



Département Informatique  
U.F.R. Sciences et Techniques  
Université de Bretagne Occidentale  
UBO  
20, av Le Gorgeu C.S. 93837  
29238 BREST Cedex 3

# RAPPORT DE STAGE

## **“Refonte de la plateforme intranet d’Oziolab sous Symfony2”**

Stage du 30 mars au 19 juin 2015

Tuteur professionnel : Monsieur Michel Gléhen

Tuteur universitaire : Monsieur Djamil Sarni

Réalisé à Oziolab



Corentin Quichaud

Année universitaire 2014-2015

## Remerciements

Je tiens à remercier dans un premier temps tous les membres de l'équipe d'Oziolab pour leur excellent accueil et leur sympathie durant toute la durée du stage.

J'adresse particulièrement mes remerciements à Brewal Renault et Michel Gléhen pour m'avoir aidé et suivi lors du développement du site web. Leurs explications m'ont permis d'approfondir mes connaissances. J'ai pu apprécier leur disponibilité tout au long de mon stage.

Enfin, j'adresse mes remerciements à l'UBO, le département informatique et l'ensemble des professeurs, pour cette année d'enseignement.

# Table des matières

<b>Introduction.....</b>	<b>5</b>
<b>1. La structure d'accueil Oziolab.....</b>	<b>6</b>
1.1 Histoire d'Oziolab.....	6
1.2 Présentation de la structure d'accueil.....	6
1.3 Les serveurs.....	7
1.4 Rapport à l'entreprise.....	8
<b>2. La mission.....</b>	<b>9</b>
2.1 Le cahier des charges.....	9
2.1.1 Contexte.....	9
2.1.2 Objectifs.....	9
2.1.3 Contrainte.....	11
2.1.4 Périmètre.....	12
2.2 Langages et logiciels.....	12
2.3 Le framework Symfony2.....	13
2.3.1 Présentation de Symfony.....	13
2.3.2 Le design pattern MVC.....	14
2.3.3 Doctrine.....	16
2.3.4 Twig.....	16
2.3.5 Sonata Project.....	16
2.4 Planning Gantt prévisionnel.....	17
<b>3. Développement de la plateforme.....</b>	<b>18</b>
3.1 Installation des outils.....	18
3.1.1 Installation initiale.....	18
3.1.2 Installation de Bundle.....	18
3.2 Création des entités.....	18
3.3 La classe Admin.....	19
3.4 Les templates.....	20
3.5 Les services.....	21
3.6 Les formulaires dynamiques.....	21
3.7 L'affichage des détails.....	22

---

3.8 Les graphiques et les statistiques.....	23
3.8.1 Les statistiques.....	23
3.8.2 Les graphiques.....	23
3.8.3 Récupération des données.....	24
3.9 Les droits.....	26
3.10 Le tableau de bord.....	26
3.11 Les notifications et mails.....	28
3.11.1 Présentation.....	28
3.11.2 Fonctionnement.....	28
3.12 Les préférences.....	29
3.13 L'espace documents.....	30
3.13.1 Présentation.....	30
3.13.2 Fonctionnement.....	30
3.14 Planning Gantt réel.....	31
3.15 Problèmes rencontrés.....	31
3.16 Améliorations possibles.....	32
<b>Conclusion.....</b>	<b>33</b>
<b>Bibliographie.....</b>	<b>34</b>
<b>Glossaire.....</b>	<b>35</b>
<b>Table des annexes.....</b>	<b>37</b>

# Introduction

Ce stage a été réalisé dans le cadre de ma formation pour l'obtention de la Licence 3 Ingénierie Informatique. Il s'est effectué entre le 30 mars et le 19 juin 2015, dans une agence Web, Oziolab, à Quimper. J'ai rejoint l'équipe d'Oziolab sous la tutelle de Michel Gléhen, gérant d'Oziolab.

Michel Gléhen et Brewal Renault ont été les deux membres de l'équipe qui m'ont plus particulièrement encadré et suivi.

Mon choix s'est dirigé vers Oziolab, car le Web est une technologie que je trouve très intéressante, que ce soit d'un point de vue réalisation ou utilisation. De plus, le PHP est un langage de programmation qui m'a toujours séduit.

Par ailleurs, le web fait partie de notre quotidien et travailler au sein d'une société Web constitue une expérience constructive et concrète.

J'ai rejoint l'équipe d'Oziolab, avec pour mission la refonte d'une plate-forme Intranet sous Symfony2. Cette plate-forme sera mise à disposition de l'équipe d'Oziolab et aura pour but d'améliorer l'organisation de la société et de simplifier la gestion des clients et des projets.

# 1. La structure d'accueil : Oziolab

## 1.1 Histoire d'Oziolab

Oziolab est une SARL fondée en octobre 2010 à Quimper, en Bretagne, par deux associés complémentaires au niveau des métiers, Michel Gléhen et Michel Pouzet. Michel Gléhen, le gérant, est un Ingénieur en Informatique, ancien directeur technique d'une agence Web sur Quimper pendant 11 ans. Michel Pouzet, le responsable technique, est un technicien en développement Web depuis 18 ans. Ils ont décidé ensemble de créer Oziolab une société qui conçoit, développe des sites et applications web et héberge aussi des données. La société a pour siège social le 143 avenue de Keradennec 29000 Quimper, et a un capital de 8000 euros.

La finalité d'Oziolab est de garantir l'excellence des solutions grâce à l'expertise métier, proposer aux clients des services correspondant à leurs demandes, faire du profit et pérenniser l'organisation.

## 1.2 Présentation de la structure d'accueil

Oziolab a un effectif de trois personnes, en effet par la suite une troisième personne à été embauchée, Brewal Renault. Brewal est un Technicien en développement Web. Oziolab est une organisation à but lucratif qui propose différents services aux entreprises, spécialisée dans deux domaines :

### Conception et réalisation d'applications web :

- Conception de chartes graphiques.
- Intégration de chartes graphiques et développement de sites web :

- 
- A l'aide de gestionnaires de contenu (CMS : Joomla, Wordpress, Prestashop, Drupal).
  - Sur mesure, à partir d'un cahier des charges spécifique.
  - Développement d'applications web sur mesure (Extranet, CRM, GED).

### Hébergement de données :

- Fourniture et hébergement de solution de messagerie électronique.
- Fourniture et hébergement de solutions de messagerie collaborative.
- Hébergement de sites Internet, intranet et extranet.
- Hébergement d'applications web.
- Hébergement de données : sauvegardes distantes.
- Hébergement de serveurs dédiés.

### 1.3 Les serveurs

Oziolab dispose de nombreux clients, en effet ils ont plus de 230 clients, tel que Imbreteq, Truffaut et AXA. Ils hébergent plus de 120 sites et souhaitant apporter toujours plus de qualité et de sécurité, la société tient à héberger tous les sites Web et services associés aux clients sur leurs plates-formes composées de différents serveurs dont ils ont la propriété et la gestion complète.

Cette infrastructure est de plus physiquement placée dans une salle blanche (Data-Center) sur Quimper dont les équipements sont dimensionnés en conséquence et doublés pour répondre à une disponibilité proche de 100%.



Figure 1 : Photo de la salle blanche où sont situés les serveurs d'Oziolab

Les niveaux de sécurité dans cette salle blanche et sur les serveurs sont de plusieurs ordres et apportent une qualité d'hébergement optimale. En effet toutes les informations liées à l'environnement, l'intrusion, la sécurité, l'accès de la salle blanche sont supervisées 24h/24 et 7j/7 à distance par un centre spécialisé. Quant aux serveurs, ils sont sauvegardés quotidiennement. Une seconde sauvegarde est répliquée sur le même rythme dans un autre data-center physiquement situé en dehors du Finistère, ainsi, même en cas d'incident majeur sur le bâtiment du data-center de Quimper, les données sont conservées dans l'autre data-center. Les serveurs sont aussi protégés par firewall, plus un firewall frontal. La bande passante minimale de 10Mb/s et jusqu'à 100Mbit/s.

