines = data;
    var d = Enumerable.From($scope.domaines).FirstOrDefault(null, "$.id == " + $scope.current.id_domaine);
    $scope.setDomaine(d);
  });

};

//Sélection d'un secteur dans la liste déroulante
$scope.setSecteur = function (secteur) {
  if (secteur === null)
    return;

  $scope.current.plateau = null;
  $scope.current.secteur = secteur;
  $scope.loadPlateau(secteur.id);
};

//récupération des secteurs en fonction de l' id d'un domaine
$scope.loadSecteur = function (id) {

  $scope.secteurs = secteurFactory.getSecteurs(1000, 0, '', id, 's.label').then(function (data) {
    $scope.secteurs = data;
    var s = Enumerable.From($scope.secteurs).FirstOrDefault(null, "$.id == " + $scope.current.id_secteur);
    $scope.setSecteur(s);
  });

};

//sélection d'un plateau dans la liste déroulante
$scope.setPlateau = function (plateau) {
  if (plateau === null)
    return;

  $scope.current.plateau = plateau;
};

//récupération des plateaux en fonction de l' id d'un secteur

```

```

$scope.loadPlateau = function (id) {
    $scope.plateaux = plateauFactory.getPlateaux(1000, 0, '', id, 'p.label').then(function (data) {
        $scope.plateaux = data;
        var s = Enumerable.From($scope.plateaux).FirstOrDefault(null, "$.id == " + $scope.current.id_plateau);
        $scope.setPlateau(s);
    });
};

//recherche d'un titre
$scope.searchTitreBy = function (str) {
    $scope.resultatTitre = titreFactory.getTitre(1000, 0, str, 0, 't.label').then(function (data) {
        $scope.resultatTitre = data;
    });
};

//affiche boite de dialogue de modification d'un titre
$scope.getDetails = function (id) {
    $scope.showDelete = true;
    $scope.show = true;

    var titre = titreFactory.getTitre(id).then(function (data) {

        $scope.current = Object.assign({}, data);
        $scope.current.type = Enumerable.From($scope.types).FirstOrDefault(null, "$.id == " +
        $scope.current.id_type);
        $scope.current.partenaire = Enumerable.From($scope.partenaires).FirstOrDefault(null, "$.id == " +
        $scope.current.id_partenaire);

        if ($scope.current === null)
            return;
        $scope.current.editeur = Enumerable.From($scope.editeurs).FirstOrDefault(null, "$.id == " +
        $scope.current.id_editeur);
        $scope.loadDomaine($scope.current.editeur.id);
        FormFactory.open("Edition d'un titre");
    });
};

//annulation de modification ou d'ajout d'un titre
$scope.cancel = function () {
    $scope.show = false;
    $scope.current = {};
    FormFactory.close();
};

//Affiche boite de dialogue d'ajout d'un titre
$scope.add = function () {

    $scope.showDelete = false;
    $scope.show = true;
    $scope.current = {
        id: 0,
        label: "",
        id_secteur: 0,
        id_editeur: 0,
        editeur: null,
        domaine: null,
        secteur: null,
        plateau: null,
        label_editeur: "",
        label_secteur: "",
        id_type: null,
        id_partenaire: null
    };
    $scope.showSecteur = false;
    //ouverture de la boite de dialogue
    FormFactory.open("Ajout d'un titre");
};

//Efface un titre
$scope.delete = function (titre)
{
    titreFactory.delete(titre).then(function (data) {
        if (data > 0) {

            for (var i = 0; i < $scope.titres.length; i++) {
                if (titre.id != $scope.titres[i].id)
                    continue;
        }
    });
};

```

```

        $scope.titres.splice(i, 1);
        $scope.show = false;
        break;
    }
}

});

//persiste les modifications ou l'ajout d'un titre
$scope.save = function () {

    $scope.current.id_editeur = $scope.current.editeur.id;
    $scope.current.id_domaine = $scope.current.domaine.id;
    $scope.current.id_secteur = $scope.current.secteur.id;
    $scope.current.id_plateau = $scope.current.plateau.id;
    $scope.current.label_editeur = $scope.current.editeur.label;
    $scope.current.label_domaine = $scope.current.domaine.label;
    $scope.current.label_secteur = $scope.current.secteur.label;
    $scope.current.label_plateau = $scope.current.plateau.label;
    $scope.current.id_type = $scope.current.type !== null ? $scope.current.type.id : null;
    $scope.current.id_partenaire = $scope.current.partenaire !== null ? $scope.current.partenaire.id : null;

    titreFactory.save($scope.current).then(function (data) {

        // SI SUCCESS
        if (data > 0) {

            FormFactory.close();
            $scope.show = false;
            //SI INSERT

            if ($scope.current.id == 0 && $scope.current.id_secteur > 0) {
                $scope.current.id = data;

                $scope.titres.push(Object.assign({}, $scope.current));
                $scope.current = {};
                return;
            }
            // SI UPDATE

            for (var i = 0; i < $scope.titres.length; i++) {
                if ($scope.titres[i].id != $scope.current.id)
                    continue;
                $scope.titres[i] = Object.assign({}, $scope.current);
                $scope.current = {};

                break;
            }
        }
    });
}
};

});
```

7.5.3 titreService.js

Voici le **service** pour les titres, celui-ci permet de communiquer directement avec le serveur php. Globalement, nous avons développé un **simili REST**, nous n'utilisons qu'un seul **verbe HTTP, POST**, avec lequel nous envoyons des **Objets JSON**.