## 1.4 Rapport à l'entreprise

Oziolab n'ayant un effectif que de trois personnes, est une très petite entreprise. En comparaison à mon stage de l'année dernière qui c'est effectué au Cimut, qui a un effectif de plus de 100 personnes, il y a de nombreuses différences. En effet dans une entreprise de petite taille chaque personne a plus de responsabilité. Chacun à ces avantages et désavantages.

## 2. La mission

### 2.1 Le cahier des charges

#### 2.1.1 Le contexte

La société Oziolab dispose actuellement d'une plate-forme intranet qui leur permet de beaucoup mieux s'organiser en général. En effet, dans cet intranet sont répertoriés tous leurs clients, leurs projets liés, les tâches qu'ils ont à accomplir liées aux projets ainsi que les temps passés sur les tâches et les facturations.

Ainsi tout leur travail est enregistré sur cette plate-forme, ce qui est très pratique et leur permet de mieux s'organiser, il peuvent ainsi se répartir les différentes tâches à accomplir, et chacun peut voir ce qu'il lui reste à faire. Mais cela les aide aussi pour gérer les différentes facturations, en effet tout cela est calculé automatiquement suivant les tâches réalisées. Tout cela apporte un gain de temps considérable.

Cette plate-forme est actuellement sous CodeIgniter, qui est un Framework PHP et pour des raisons de sécurité et d'évolution, Oziolab souhaite la refonte de la plate-forme intranet sous Symfony2 en utilisant Sonata. Les fonctionnalités actuelles devront être présentes et donc programmées à nouveau, et des nouvelles fonctionnalités viendront s'y ajouter.

#### 2.1.2 Objectifs

Cette plate-forme disposera d'une base de données pour enregistrer les différents éléments de cet Intranet. En effet cet plate-forme contiendra de nombreux éléments, qui sont :

- Les utilisateurs : Ce sont tous les utilisateurs de la plate-forme Intranet, ils peuvent appartenir à des groupes, ce qui détermine leurs droits d'accès sur tel ou tel élément. En

---

effet un utilisateur n'a pas forcément le droit de tout faire, on peut par exemple lui interdire de créer un Projet.

- Les clients : Ce sont tout les différents clients d'Oziolab.
- Les projets : Ils sont reliés à un client, cela peut être par exemple la création d'un nouveau site Web.
- Les tickets : Ils sont eux aussi reliés à un client, et correspondent à des petites tâches à faire, tel que rappeler un client. Il est aussi associé à un utilisateur, celui qui doit effectuer ce "ticket". Cet utilisateur reçoit un mail lors de la création de ce dernier.
- Les tâches : elles sont reliées à un projet, c'est elles qui précisent sur quoi à travaillé l'équipe d'oziolab. Elles peuvent être facturables ou non. Cela peut par exemple être "Installation de Joomla".
- Les temps : Ils sont reliés à une tâche, l'équipe d'Oziolab entre leur temps passé sur telle ou telle tâches, pour, par le suite, avoir le temps total passé sur un projet.
- Les todos : Ce sont comme les tickets sauf qu'au lieu d'être en relation avec un client, ils sont en relation avec un projet.
- Les facturations : Une fois un projet terminé, il est facturé, chaque projet dispose d'un temps facturable et d'un prix à l'heure, ce qui permet de connaître le prix facturable.

Chacun de ces éléments devra être disponible depuis un menu affiché en permanence sur le site. Ils disposent de trois mode, l'affichage sous forme de liste peu détaillée, qui présentent toutes les occurrences d'un élément, par exemple tous les différents projets. L'édition, qui permet d'éditer cet élément ou d'en créer un nouveau. Et l'affichage sous forme de détail, qui présente une occurrence d'un élément en détail.

Un système de tableaux de bords différents pour chaque utilisateur devra être présent. Il référencera les projets, tâches, todos, tickets en cours pour cette utilisateur. Ainsi, cela correspond au résumé de ce qu'il lui reste à faire.

---

Tout ce qui a été dit précédemment fait partit des choses existant déjà sur l'ancien intranet et à programmer de nouveau sous Symfony. Mais de nouvelles fonctionnalités sont à ajouter :

- Un système de notification permettant de recevoir une notification lors de différentes nouveautés, par exemple lorsqu'il y a création d'un nouveau projet.
- Un système de statistiques permettant de voir différentes statistiques sur les projets et les utilisateurs. Comme par exemple un graphique du temps passé sur les différents projets.
- Un système de gestion de fichiers permettant d'ajouter différents fichiers à un projet, chaque projet disposera d'un gestionnaire du système de fichiers, et seul les utilisateurs précisés en auront l'accès.

### **2.1.3 Contrainte**

Voici les différentes contraintes de ce projet :

- Le site Web sera créé sous Symfony2.
- Le site Web devra utiliser Sonata Project, qui est un groupe d'utilitaires pour Symfony2.
- Le fonctionnement global doit être similaire à l'ancien site Web, en effet par exemple un projet doit encore avoir un nom, sa date d'ajout, un client associé etc...

### **2.1.4 Périmètre**

Cette plate-forme est créée pour Oziolab, mais aussi pour Alsyone, une entreprise travaillant dans les mêmes locaux qu'Oziolab et utilisant elle aussi cette plate-forme Intranet.

## 2.2 Langages et logiciels

### PHP

Langage de programmation pour des applications sur le web. PHP est un langage de script exécuté côté serveur. Langage élaboré inclus dans une page HTML qui permet une interaction avec l'utilisateur. Technologie permettant la création de pages web au contenu dynamique.



### HTML

L'Hypertext Markup Language(HTML), est le format de données conçu pour gérer et organiser le contenu d'une page web. C'est un langage de balisage qui permet d'écrire de l'hypertexte, d'où son nom. C'est un langage de description de données, et non un langage de programmation. Je l'ai utilisé pour créer la partie statique du site web.



### CSS

Cascading Style Sheets : feuilles de style en cascade est un langage informatique qui sert à décrire la présentation des documents HTML (et XML). Les standards définissant les CSS sont publiés par le World Wide Web Consortium (W3C). Introduit au milieu des années 1990, le CSS devient couramment utilisé dans la conception de sites web et bien pris en charge par les navigateurs web dans les années 2000.

### JavaScript

Javascript est un langage de programmation de type script, non compilé, orienté objet, principalement utilisé dans les pages Web. C'est un langage exécuté



côté client, c'est-à-dire par le navigateur de l'utilisateur. Il a pour but de dynamiser les sites Internet.

### jQuery

En JavaScript j'ai utilisé plus particulièrement jQuery, qui est une bibliothèque



JavaScript libre qui porte sur l'interaction entre JavaScript (comprenant Ajax) et HTML, et a pour but de simplifier des commandes communes de JavaScript. C'est avec cette technologie que j'ai réalisé la partie dynamique du site web.

### NetBeans

NetBeans est un environnement de développement intégré (EDI) open source. Il a été créé par Sun en juin 2000. NetBeans permet de supporter de nombreux langages dont PHP, HTML, Twig, Javascript et YML. C'est sous cet environnement que j'ai programmé l'intégralité de la plate-forme intranet.