```
//déclaration d'une factory et injection du service $http
app.factory("titreFactory", function ($http) {
  var factory = {
    titres: {},
    current: {},
    //récupère liste de titres
    getTitres: function (nombre, index, search, idParent, orderBy) {

      return $http.post(App.WEBROOT + 'plug/bayard/API/CONTROLS/titre/GetTitre.php', {nombre: nombre, index: index, search: search, id parent: idParent, orderBy: orderBy}).then(function (response) {
```

```

factory.titres = response.data;
return factory.titres;
}, function (msg) {
    console.log(msg);
});

},
//récupère un titre
getTitre: function (id) {
    return $http.post(App.WEBROOT + 'plug/bayard/API/CONTROLS/Titre/GetTitre.php', {id: id}).then(function (response) {
        factory.current = response.data;
        return factory.current;
    }, function (msg) {
        console.log(msg);
    });
},
//sauvegarde un titre
save: (function (titre) {

    return $http.post(App.WEBROOT + 'plug/bayard/API/CONTROLS/Titre/SaveTitre.php', {
        id: titre.id,
        label: titre.label,
        id_plateau: titre.id_plateau,
        key: titre.key,
        hauteur: titre.hauteur,
        largeur: titre.largeur,
        id_type: titre.id_type,
        id_partenaire: titre.id_partenaire
    }).then(function (response) {
        return response.data;
    });
}),
//efface un titre
delete: function (titre) {
    return $http.post(App.WEBROOT + 'plug/bayard/API/CONTROLS/Titre/DeleteTitre.php', {id: titre.id}).then(function (response) {
        return response.data;
    });
}
};

return factory;
});

```

7.5.4 FormService.js

Un autre service, qui utilise une **library** déjà disponible dans la solution adjoint pour créer une **boîte de dialogue**.

```

//déclaration du service
app.factory("FormFactory", function () {
    var fac = {
        reset: function(){
            fac.win = null;
        },
        //récupération du formulaire
        getForm: function (title) {
            if (fac.win === null) {
                //création de la boîte de dialogue
                fac.win = new App.ui.cmp.Window({
                    closable: true,
                    maximizable: true,
                    minimizable: true,
                    movable: true,
                    title : title,
                    resizable: true,
                    blurBelow: true,

```

```
        stayOnTop: true,
        fullscreen: false,
        w : 800,
        destroyOnClose: false,
        focus: true,
        content: $("#form" )

    }) ;

}

return fac.win;
},
win: null,
// ouverture de la boite de dialogue
open: function (title) {

    fac.getForm(title).center().open().focusForm();

}

, close: function () {

    fac.getForm().close();
}

};

return fac;

});
```

7.6 - LA PARTIE BACK-END EN PHP

J'ai décidé de **concevoir** la partie **serveur** en **PHP** comme une **API**, qui servira à servir et recevoir les données, l'avantage de cette solution, c'est de ne pas à avoir dans le futur à développer à nouveau côté serveur si je souhaitais créer une **application mobile**, ou si je souhaitais y accéder depuis un autre **module** de l'application principale.

7.6.1 Connexion.php

Pour me connecter au **SGBR MySQL**, j'ai créé ce fichier qui se trouve dans le répertoire UTILS, il sera **inclus** dans tous les **controls**. J'aurais préféré pouvoir utiliser **PDO** qui, avec une **couche d'abstraction** supplémentaire, offre des possibilités bien supérieures, notamment celle d'utiliser plusieurs pilotes, et donc communiquer avec différents SGBDR. Mais le directeur technique, m'a demandé d'utiliser **mysqli**, pour des raisons de rétro-compatibilité.

```
<?php

try {
    $connexion = new mysqli("127.0.0.1", "root", "2itech", "xplan", 3306);
    $connexion->set_charset('utf8');
} catch (Exception $ex) {
    echo $connexion->error;
    exit;
}
```

7.6.2 Titre.php

J'ai créé ensuite mes **Data Transfert Object(DTO)** ou **Entities**, qui seront le plus souvent utilisés comme composants à **persister**, ils reflètent les **tables métier**. Ce sont des **classes**, avec des **attributs privés**, un **constructeur** et des **getters et setters**.

```
<?php
/*
 * Description of Titre
 *
 * @author olivier
 */
class Titre {

    private $idTitre;
    private $nomTitre;
    private $codeTitre;
    private $hauteurTitre;
    private $largeurTitre;
    private $idTypeTitre;
    private $idPlateau;
    private $idPartenaire;
```

```

        function __construct($nomTitre, $codeTitre, $hauteurTitre,
$largeurTitre, $idTypeTitre, $idPlateau, $idPartenaire, $idTitre = 0)
{
    $this->idTitre = $idTitre;
    $this->nomTitre = $nomTitre;
    $this->codeTitre = $codeTitre;
    $this->hauteurTitre = $hauteurTitre;
    $this->largeurTitre = $largeurTitre;
    $this->idTypeTitre = $idTypeTitre;
    $this->idPlateau = $idPlateau;
    $this->idPartenaire = $idPartenaire;
}

public function getIdTitre() {
    return $this->idTitre;
}

public function getNomTitre() {
    return $this->nomTitre;
}

public function getCodeTitre() {
    return $this->codeTitre;
}

public function getHauteurTitre() {
    return $this->hauteurTitre;
}

public function getLargeurTitre() {
    return $this->largeurTitre;
}

public function getIdTypeTitre() {
    return $this->idTypeTitre;
}

public function getIdPlateau() {
    return $this->idPlateau;
}

public function getIdPartenaire() {
    return $this->idPartenaire;
}

public function setIdTitre($idTitre) {
    $this->idTitre = $idTitre;
    return $this;
}

public function setNomTitre($nomTitre) {
    $this->nomTitre = $nomTitre;
    return $this;
}

public function setCodeTitre($codeTitre) {
    $this->codeTitre = $codeTitre;
    return $this;
}

```

```

public function setHauteurTitre($hauteurTitre) {
    $this->hauteurTitre = $hauteurTitre;
    return $this;
}

public function setLargeurTitre($largeurTitre) {
    $this->largeurTitre = $largeurTitre;
    return $this;
}

public function setIdTypeTitre($idTypeTitre) {
    $this->idTypeTitre = $idTypeTitre;
    return $this;
}

public function setIdPlateau($idPlateau) {
    $this->idPlateau = $idPlateau;
    return $this;
}

public function setIdPartenaire($idPartenaire) {
    $this->idPartenaire = $idPartenaire;
    return $this;
}

```

7.6.3 IDAO.php

Prévoyant de créer un **Data Access Object(DAO)** par **DTO**, j'ai développé une **interface IDAO**. **Une interface est un contrat**, toutes les classes qui l'implémentent doivent **implémenter toutes ses méthodes**. Cela permet de garder une certaine **cohérence** entre les différentes méthodes de ma multitude de DAO.

```

<?php

/**
 * interface IDAO
 */
interface IDAO {

    public function insert($dto);

    public function update($dto);

    public function selectOneById($id);

    public function selectAll($nombre, $index, $search, $idParent,
    $orderBy);

    public function delete($id);

    public function save($dto);
}

```

7.6.4 TitreDAO.php

Les **DAO** sont les objets qui permettent de communiquer avec la base de données. C'est ici que nous utiliserons le **LMD (Langage de Manipulation des Données)**. C'est tout simplement le **CRUD(Create, Read, Update, Delete)** de la table bayard_titre dans le cas présent. Pour me simplifier la gestion de l'**insert** et l'**update**, j'ai choisi d'implémenter une méthode **save** qui sert à aiguiller en fonction de l'**état de l'objet à persister**(J'ai déjà vu ça dans l'implémentation du pattern Active Record de prestashop 1.5, j'ai donc emprunté l'idée). J'injecte également l'instance de mysqli.

```
<?php

require '../IDAO.php';

class TitreDAO implements IDAO {

    private $connexion;

    public function __construct($connexion) {
        $this->connexion = $connexion;

        if (!isset($this->$connexion)) {
            throw new Exception('Erreur, connection non initialisée !');
        }
    }

    /**
     * efface un titre
     * @param int $id
     * @return bool
     */
    public function delete($id) {
        $stmt = $this->connexion->prepare("DELETE from bayard_titre where id_titre = ? ");
        $stmt->bind_param("i", $id);
        return $stmt->execute();
    }

    /**
     * insertion d'un titre
     * @param Titre $dto
     * @return int last_insert_id
     */
    private function insert($dto) {
        $retour = 0;

        $query = "INSERT INTO bayard_titre"
            . "(nom_titre, code_titre , id_plateau, hauteur_titre, "
            . "largeur_titre , id_type_titre , id_partenaire )"
            . "VALUES(?,?,?,?,?,?)";
        if (($stmt = $this->connexion->prepare($query))) {
            $stmt->bind_param("ssiiii", $dto->getNomTitre(), $dto->getCodeTitre(), $dto->getIdPlateau(), $dto->getHauteurTitre(), $dto->getLargeurTitre(), $dto->getIdTypeTitre(),
            $dto->getIdPartenaire());
            $stmt->execute();
            $retour = $stmt->insert_id;
        }

        return $retour;
    }

    /**
     * Aiguille vers update ou insert en fonction de l'id_titre
     * @param Titre $dto
     * @return int
     */
    public function save($dto) {
        $retour = 0;

        if ($dto->getIdTitre() > 0) {
```

```

        $retour = $this->update($dto);
    } else if ($dto->getIdTitre() == 0) {
        $retour = $this->insert($dto);
    }

    return $retour;
}

/**
 *
 * @param int $nombre LIMIT
 * @param int $index OFFSET
 * @param string $search un terme à rechercher
 * @param int $idPparent id_plateau
 * @param string $orderBy ASC ou DESC
 * @return array of stdClass
 */
public function selectAll($nombre, $index, $search, $idPparent, $orderBy) {