## 2.3 Le framework Symfony2

### 2.3.1 Présentation de Symfony

Symfony est un Framework PHP français, très utilisé dans le monde des entreprises. Le projet a été lancé en 2005 par l'agence web Sensio. Au début c'était pour ses propres besoins et ensuite elle a souhaité le faire partager à toute la communauté de développeurs PHP. En tant que Framework, il facilite le développement de sites et d'applications Internet et Intranet. Symfony propose de nombreuses fonctionnalités :



- Une séparation du code en trois couches, selon le modèle MVC. Cela facilite la maintenance et l'évolution du code.
- Un système de templates évolué (Utilisant le langage Twig)
- Des performances optimisées et un système de cache pour garantir des temps de réponse optimums

- Une gestion des url parlantes, qui permet de formater l'url d'une page indépendamment de sa position dans l'arborescence fonctionnelle
- Un système de configuration en cascade qui utilise de façon extensive le langage YAML 8
- Symfony est multi langue, donc il y a un système de gestion de langue pour pour internationaliser un site.
- Le support de l'Ajax 9
- Une architecture extensible, permettant la création et l'utilisation de plugins

Il fonctionne sous forme de bundle, qui sont des modules qui ont chacun leurs propres fonctionnalités. Des bundles déjà créés par d'autres utilisateurs peuvent être réutilisés, pour faciliter la programmation, et on peut ainsi créer ses propres bundles et les faire partager à la communauté. De nombreux sites et applications de toutes tailles et de tous les types utilisent Symfony comme par exemple Yahoo! et Dailymotion. La documentation est complète et elle est mise à jour assez régulièrement, ce qui m'a facilité l'apprentissage de ce Framework qui m'était inconnu.

### 2.3.2 Le design pattern MVC

Un design pattern est traduit par "patron de conception". Le design pattern MVC, accronyme de l'expression "**Modèle - Vue - Contrôleur**" est un modèle qui a fait ses preuves et qui s'avère indispensable à connaître dans le domaine de la programmation. Le code est ainsi séparé en trois couches :

- **Modèle** : Le modèle est la partie du code qui se charge d'interagir avec la base de données. On y retrouve donc les appels SQL que nous aurons définis. Dans le cas de Symfony, cette partie utilisera Doctrine, qui sera présenté par la suite.
- **Vue** : La vue est la partie du code qui se charge uniquement d'afficher les données qui lui ont été fournies. En aucun cas, la vue ne doit effectuer de traitement sur les données, sauf dans le but d'en embellir l'affichage. Pour ce faire Symfony utilise le langage Twig qui permet de gérer l'affichage de données.

● **Contrôleur :** Le contrôleur est le centre de notre design pattern. Il reçoit la requête HTTP, l'interprète et coordonne le tout. Il se charge de demander au modèle les données puis effectue plus ou moins de traitements dessus afin d'envoyer à la vue les données à afficher et de retourner une réponse à l'émetteur de la requête.

Voici un schéma présentant le design pattern MVC:

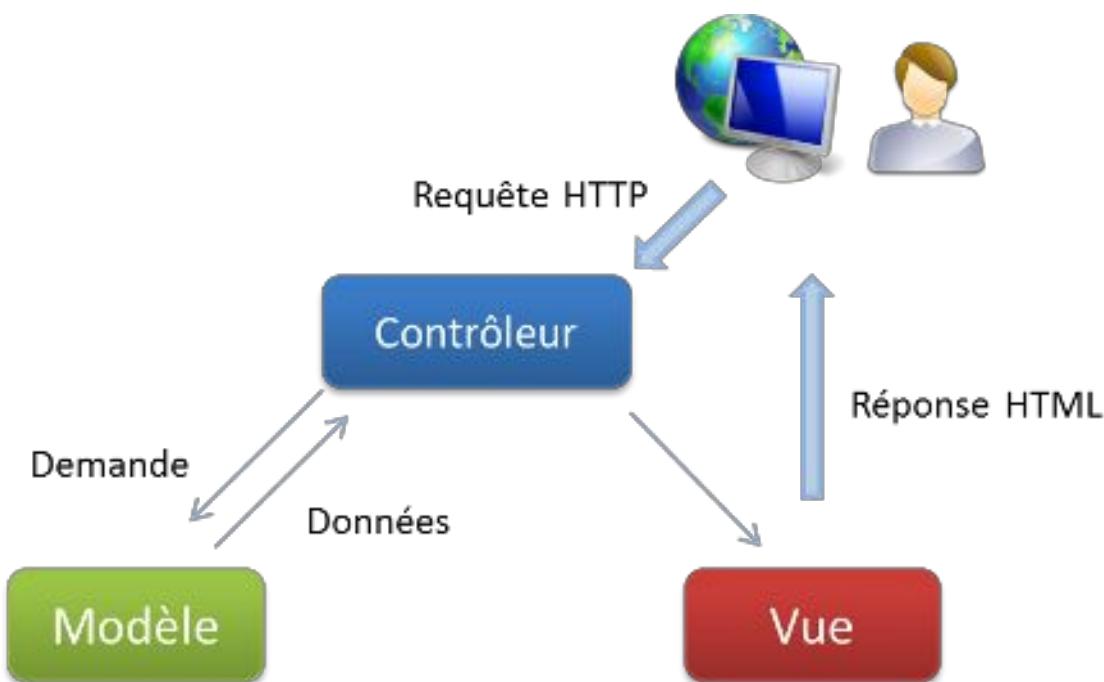


Figure 2 : Schéma MVC (Source: zestedesavoir.com)

### 2.3.3 Doctrine

Plusieurs outils de mapping objet-relationnel (ORM, pour «Object-RelationalMapping») sont disponibles pour PHP, tels que Propel, [Doctrine], PHPDataMapper ou encore PHPMyObject. Les deux premiers sont néanmoins les plus réputés et les plus utilisés. Ils sont respectivement soumis aux licences MIT et LGPL.



La version 2 de Doctrine utilise un patron de conception : Data Mapping. Cela consiste à faire correspondre deux modèles de données : le modèle objet et le modèle de la base de données. L'avantage est que l'on peut ainsi accéder aux données sans connaître la structure de la base. Doctrine possède son propre langage de requêtage,

DQL (pour « Doctrine Query Language») qui permet de gérer la persistance des données d'une manière SQL orienté objet.

### 2.3.4 Twig

Dans Symfony le PHP et le HTML sont entièrement séparer, le HTML est inséré dans les fichiers Twig. Twig est un moteur de template PHP directement intégré dans Symfony2 et créé lui aussi par Sensio. Très puissant, Twig permettra de gérer de l'héritage entre templates et layout, séparer les couches de présentation et couche métiers. Ce qui est idéal surtout si le travail est effectué avec des intégrateurs, qui n'auront qu'à modifier les templates dans le répertoire “Views” des différents bundles.



### 2.3.4 Sonata Project

Pour réaliser ce projet j'ai utilisé Sonata Project qui est un regroupement de bundles open source pour Symfony permettant la simplification de nombreuses fonctionnalités. Il est majoritairement basé sur le travail de la communauté et utilise les bundles les plus connus permettant par exemple de gérer un système de comptes utilisateurs. Sonata permet aussi d'avoir une barre de navigation, de gérer les listings et les éditions d'occurrences dans la base de données.