    $arrayObj = array();

    //si $search contient des données
    $like = $search != '' ? " WHERE (t.nom_titre LIKE ? OR t.code_titre LIKE ? "
        . "OR p.nom_plateau LIKE ? OR s.nom_secteur LIKE ? "
        . "OR d.nom_domaine LIKE ? OR e.nom_editeur LIKE ?) " : "";

    //on ajoute %% à search
    $search = '%' . $search . '%';

    //si like est vide $close contient WHERE, sinon AND
    $close = $like == '' ? ' WHERE ' : ' AND ';

    //si idParent différent de 0 on concatene $close avec id_plateau=$idParent
    $str = $idParent == 0 ? "" : ($close . " t.id_plateau = " . $idParent);

    //contrôle orderBy
    $orderBy = $orderBy === 'DESC' ? "DESC" : "ASC";

    //construction de la requête
    $query = 'SELECT CONCAT(t.nom_titre, t.code_titre) as label_key, t.id_titre AS id, '
        . 't.nom_titre AS label , t.code_titre AS key , p.nom_plateau AS
label_plateau,' . 'p.id_plateau AS id_plateau, s.id_secteur AS id_secteur, s.nom_secteur AS
label_secteur,' . 'd.id_domaine AS id_domaine ,d.nom_domaine AS label_domaine, e.id_editeur
AS id_editeur,' . 'e.nom_editeur AS label_editeur, ty.id_type_titre as id_type,
pa.id_partenaire as id_partenaire,' . 'ty.nom_type_titre as label_type, pa.nom_partenaire AS label_partenaire '
        . 'FROM bayard_titre t LEFT JOIN bayard_plateau p ON p.id_id_plateau =
t.id_plateau '
        . 'LEFT JOIN bayard_secteur s ON s.id_secteur = p.id_secteur '
        . 'LEFT JOIN bayard_domaine d ON d.id_domaine = s.id_domaine '
        . 'LEFT JOIN bayard_editeur e ON e.id_editeur = d.id_editeur '
        . 'LEFT JOIN bayard_partenaire pa on pa.id_partenaire = t.id_partenaire '
        . 'LEFT JOIN bayard_type_titre ty on ty.id_type_titre = t.id_type_titre '
        . "' . $like . $str . ' ORDER BY ' . $orderBy . ' LIMIT ? OFFSET ?;';

    if (($stmt = $this->connexion->prepare($query))) {

        if ($like != '') {
            $stmt->bind_param('ssssssii', $search, $search, $search, $search, $search,
$search, $nombre, $index);
        } else {
            $stmt->bind_param('ii', $nombre, $index);
        }

        $stmt->execute();

        $result = $stmt->get_result();

        while ($o = $result->fetch_object()) {
            $arrayObj[] = $o;
        }
    }

    return $arrayObj;
}

```

```

}

/**
 * Selectionne un titre par id
 * @param int $id
 * @return stdClass
 */
public function selectOneById($id) {

    $obj = false;

    $query = 'SELECT t.id_titre AS id , t.hauteur_titre AS hauteur , t.largeur_titre AS
largeur,' .
        . ' concat(t.nom_titre , t.code_titre ) AS label_key , t.nom_titre AS label
,' .
        . 't.code_titre as key , p.nom_plateau AS label_plateau , p.id_plateau AS
id_plateau,' .
        . 's.id_secteur AS id_secteur , s.nom_secteur AS label_secteur ,
ty.id_type_titre AS id_type , ' .
        . 'pa.id_partenaire AS id_partenaire , d.id_domaine AS id_domaine ,
d.nom_domaine AS label_domaine ,'
        . 'e.id_editeur AS id_editeur , e.nom_editeur AS label_editeur ,
ty.nom_type_titre as label_type,' .
        . ' pa.nom_partenaire AS label_partenaire '
        . 'FROM bayard_titre t LEFT JOIN bayard_plateau p ON p.id_plateau =
t.id_plateau ' .
        . 'LEFT JOIN bayard_secteur s ON s.id_secteur = p.id_secteur '
        . 'LEFT JOIN bayard_domaine d ON d.id_domaine = s.id_domaine '
        . 'LEFT JOIN bayard_editeur e ON e.id_editeur = d.id_editeur '
        . 'LEFT JOIN bayard_partenaire pa on pa.id_partenaire = t.id_partenaire '
        . 'LEFT JOIN bayard_type_titre ty on ty.id_type_titre = t.id_type_titre '
        . 'WHERE t.id = ?';

    if (($stmt = $this->connexion->prepare($query))) {

        $stmt->bind_param('i', $id);
        $stmt->execute();

        $result = $stmt->get_result();

        while ($o = $result->fetch_object()) {
            $obj = $o;
        }
    }
    return $obj;
}

/**
 * mise à jour d'un titre
 * @param Titre $dto
 * @return int affected rows
 */
private function update($dto) {
    $retour = 0;