## 2.4 Planning Gantt prévisionnel

Voici le planning prévisionnel de mon projet:

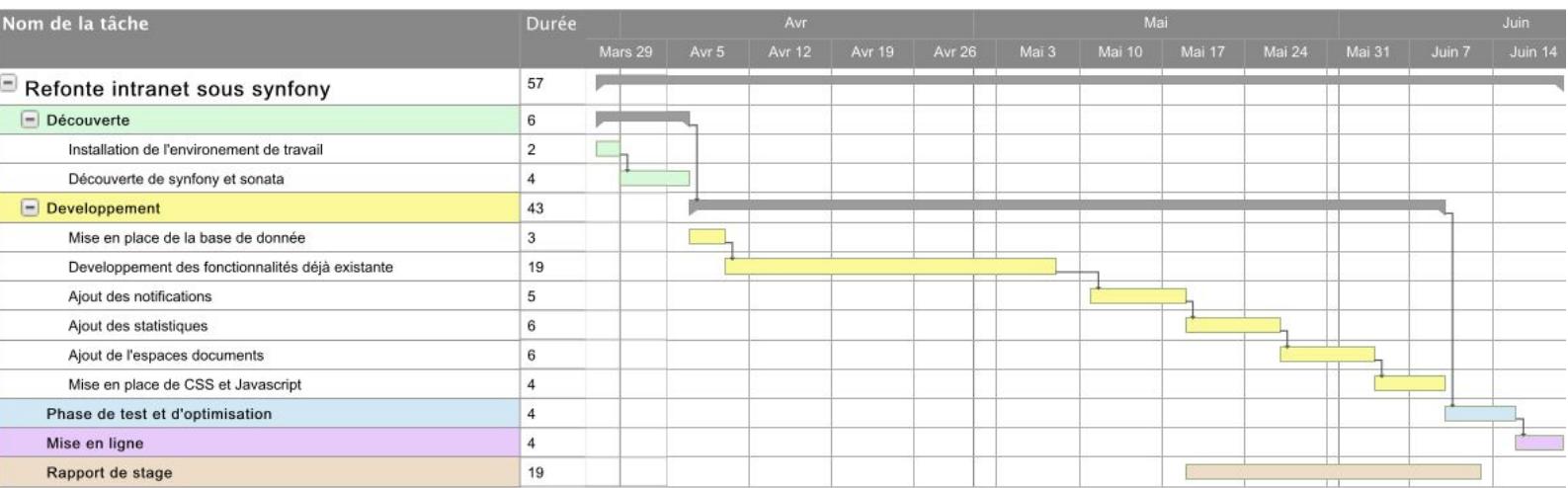


Figure 3 : Planning prévisionnel du projet

## 3. Développement de la plateforme

### 3.1 Installation des outils

#### 3.1.1 Installation initiale

Tout d'abord, j'ai dû installer Ubuntu desktop sur mon ordinateur. En effet le système d'exploitation actuel étant windows, cela ne correspondait pas à mes besoins. Par la suite j'ai installé une sandbox de Sonata Project. Ce qui permet d'avoir un environnement Sonata, comprenant Symfony2 ainsi que tous les Bundles de Sonata. Et enfin j'ai installé NetBeans, c'est dans ce dernier que je programmerai l'intégralité du site Web.

#### 3.1.2 Installation de Bundle

Une fois l'environnement Symfony/Sonata installé, il m'a fallut créer mes premiers Bundle, dans lesquels je programmerai les différentes fonctionnalités de l'Intranet. Pour commencer j'ai créé trois bundles. Un nommé "Projet", qui comprendra tout ce qui concerne les Projets (Projets, Tâches, Todos, Temps, Facturation). Un nommé "Client" qui comprendra tout ce qui concerne les Clients et pour finir un nommé "Ticket".

Un Bundle peut être représenté comme un dossier comprenant des sous dossiers. Un bundle est découpé suivant le patron de conception MVC. En effet il y aura un dossier comprenant les contrôleurs, un autre pour les entités, et un dernier pour les templates. Un bundle sous Sonata dispose aussi d'un dossier Admin, qu'on expliquera plus tard, et un dossier Ressources, pour les différents fichiers Javascript et Css.

### 3.2 Crédation des entités

Pour commencer j'ai créé les entités, cela correspond à la partie base de données, il peut y en avoir plusieurs dans un même Bundle. Elles sont créées en utilisant doctrine. Une entité est classe PHP qui correspondra à une table dans la BDD. Dans cette classe sont détaillées les différentes colonnes, leurs types, leurs relations avec les autres colonnes si elles en ont. Il existe trois types de relations :

- **Relation OneToMany** : Une relation se produit lorsque chaque enregistrement dans une table A peut avoir plusieurs enregistrements liés dans une table B, mais chaque enregistrement de la table B peut n'avoir qu'un seul enregistrement correspondant dans la table A.
- **Relation ManyToMany** : Une ligne dans la table A peut avoir plusieurs lignes correspondantes dans la table B, et vice versa.
- **Relation ManyToMany** : Une ligne dans la table A peut avoir une ligne correspondante dans la table B, et vice versa.

Par exemple un projet a un unique client, mais un client peut avoir plusieurs projets. Ainsi dans la table projet il y aura une colonne clients qui aura une relation ManyToOne vers Client. Et dans client il y aura une colonne projet qui aura une relation OneToMany vers projet.

Sont précisés aussi dans cette classe les différents “getter” et “setter”, permettant de récupérer ou définir les colonnes. Et d'autres fonctions personnalisées peuvent y être ajoutées, par exemple une fonction permettant de calculer le temps total passé sur un projet. Chacune des occurrences d'une entité, aura l'accès à ces fonctions. Par exemple si on manipule un objet de type “Projet” on pourra accéder au temps total passé sur ce dernier ICI un UML

### 3.3 La classe Admin

Sonata permet de faciliter de nombreuses fonctionnalités. Par exemple, pour chaque classe entité, elle peut générer une page contenant une liste, contenant les différentes occurrences de cette dernière, une page contenant un formulaire pour modifier ou créer une entité et une page permettant d'afficher les détails d'une entité particulière.

Tout ce qui est affiché est paramétrable dans la classe Admin de l'entité. Ainsi on détaille les champs que l'on souhaite dans la liste, dans le formulaire, etc...

Voici comment est configuré la liste :

```
protected function configureListFields(ListMapper $listMapper) {
    $listMapper
        ->add('nom', null, array('template' => 'OzioProjetBundle::nomList.html.twig'))
        ->add('client', null, array('template' => 'OzioProjetBundle::projetClientList.html.twig'))
        ->add('passé', 'string', array('label' => 'Temps passé', 'template' => 'OzioProjetBundle::projetPasse'))
        ->add('facturable', 'string', array('label' => 'Temps fact.', 'template' => 'OzioProjetBundle::projetFacturable'))
        ->add('facture', null, array('label' => 'Temps facturé', 'template' => 'OzioProjetBundle::projetFacture'))
        ->add('datetime', 'date', array('label' => 'Ajouté le', 'pattern' => 'dd/MM/YY', 'locale' => 'fr', 'timeZone' => 'Europe/Paris'))
        ->add('finit', null, array('editable' => true, 'label' => 'Terminé'))
        ->add('Fact', null, array('label' => 'Actions', 'template' => 'OzioProjetBundle::projetBoutonFacture.html.twig'))
}
```

Figure 4 : Code pour la configuration de la liste

On ajoute (“add”) à la liste chacune des colonnes de l’entité. Si l’entité est construite directement grâce à la valeur en base de données, il n’y a pas besoin de lui donner un template. Mais par contre dans le cas inverse, si on veut modifier cette valeur, comme pour ajouter un lien cliquable, ou encore faire un calcul (pour le temps facturable par exemple). Dans ces cas là on précise un template.

Sonata gère aussi un système de filtre pour les listes qui est lui aussi paramétrable, ce qui permet de filtrer les résultats sur différents critères.