    $query = "UPDATE bayard_titre SET nom_titre = ? ,code_titre = ? , id_plateau = ?,"
        . "hauteur_titre = ? ,largeur_titre = ? , id_type_titre = ? ,id_partenaire
= ? "
        . "WHERE `id` = ?";
    if (($stmt = $this->connexion->prepare($query))) {
        $stmt->bind_param("ssiiiiii", $dto->getNomTitre(), $dto->getCodeTitre(), $dto-
>getIdPlateau(), $dto->getHauteurTitre(), $dto->getLargeurTitre(), $dto->getIdTypeTitre(),
$dto->getIdPartenaire(), $dto->getIdTitre());
        $stmt->execute();
        $retour = $stmt->affected_rows;
    }

    return $retour;
}
}

```

7.7 - LES CONTROLS PHP

Les **controls** servent généralement de **glue** entre les **entities** et les **boundaries** dans le pattern **ECB**. Dans notre cas nous n'avons pas réellement de boundaries, nos controls servent donc à faire communiquer directement les **services d'AngularJS** avec notre **couche entities**. Globalement, il reçoivent une requête HTTP de type POST contenant un **Objet JSON**, ils **contrôlent** les données entrantes, ils **instancient** un **DAO** et éventuellement un **DTO**, exécutent la **méthode** correspondante et retournent une réponse.

7.7.1 GetTitre.php

Ce control, reçoit un objet JSON contenant divers attributs et aiguille sur la bonne méthode du DAO en fonction de conditions(si l'id est défini ...) puis il retourne une réponse, sous forme d'Objet JSON également. On pourra noter l'utilisation de fonction **header** de php qui permet de spécifier le **Content-type** de la réponse.

Dans ce control, nous n'avons pas utilisé de DTO.

```
<?php

//import de la connexion
require '../UTILS/connexion.php';
require '../DAO/titre/TitreDAO.php';

header('Content-type: application/json');
$fArray = array();

//Définition des filtres
$filters = array(
    'id'=>FILTER_SANITIZE_NUMBER_INT,
    'nombre'=>FILTER_SANITIZE_NUMBER_INT,
    'index'=>FILTER_SANITIZE_NUMBER_INT,
    'search'=>FILTER_SANITIZE_STRING,
    'id_parent'=> FILTER_SANITIZE_NUMBER_INT,
    'orderBy'=>FILTER_SANITIZE_STRING
);

try {
    $data = file_get_contents("php://input");
    $src = json_decode($data);
    $dao = new TitreDAO($connexion);

    //filtrage des données
    foreach($src as $key=> $value) {
        $filteredArray[$key] = filter_var($value, $filters[$key]);
    }

    if (isset($fArray->id)) {
        echo json_encode($dao->selectOneById($fArray->id));
    } else if (isset($fArray->nombbre) && isset($fArray->index)) {

```

```

    $idParent = isset($fArray->id_parent) ? $fArray->id_parent : 0;
    echo json_encode($dao->selectAll($fArray->nombre, $fArray->index,
$fArray->search, $idParent, $fArray->orderBy));
}
} catch (Exception $e) {
    echo $e;
}

$connexion->close();

```

7.7.2 SaveTitre.php

Le control SaveTitre fonctionne globalement de la même manière que le control précédent, sauf qu'il **instancie un Objet Titre(DTO)** pour l'**injecter** dans la méthode **save** du DAO.

```

<?php
//import de la connexion
require '../../UTILS/connexion.php';
require '../../DAO/titre/TitreDAO.php';
require '../../ENTITIES/Titre.php';

header('Content-type: application/json');
$fArray = array();

//Définition des filtres
$filters = array(
    'id'=>FILTER_SANITIZE_NUMBER_INT,
    'label'=>FILTER_SANITIZE_STRING,
    'id_plateau'=>FILTER_SANITIZE_NUMBER_INT,
    'key'=>FILTER_SANITIZE_STRING,
    'hauteur'=> FILTER_SANITIZE_NUMBER_INT,
    'largeur'=>FILTER_SANITIZE_NUMBER_INT,
    'id_type'=>FILTER_SANITIZE_NUMBER_INT,
    'id_partenaire'=>FILTER_SANITIZE_NUMBER_INT,
);
;

try {
    $data = file_get_contents("php://input");
    $src = json_decode($data);
    $dao = new TitreDAO($connexion);

    //filtrage des données
    foreach($src as $key=> $value) {
        $filteredArray[$key] = filter_var($value, $filters[$key]);
    }

    $titre = new Titre($fArray->label, $fArray->key, $fArray->hauteur,
    $fArray->largeur, $fArray->id_type, $fArray->id_plateau, $fArray-
    >id_partenaire, $fArray->id);
    echo json_encode($dao->save($titre));
}

```

```
} catch (Exception $e) {
    echo $e;
}
$connexion->close();
```

7.7.3 DeleteTitre.php

Le control DeleteTitre est le plus simple des 3 concernants la table bayard_titre.

```
<?php

//import de la connexion
require '../UTILS/connexion.php';
require '../DAO/titre/TitreDAO.php';

header('Content-type: application/json');
$fArray = array();

//Définition des filtres
$filters = array(
    'id'=>FILTER_SANITIZE_NUMBER_INT
);

try {
    $data = file_get_contents("php://input");
    $src = json_decode($data);
    $dao = new TitreDAO($connexion);

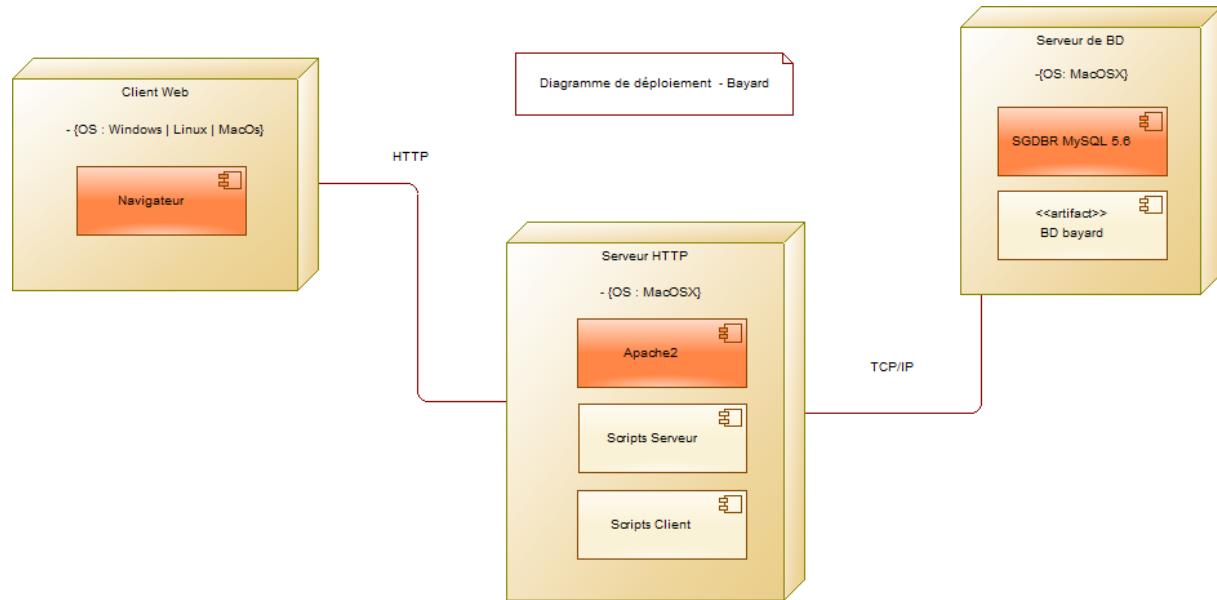
    //filtrage des données
    foreach($src as $key=> $value) {
        $filteredArray[$key] = filter_var($value, $filters[$key]);
    }

    if (isset($fArray->id)) {
        echo json_encode($dao->delete($fArray->id));
    }
} catch (Exception $e) {
    echo $e;
}

$connexion->close();
```

CHAPITRE 8 - DÉPLOIEMENT

8.1 - LE DIAGRAMME DE DÉPLOIEMENT



8.2 - LE DÉPLOIEMENT

Le **diagramme de déploiement** permet de représenter la **structure physique** du système, sur laquelle sont répartis les différents **composants** ainsi que leurs **relations**.

On retrouve principalement, dans un diagramme de déploiement, 4 types d'éléments :

- Les **nœuds(nodes)**, qui sont représentés par des cubes, sont des composants physiques du système(Serveur, Ordinateur...)
- Les **composants**, qui sont représentés par un rectangle, représentent les différentes briques logicielles du système. Les composants colorés(en orange dans notre cas), représentent des composants que nous n'avons pas conçu, ce sont le plus souvent des solutions logicielles tierces déjà en place sur le système.
- Les **associations**, sont de simples lignes représentant les voies de communications entre les nœuds du système, ainsi que leurs composants.
- Les composants avec un stéréotype **artifact**, permettent, dans notre cas, de représenter un composant qui sera consommé **durant le déploiement du système**.

Pour Le projet Bayard, le déploiement correspond donc à :

- Un poste client, sur lequel, se trouve un **navigateur web** qui lui permettra d'accéder à l'application.
- Un **serveur apache2** sur lequel nous déployeront nos différents **scripts HTML, PHP et Javascript**. Il permettra la consultation à distance.
- Un **serveur de base de données relationnelles MySQL**, sur lequel sera exécuté notre LDD lors du déploiement, ce qui aura pour effet de créer le schéma.

CHAPITRE 9 - LA GESTION DE PROJET

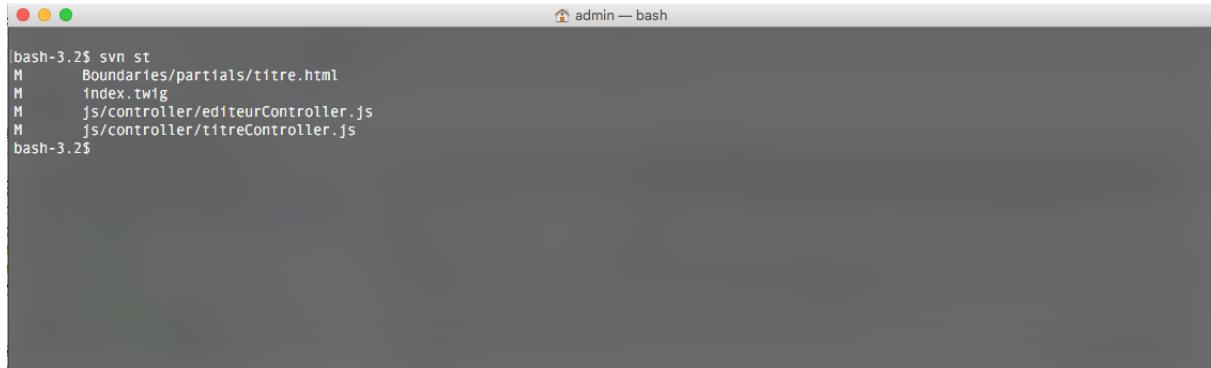
9.1.1 - SUBVERSION

Pour le versionning, l'équipe d'adiict utilise Subversion, qui est un système de gestion de version décentralisé. Il permet de travailler à plusieurs sur une ou des branches de l'application.

Les principales commandes sont :

- **update** qui permet de télécharger les dernières modification du serveur vers le poste local..
- **status** qui permet de vérifier l'état des modifications en local par rapport au serveur.
- **add** qui permet d'ajouter les nouveaux fichiers du projet au versionning.
- **commit** qui permet d'uploader les modifications et ajout locaux vers le serveur.

J'ai utilisé subversion en ligne de commande, avec le terminal :