On peut aussi modifier les actions à réaliser à certains moments, tel que lors d’une création, d’une suppression... Dans cette classe on peut aussi préciser les templates à utiliser pour l’affichage. Et ajouter des routes, qui peuvent être représentées comme des nouvelles pages, ou une action sera réalisée (dans le contrôleur), et cette dernière pourra soit retourner un template, qui sera affiché, soit retourner des informations, en JSON par exemple. Ainsi chacune de nos entités créées précédemment auront leur propre classe admin.

### 3.4 Les templates

Les templates sont écrits en Twig et sont utilisés pour l’affichage de certains éléments. Ainsi par exemple, on peut changer ce que doit afficher une colonne de la liste générée par Sonata. Ou on peut encore totalement modifier l’affichage de la page de détail de Sonata. Voici l’exemple d’un template extrêmement simple :

```
{% extends 'SonataAdminBundle:CRUD:base_list_field.html.twig' %}

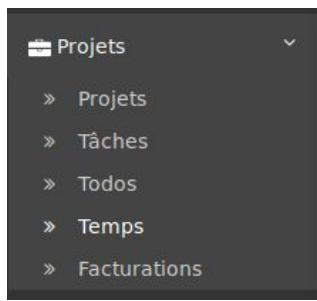
{% block field %}
    <div>
        <u title="{{object.description}}">
            {{object.nom}}
        </u>
    </div>
{% endblock %}
```

Figure 5 : Exemple de template

Il est utilisé pour l'affichage de la colonne “Nom” d'un projet, ici on souligne juste le nom, et au sur-volage de ce dernier on affiche sa description.

### 3.5 Les services

Une fois la classe admin créée, on crée un service qui “contiendra” cette classe. Les



services sont programmés en yml. Par la suite on ajoutera le service au configuration de Sonata.

En effet le menu de Sonata est configuré sous forme de groupe, chaque groupe peut contenir plusieurs services Admin. Visuellement un groupe est une liste déroulante (voir le schéma à gauche).

Ainsi maintenant lorsqu'on cliquera sur “Projets” par exemple, on arrivera sur la liste des projets. Cette liste peut contenir différents champs, et différentes actions, tel que “éditer”, “créer” ou encore “facturer”.

Les services seront par la suite aussi utilisés pour ajouter un bloc au tableau de bord de Sonata ou pour créer des types personnalisés pour les formulaires.

### 3.6 Les formulaires dynamiques

A de nombreux endroits j'ai eu besoin d'avoir des éléments dynamiques. Par exemple, une facturation est enregistrée sous forme d'un prix en euro. Mais les tâches facturables sont enregistrées sous forme d'un temps. Ainsi pour faire le lien entre les

deux, les clients disposent chacun d'un prix horaire. Ce prix peut être aussi personnalisé pour un projet particulier. Ainsi dans le formulaire de facturation, un détail du nombre d'heures facturables est affiché.

L'utilisateur peut entrer une heure, et à chaque modification de ce champ, le champ du prix total sera modifié en direct suivant le tarif horaire du projet. L'utilisateur peut aussi, si il le souhaite entrer directement un prix. Le champ d'heure étant là, juste pour calculer le prix, et donc pas enregistré dans la base de données, son renseignement n'est pas obligatoire. Lors d'une modification du projet dans le formulaire de facturation, tout doit être recalculé avec le nouveau tarif horaire et on doit afficher le nouveau temps facturable.

Ainsi j'utilise des appels Ajax, qui permettent d'accéder à des routes. J'ai donc créé une route renvoyant les différents temps passé en JSON suivant le projet donné en paramètre. Grâce à Ajax, la requête est faite en background, ainsi l'utilisateur ne voit pas qu'une requête est effectuée. A de nombreux endroits dans le site web j'utiliserai Ajax.

### 3.7 L'affichage des détails

Pour l'affichage des détails d'un projet ou d'un client, j'ai entièrement modifié ce qui est généré par sonata. En effet Sonata génère un tableau sous forme

Champs	Valeur du champs
--------	------------------

Mais cela ne correspondait pas à la demande. Prenons pour exemple l'affichage des projets. Pour ce dernier il faut afficher un graphique "Camembert" où est précisé le pourcentage de temps passé sur les 5 tâches avec le plus de temps et sur les "autres", nous parlerons plus en détails des graphiques par la suite. Il faut aussi afficher plusieurs listes sur les éléments concernant ce projet. Une liste pour les Todos en cours, une autre sur l'ensemble des Todo. Une sur les Tâches en cours et une dernière sur l'ensemble des tâches. J'utiliserai ainsi des onglets pour passer d'une liste à l'autre pour faciliter l'affichage. Dans ces listes nous pourrons modifier quelques informations en direct grâce à des appels, tel que passer à terminer une tâche, rendre facturable une tâche. Todo et

tâches ont deux listes chacun, ainsi lors de chaque modification dans une des listes il faudra aussi répercuter cette modification sur l'affichage de la deuxième liste. J'ai aussi créé une dernière liste pour l'espace document d'un projet, nous en parlerons par la suite.

Pour faire tout cela j'ai créé un template où j'ai créé les différents éléments de cette affichage. Grâce au différentes fonctions dans l'entité j'ai eu accès à toutes les données dont j'avais besoins sans faire de nouvelle requête dans la base de données. J'y ai aussi chargé différents Javascript pour les appels Ajax et le graphique. J'ai précisé à Sonata d'utiliser ce template plutôt que celui qu'il utilise normalement.

## 3.8 Les graphiques et les statistiques

### 3.8.1 Les statistiques

A différents endroits dans l'application j'ai dû créer des graphiques. Comme dit précédemment dans les détails des projets ou des clients, mais aussi dans une partie nommé "Statistiques". Cette partie contiendra deux types de statistiques, tout d'abord des statistiques sur l'ensemble des projets, et des statistiques sur les différents utilisateurs, leurs temps passé etc... Ici aussi je vais donc créer deux onglets pour séparer ses parties distinctes et dans l'onglet "Utilisateurs" je créerai plusieurs sous onglets pour chaque utilisateur, les graphiques serons par la suite affichés grâce au Javascript. Je vais prendre pour exemple les statistiques de projet et en parler plus en détails.

### 3.8.2 Les graphiques

Pour faire les graphiques j'ai utilisé deux plug-ins différents. Chartjs pour les graphiques radial pour le détails d'un projet et d'un client. Et Hightcharts pour les graphiques des statistiques générales. Tous deux sont écrits en javascript, mais hightcharts à l'avantage d'être facilement paramétrable, en effet grâce à ce dernier on peut plus facilement modifier le graphique en lui même.

Pour les statistiques de l'ensemble des projets, j'ai créé un histogramme, qui sera composé sur l'axe des abscisses des 7 projets les plus récents (Avec un temps ajouté sur

l'une des tâches récemment), et sur l'axe des ordonnées d'un temps. Chacun de ces projets sera séparé en deux "barres".

Une barre concernera les temps séparé en deux, une partie pour le temps facturable et une autre pour le temps passé non facturable. Ces deux parties représenteront le temps total passé.

L'autre barre concernera le nombre d'heures facturées. Comme vue précédemment, ce ne sont pas des heures qui sont facturées mais un prix. Ainsi ce nombre d'heure sera calculée suivant le prix horaire.

Une comparaison est donc possible entre le temps passé et facturable et le temps facturé. Et on peut aussi facilement voir sur quel projet il y a le plus de temps passé. J'ai aussi créé un système de période, pour voir les statistiques sur une période donnée, lors du changement d'une période les graphiques sont rechargés grâce à des appels Ajax, sans que la page n'ait à se recharger.