```
bash-3.2$ svn st
M      Boundaries/partials/titre.html
M      index.twig
M      js/controller/editeurController.js
M      js/controller/titreController.js
bash-3.2$
```

CHAPITRE 10 - CONCLUSION

Passionné par le développement informatique depuis longtemps, avec mon parcours d'autodidacte, j'ai naturellement choisi de faire cette formation dans le but de me reconvertis. Mon stage en entreprise m'a définitivement conforté dans ce choix. J'ai appris énormément dans ce contexte professionnel et je continuerai à y apprendre, car l'entreprise m'a offert la possibilité de l'intégrer afin de continuer à travailler sur la solution adjoint.

J'ai grandement apprécié le principe qui m'impose de toujours me remettre en question, afin de trouver et proposer la meilleure solution pour résoudre un problème, grâce à la créativité et la logique qui m'animent.

J'ai également aimé l'idée d'avoir la possibilité d'élargir le spectre de mes compétences en restant, une partie de mon temps, en état de veille sur des technologies en constante évolution.

J'ai largement profité des phases intenses de réflexion, dont, paradoxalement, je trouve qu'elles me détendent.

Encore merci à toutes les personnes que j'ai côtoyé tout au long de ma formation et de mon stage en entreprise.

CHAPITRE 11 - ANNEXES

11.1 - CORRESPONDANCES PROJET/REAC

Activité	Compétence	Correspondance
Développer des composants d'interface		
	Maquetter une application	UML (DCU, Maquettes, DSEQ, DNAV) AGL StarUML
	Développer une interface utilisateur	
	Développer des composants d'accès aux données	Langage serveur (PHP)
	Développer des pages web en lien avec une base de données	HTML, CSS, JavaScript, AngularJS, AJAX. PHP
Développer la persistance des données		
	Concevoir une base de données (Schéma entité-Association, modèle physique normalisé)	UML (DCL) et MPD AGL PowerAMC
	Mettre en place une base de données (intégrité des données, ...)	SQL côté administrateur Création de la BD, création des tables (les types, la PK, les index UNIQUE, ...) mise en place des contraintes (FK)
	Développer des composants dans le langage d'une base de données	
	Utiliser l'anglais dans son activité professionnelle en informatique (Ecrit et parlé)	Anglais
Développer une application n-tiers		
	Concevoir une application	UML
	Collaborer à la gestion d'un projet informatique	
	Développer des composants métier	Classes « Entities »
	Construire une application organisée en couches	ECB (Diagrammes de séquence détaillés)
	Développer une application de mobilité numérique	
	Préparer et exécuter les plans de tests d'une application	
	Préparer et exécuter le déploiement d'une application	Diagramme de déploiement (UML) Déploiement. Installation de la BD. Installation des codes HTML, CSS et JavaScript. Installation des codes PHP.

11.2 CAHIER DES CHARGES & SPÉCIFICITÉS TECHNIQUES

Dans la version numérique de ce rapport de stage, vous pourrez trouver dans le répertoire Annexes du fichier zip, le cahier des charges et spécificités techniques.

Celui-ci porte le nom de « **Bayard_Spécificités techniques - V4.pdf** »

11.2.1 Autres documents

Avec le cahier des charges, j'ai également reçu des documents qui désignent la cartographie fonctionnelle. Ces documents se nomment :

- **Bayard_organigramme dev_V2.pdf**
- **Detail_arborescenceV2.xlsx**

Ils se trouvent également de dans le répertoire Annexes du fichier zip.

11.3 - OUTILS UTILISÉS ... POUR QUELS OBJECTIFS ?

Outil	Objectif	Commentaire
MySQL Workbench	Gestion de la BD	Lourd, mais son interface est agréable. Il est très efficace pour concevoir des modèles
NetBeans	IDE	Pour développer en PHP, HTML et Javascript. Peut devenir lent dans les environnements avec beaucoup de fichiers(MacOSX)
PowerAMC	Modéliser en UML	AGL assez puissant pour la génération de code. Personnellement je m'en suis servi uniquement pour les diagrammes de classe et de déploiement. Ne fonctionne pas sur MacOSX en émulation.