### 3.8.3 Récupérations des données

Pour récupérer les différentes données j'ai utilisé un contrôleur, ce contrôleur peut prendre en paramètre une période, si aucune période n'est donnée elle est définie par défaut aux 30 derniers jours. Par la suite une requête vers la base de donnée est effectuée suivant la période donnée. Une requête en base de données en SQL est différente d'avec Doctrine. En effet avec doctrine pour effectuer une requête il y a deux méthodes. Tout d'abord récupérer "l'entity manager", qui comme son nom l'indique est ce qui gère les entités, une fois ce dernier récupéré on peut récupérer le "Repository" qui correspond à une entité, par exemple client. Une fois ce dernier récupéré on peut y chercher des occurrences grâce à des requêtes simples tel que find, pour trouver une occurrence suivant son ID, ou encore findBy, pour trouver des occurrences suivant une valeur d'une ou plusieurs colonnes. C'est cette méthode que j'ai utilisé pratiquement tout le reste du projet.

Mais dans notre cas, la requête étant plus complexe, nous devons utiliser la deuxième méthode. Au lieu de récupérer le Repository d'une entité, on va créer une "queryBuilder", qui permet de construire une requête. Dans cette dernière on va détailler

grâce à des fonctions déjà existante les différentes parties de notre requête. Voici la requête effectuée pour récupérer la liste des projets :

```
$qb = $em->createQueryBuilder();
$qb->select('p,' . $qb->expr()->max('d.datetime') . ' AS dernier_temps')
    ->from('OzioProjetBundle:Projet', 'p')
    ->join('p.taches', 't')
    ->join('t.duree', 'd')
    ->where('d.datetime BETWEEN :debut AND :fin')
    ->groupBy('p.id')
    ->orderBy('dernier_temps', 'DESC')
    ->setParameters(array('debut' => $debut, 'fin' => $fin));

$query = $qb->getQuery();
$projects = $query->getResult();
```

Figure 6 : Exemple de requête

Dans cette requête on sélectionne tout les projets (p), triés par “dernier\_temps”, qui correspond à la date ou le dernier temps d'un projet a été ajouté durant une période. Cette période est composé d'une date de début et de fin, qui sont passé en paramètre. Ainsi on aura une liste de projets, avec en tête de liste le projet le plus “récent” et en fin de liste le projets le plus anciens, lors de cette période. Si cette liste contient plus de 7 projets, je ne garde que les 7 premiers. Et je crée un JSON qui contient quatre tableaux. Un référençant les noms des projets, un référençant le temps passé, un autre le temps facturable et un dernier le temps facturé.

En effet pour créer le graphique par la suite, on détaille des tableaux de valeurs, par exemple pour la partie temps facturable du graphique on y insère le tableau avec tout les temps facturables. Les index étant dans l'ordre, chaque donnée sera au bon endroit. Ces données sont récupérées en Javascript et affichées par le suite

### 3.9 Les droits

Symfony met à disposition un bundle permettant de gérer des groupes avec différents droits. Ainsi j'ai exploité ce bundle pour créer plusieurs groupes :

- Le groupe Administrateurs qui aura tous les droits
- Le groupe Utilisateurs qui pourra :
  - Voir tous les Todos, Projets, Clients, Facturations, Temps, Tickets, Tâches, Clients.
  - Ajouter des occurrences à ces derniers.
  - Éditer seulement les occurrences qu'il aura créé
- Le groupe Observateurs qui pourra seulement voir tous les Todos, Projets, Clients, Facturations, Temps, Tickets, Tâches, Clients.
- Et un groupe pour chaque entité (Ticket, Tache, etc..)

Ainsi chaque utilisateur peut appartenir à un ou plusieurs groupes, pour lui donner les droits d'accès sur ces derniers. Par la suite j'ai effectué dans tout le site web des vérifications pour savoir si l'utilisateur a le droit d'accès à tel ou tel endroit. J'ai aussi modifié l'affichage de certaines parties du site, par exemple un utilisateur qui n'aura pas le droit de facturer un projet, n'aura pas le bouton "facturer" devant un projet.

### 3.10 Le tableau de bord

Le tableau de bord est différent pour chaque utilisateur, celui ci référence plusieurs éléments en liens avec l'utilisateur, suivant les droits que dispose l'utilisateur, ce dernier peut avoir accès à plusieurs boutons, tel que l'édition ou la suppression ou encore l'ajout d'un élément. Le tableau de bord est constitué de block, chaque block concerne un élément :

- La liste paginée des rendez-vous de la semaine, en effet dans un ticket peut être précisé une date de rendez-vous, ce rendez-vous est rappelé sur le tableau de bord
- La liste paginée des tâches en cours, depuis le tableau de bord l'utilisateur peut leur ajouter un temps ou les marquer comme terminé (Si il en a les droits).

- La liste paginée des todos en cours, depuis le tableau de bord l'utilisateur peut changer l'état d'un todo (Si il en a les droits).
- La liste paginée des tickets en cours, depuis le tableau de bord l'utilisateur peut changer l'état d'un ticket (Si il en a les droits).
- La liste paginée des projets, depuis le tableau de bord l'utilisateur peut facturer un projet ou le marquer comme terminé (Si il en a les droits).
- Le graphique des temps passés d'un utilisateur

Sonata dispose d'un bundle appelé block, qui permet la création de ces derniers. Ainsi on crée une classe block pour chacun d'eux, on précise dans cette classe ce qu'il y a à réaliser, et le template à utiliser pour l'affichage. Par la suite comme pour la classe admin, on crée un service que l'on ajoute à la configuration de sonata. Voici l'exemple d'un block :

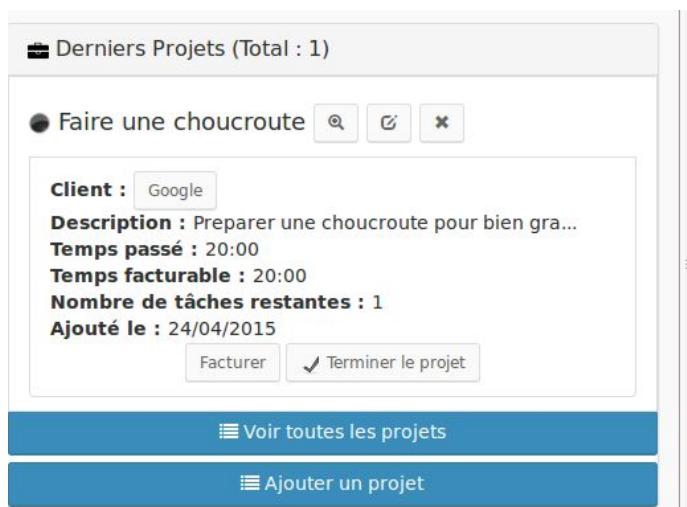


Figure 7 : Exemple d'un block

## 3.11 Les notifications et mails

### 3.11.1 Présentation

J'ai créé un système de notifications. Ainsi lors de la création d'une tâche, d'un projet, d'un client ou lors d'autres actions importantes concernant un utilisateur, ce dernier reçoit une notification pour lui en informer.

Ces notifications sont affichées en haut à droite grâce à un icône, à côté de cet icône est présent un compteur qui correspond au nombre de notifications non vues. Lorsqu'on clique dessus, cela fait apparaître une liste déroulante de notifications. Cette liste contient toutes les notifications non vues, et au minimum, 10 notifications. C'est à dire que si il y a moins de 10 notifications non vues, on affiche les dernières notifications vues.

Elles sont accessibles depuis n'importe quelle page du site. Une notification peut être marquée en vue en cliquant sur l'icône valider à côté d'elle. L'utilisateur peut aussi marquer toutes les notifications en vue. Quand une notification est marquée comme lu, le compteur est mis à jour sans avoir besoin de recharger la page.