Balsamiq Mockups	Outil de maquettage	Offre de jolies possibilités de création de liens entre maquettes. Disponible sur Les OS les plus communs.
Bracket	Editeur de texte	Offre des fonctionnalités très intéressantes, notamment la possibilité d'insérer plusieurs curseurs pour des modifications multilignes.
StarUML 2	Modéliser en UML	Moins intéressant que StarUML 1, mais est disponible sur MacOSX. Il est encore assez immature et ne modélise pas encore toute les spécificités d'UML. Je n'ai pas trouvé de solution pour utiliser une représentation iconique sur mes diagrammes.

11.4 - BIBLIOGRAPHIE ET WEBOGRAPHIE

11.4.1 - Bibliographie

Sébastien OLLIVIER et Pierre Alexandre GURY « AngularJS Développez aujourd'hui les applications web de demain », Editions Eni, 2015

Christian Soutou, « Programmer avec MySQL », Eyrolles, 2015

Pascal Roques, « UML 2, Modéliser une application web », Eyrolles, 2008.

Pascal Roques, « UML2 Pour les bases de données », Eyrolles, 2007.

Christian Soutou avec la contribution de Frédéric Brouard, « UML 2 pour les bases de données », Eyrolles, 2012.

11.4.2 - Webographie

11.4.2.1 - HTML

<http://www.w3.org/>

11.4.2.2 - PHP

<http://www.php.net/>

11.4.2.3 - SQL

<http://sqlpro.developpez.com/>

11.4.2.4 - AngularJS

<https://angularjs.org/>

11.4.2.5 - Linq.js

<https://linqjs.codeplex.com/>

11.4.2.6 - MySQL

<http://www.mysql.fr/>

11.4.2.7 - JavaScript

<https://developer.mozilla.org/fr/docs/Web/JavaScript>

11.4.2.8 Bootstrap

<http://getbootstrap.com/>

11.5 - GLOSSAIRE / LEXIQUE

Mot clé	Description
CRUD	Concerne le LMD (Langage de manipulation de données). C : create (INSERT) R : read (SELECT) U : update (UPDATE) D : delete (DELETE)
HTTP	Protocole de communication entre 2 logiciels (un client et un serveur)
DAO	Data Access Object
DTO ou Entité	Data Transfert Object
SGDBR	Système de gestion de base de donnée relationnelle
BD	Base de données
UML	Unified Modeling Language
LDD	Langage de définition des données (CREATE, ALTER, DROP)
MPD	Modèle physique de données
MVC	Model Views Controller
ECB	Entities Controls Boundaries
MVVM	Model-View View-Model

CHAPITRE 12 - TABLES

12.1 - TABLE DES ILLUSTRATIONS

Index des images & illustrations

Adiict par O2I.....	1
Logo Bayard.....	1
Démarche Roques.....	9
Capture écran adiict.....	12
Capture écran chemin de fer.....	12
Temps de chargement adiict.....	15
Ellipse - Chemin de fer.....	17
Ellipse _ Configuration parutions et folios.....	17
Ellipse _ Saisie des imprimeurs.....	18
Ellipse _ Saisie d'un profil.....	18
Diagramme de gantt - Front office.....	21
Diagramme de gantt - Génération PDF.....	21
Diagramme de gantt - Chemin de fer.....	21
Diagramme de gantt - Back office.....	21
Légende des gantt.....	21
Diagramme UC - Projet Bayard.....	23
Diagramme UC = Gérer les publications.....	24
Diagramme UC - Gérer les titres.....	25
Maquettes - Gérer les titres.....	28
Maquettes - Modifier un titre.....	29
Maquettes - Ajouter un titre.....	29
Maquettes - Ajouter un titre 2.....	30
Maquettes - Ajouter un titre 3.....	30
Diagramme de navigation - Admin.....	31
Diagramme de séquence système.....	32
Dictionnaire de données - méthode.....	35
Matrice de dépendances fonctionnelles.....	37
Graphe de dépendances fonctionnelles.....	38
Diagramme de classe entité.....	40
Modèle physique de données.....	41
Diagramme de séquence détaillé.....	49
Copie écran - interface - Les titres.....	53
Copie écran - interface - La recherche des titres.....	53
Copie écran - interface - Modification d'un titre.....	54
Copie écran - interface - Ajout d'un titre 1.....	54
Copie écran - interface - Ajout d'un titre 2.....	55
Créer un chemin de fer.....	55
Créer un chemin de fer(suite).....	56
Modifier un chemin de fer.....	56
Arborescence du projet.....	57
Diagramme de déploiement.....	79

terminal - subversion - status.....	82
-------------------------------------	----