Quant aux mails, ils ne sont envoyés qu'à deux moments, lorsqu'on crée un ticket qui est à réaliser par un autre utilisateur, ce dernier reçoit un mail. Et lorsqu'on ajoute un utilisateur à l'espace document d'un projet.

### 3.11.2 Fonctionnement

Tout d'abord j'ai créé un nouveau bundle avec deux entités, notification et mail. Ainsi ils seront insérés en base de données. Les notifications ont une colonne "vue" pour savoir si cette dernière a déjà été vue ou non. Il est précisé aussi l'utilisateur concerné par cette notification, le message et la date.

Par la suite j'ai créé un service, envoyant les mails et un autre les notifications. Ce service prend en paramètre l'action qui est réalisée et l'élément en question, ceux ci vont permettre de définir le message à envoyer, et à qui l'envoyer. En effet lors de la création

de Todo ou de Tâches, la notification n'est envoyée qu'à une personne. Tandis que lors de la création d'un projet, tous les utilisateurs doivent être mis au courant.

Toutes les classes Admin extenedes la classe Admin de sonata, ainsi j'ai réécrit différentes fonctions de cette classe, comme la création. Et j'ai fait appel au service créé précédemment. Par la suite toutes les classes nécessitant l'envoi de notifications ou de mail, extenderons désormais la nouvelle classe Admin créée. J'ai aussi modifié la fonction de suppression, en effet, si l'on crée un projet et qu'on le supprime après, il faut aussi supprimer les notifications envoyées aux différents utilisateurs.

Une fois cela créée j'ai modifié le template général de Sonata pour y insérer ma partie notification, et intégrer les différents javascript.

Voici un aperçu final des notifications:

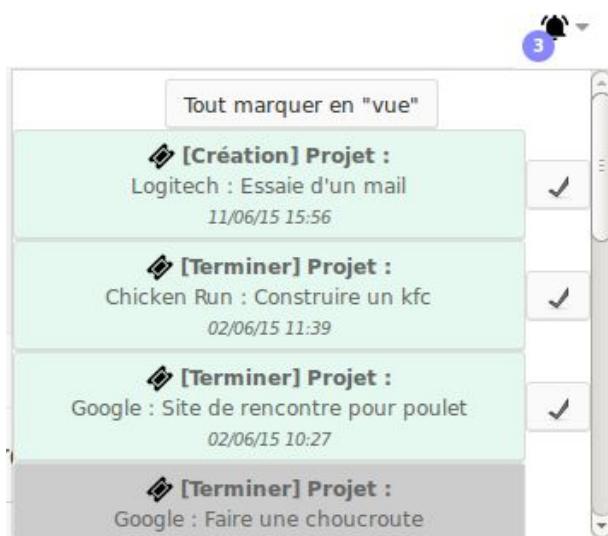


Figure 8 : Les notifications

En vert, les notifications non vues et en gris, les notifications vues. Lorsqu'on clique sur une notification on est redirigé à l'endroit correspondant. Par exemple pour une notification d'un projet terminé, on sera redirigé à la page de détail de ce projet.

### 3.12 Les préférences

J'ai créé un système de préférences pour les notifications et les blocs, ce qui permet à l'utilisateur de choisir quels genres de notifications il souhaite recevoir et quels blocs sur le tableau de bord il souhaite afficher.

---

En effet un utilisateur n'étant pas concerné par les Todos par exemple, ne souhaite pas forcément recevoir de notifications pour celles-ci, et ne souhaite pas avoir le bloc Todos. Ainsi chaque utilisateur peut personnaliser son tableau de bord.

## 3.13 L'espace document

### 3.13.1 Présentation

Chaque projet dispose de son propre espace documents. Ce dernier n'est accessible que par les utilisateurs autorisés. En effet dans les paramètres d'un projet on précise quels utilisateurs doivent avoir l'espace document, et lesquels peuvent juste voir les documents ou en ajouter.

Cet espace documents est accessible, soit dans les détails d'un projet, soit depuis un nouveau, "documents", groupe dans le menu Sonata. Lorsqu'on clique sur ce groupe, on arrive sur la liste des projets dont on a accès à l'espace documents. Et par la suite on peut entrer dans tel ou tel espace documents.

L'espace documents est sous forme de listes, est précisé dans cette dernière, le type de chaque document, son nom, son poids et sa description. De là l'utilisateur peut télécharger les documents qu'il souhaite.

Une catégorie historique est aussi présente, cette catégorie permet de lister les dernières actions réalisées, c'est à dire les différents ajouts et téléchargements de documents. Cela permet à l'administrateur de savoir qui a vu tel document et qui ne l'a pas encore regardé.

### 3.13.2 Fonctionnements

J'ai créé un nouveau bundle, "document", qui contient deux entités, les documents et l'historique. Chaque document contient un nom, un projet, et est associé à un fichier grâce à une URL. Lors de l'ajout d'un document, ce dernier est enregistré dans un dossier "documents" sur le serveur.

J'ai aussi créé une action dans le contrôleur pour télécharger un document passé en paramètre. Cette action vérifie entre autre que l'utilisateur a les droits pour télécharger ce document. Pour la liste de document j'ai utilisé "cute files browser", que j'ai modifié. Ce dernier permet d'avoir un affichage similaire à l'explorateur windows. Une zone de recherche est aussi disponible pour rechercher un document précis.

### 3.14 Planning Gantt réel

*Voici le planning réel de mon projet :*



Figure 9 : Planning réel du projet

### 3.15 Problèmes rencontrés

Ce stage est une entière découverte pour moi ne connaissant absolument pas Symfony, donc par conséquent pas non plus Sonata, Doctrine et Twig. Par conséquent j'ai rencontré de nombreux problèmes, et j'ai parfois passé beaucoup de temps sur des choses qui s'avéraient finalement très simple. Mais cela m'a permis de gagné beaucoup en autonomie et en logique pour résoudre des problèmes. Une fois le Framework pris en main, j'ai donc avancé beaucoup plus rapidement.

### 3.16 Améliorations possibles

A certains endroits le design du site web pourrait être plus embellit. Et dans les détails du projet, il manque un graphique, plus ou moins similaire à un Gantt. Ce graphique permettrait de savoir à quel moment on a passé du temps sur tel ou tel tache. Mais n'ayant pas trouvé de plug-in permettant de faire un tel graphique et n'ayant pas eu le temps de le programmer moi même grâce à Highcharts par exemple, ce graphique n'est, par conséquent, pas encore présent sur le site. Mais j'ai en globalité respecté l'ensemble du cahier des charges.

# Conclusion

Oziolab utilise une plate-forme Intranet pour la gestion de ces clients et de ces projets, elle permettait de faciliter leur organisation. Pour des questions de sécurité et d'évolution, on m'a confié la mission d'effectuer la refonte sous Symfony2 de cette plateforme qui est actuellement sous codeigniter.

Ne connaissant absolument pas Symfony2 ce stage a été de l'entièvre découverte. J'ai eu du mal à prendre le framework en main, mais une fois ceci fait, cela a été beaucoup plus simple. J'ai ainsi réalisé pratiquement la totalité du cahier des charges. L'ensemble des anciennes fonctionnalités, ont été programmées à nouveau et de nouvelles ont été ajoutées.

Ce stage s'est merveilleusement bien passé. Il a été très enrichissant, j'ai pu y découvrir de nouvelles technologies que je n'ai pas eu l'opportunité de voir lors de mon cursus à l'université telles que Symfony, Sonata et d'autres. Ce qui pourra grandement me servir dans mon avenir professionnel.

De plus j'ai découvert un environnement professionnel intéressant, en effet mon précédent stage s'étant effectué dans une entreprise relativement grande, j'ai pu voir la différence avec une petite société et y voir les avantages comme les inconvénients.

# Bibliographie

Site web d'Apache, <http://www.apache.org/>

Site web du W3C, <http://www.w3c.org/>

Site web de Developpez.com, <http://www.developpez.com/>

Site web de l'encyclopédie libre Wikipedia, <http://www.wikipedia.fr/>

Site web de Netbeans, <https://netbeans.org/>

Site web d'un développeur, <http://www.jmdoudoux.fr/>

Site web d'openclassroom, <http://openclassrooms.com/>

# GLOSSAIRE

**Framework:** "Framework" est un terme tout droit venu de l'anglais signifiant "cadre de travail". Un framework est là pour vous aider à structurer votre projet. Il apporte un ensemble d'éléments qui définissent le squelette d'une application et certaines briques de base qui vous facilitent la vie. Un framework offre tout un panel d'outils pour aider le développeur dans la réalisation de son application tout en offrant une certaine liberté dans la réalisation technique de celle-ci.

**AJAX:** acronyme pour Asynchronous JavaScript And XML. C'est un aspect de la programmation web qui permet de modifier certaines parties de la page web sans devoir recharger la page entière.

**Extranet :** Un extranet (ou réseau interne étendu) est un réseau de télécommunications de type internet conçu pour faciliter les échanges entre une organisation sociale et ses correspondants extérieurs.

**Design pattern:** patron de conception en français, il s'agit d'un arrangement caractéristique de modules, reconnu comme bonne pratique en réponse à un problème de conception d'application. Il existe énormément de types de patrons, mais certains se démarquent davantage grâce à leur stabilité et leur utilité.

**OpenSource:** la désignation open source s'applique aux logiciels dont la licence respecte des critères établis par l'Open Source Initiative, i.e. la possibilité de redistribution et d'accès au code source principalement.

**Template:** Un template est un modèle de présentation des données. On parle aussi de «patron» comme en couture ou de gabarit. Ils sont écrits en Twig et contiennent du HTML, il fait partie de la partie graphique.

---

**YML:** Langage de sérialisation de données qui reprend des concepts d'autres langages comme XML, mais en organisant les données sous forme de listes.

**Plugin :** Un plugin, aussi nommé module d'extension, module externe ou add-on, est un paquet qui complète un logiciel hôte pour lui apporter de nouvelles fonctionnalités.

# Table des annexes

Annexe 1 : Vue d'ensemble et tableau de bord.....	38
Annexe 2 : Exemple de liste avec filtres (Projets).....	38
Annexe 3: Exemple de formulaire (Projets).....	39
Annexe 4 : Détails d'un projet.....	39
Annexe 5 : Détails d'un client.....	40
Annexe 6 : Espace document d'un projet.....	40
Annexe 7 : Statistique des projets.....	41
Annexe 8 : Statistique d'un utilisateur.....	41
Annexe 9 : Modification de son compte (Utilisateur).....	42
Annexe 10 : Liste des utilisateurs.....	42
Annexe 11 : Planning Gantt réel.....	43
Annexe 12 : Planning Gantt prévisionnel.....	44
Annexe 13 : UML de la base de donnée.....	45

*NB : Le menu à droite de l'écran et la barre en haut sont présente sur chacune des pages, mais par souci de visibilité elle ne serons montré que dans la première annexe*

## Annexe 1 : Vue d'ensemble et tableau de bord

The screenshot shows the Oziolab Intranet dashboard. On the left, a sidebar lists 'Projets', 'Tickets', 'Clients', 'Administration', 'Documents', and 'Statistiques'. Below this is a 'sonata project' section. The main area displays several cards:

- Rendez vous de la semaine:** 19/06/2015 14:15, Client: Oziolab, Ticket: azerty.
- Mes Tâches (Total : 11):**
  - projet d'eau:** Projet: C'est un projet d'eau, Client: Google, Description: coucou, Temps passé: 12:00, Facturable: oui, Ajouté le: 21/05/15. Buttons: Terminer la tâche, Voir toutes les tâches, Ajouter une tâche.
  - tache pas facturable:** Projet: Pondre un oeuf, Client: Oziolab, Description: Eh ouai, Temps passé: 13:15, Facturable: non, Ajouté le: 21/05/15. Buttons: Terminer la tâche, Voir toutes les tâches, Ajouter une tâche.
  - Tache facturable:** Projet: Trouver les poulemouth, Client: Chicken Run, Description: coucou, Temps passé: 13:30, Facturable: oui, Ajouté le: 21/05/15. Buttons: Terminer la tâche, Voir toutes les tâches, Ajouter une tâche.
- Mes Tickets (Total : 1):** azerty
  - Client: Oziolab, Objets: azerty, Catégorie: Administratif, Crée par: corentin, Technicien: corentin, Rendez vous: 19/06/2015, Ajouté le: 12/06/2015. Buttons: Ticket 'Non traité', Voir toutes les tickets, Ajouter un ticket.
- Derniers Projets (Total : 1):** La ça va marcher
  - Client: Chicken Run, Description: coucou, Temps passé: 00:00, Temps facturable: 00:00, Nombre de tâches restantes: 0, Ajouté le: 09/06/2015. Buttons: Facturer, Terminer le projet, Voir toutes les projets, Ajouter un projet.
- Statistique de temps des sept derniers jours:** Temps total passé 17:30. A bar chart showing time spent in hours for each day.

## Annexe 2 : Exemple de liste avec filtres (Projets)

	Nom	Client	Temps passé	Temps fact.	Temps facturé	Ajouté le	Terminé	Actions
<input type="checkbox"/>	<u>C'est un projet d'eau</u>	Google	09:00	09:00	0€	06/05/15	<span style="color: red;">non</span>	<span style="color: red;">[ ]</span> <span style="color: red;">[ ]</span> <span style="color: red;">[ ]</span> Facturer <span style="color: red;">[ ]</span>
<input type="checkbox"/>	<u>Faire une choucroute</u>	Google	20:00	20:00	648€	24/04/15	<span style="color: red;">non</span>	<span style="color: red;">[ ]</span> <span style="color: red;">[ ]</span> <span style="color: red;">[ ]</span> Facturer <span style="color: red;">[ ]</span>
<input type="checkbox"/>	<u>Une poule chanteuse</u>	Google	42:45	31:30	2800€	17/04/15	<span style="color: red;">non</span>	<span style="color: red;">[ ]</span> <span style="color: red;">[ ]</span> <span style="color: red;">[ ]</span> Facturer <span style="color: red;">[ ]</span>
<input type="checkbox"/>	<u>Faire un poule au verre</u>	Google	07:15	07:15	1150€	17/04/15	<span style="color: red;">non</span>	<span style="color: red;">[ ]</span> <span style="color: red;">[ ]</span> <span style="color: red;">[ ]</span> Facturer <span style="color: red;">[ ]</span>
<input type="checkbox"/> <b>Tous les éléments (4)</b> <span style="border: 1px solid #ccc; padding: 2px;">Supprimer</span> <span style="border: 1px solid #ccc; padding: 2px;">OK</span> <span style="border: 1px solid #ccc; padding: 2px;">Export</span> - 1 / 1 - 4 résultats - Par page <span style="border: 1px solid #ccc; padding: 2px;">25</span>								

**Filtres**

**Nom**:

**Client**:  x ▼

**Ajouté le**: x x x x x x

**Terminé**: non x ▼

Filtrer Effacer

### Annexe 3 : Exemple de formulaire (Projets)

Créer

Liste d'actions ▾

**Projets**

**Nom \***

**Client \***

**Description \***

**Prix Heure \***

**Utilisateurs Ecriture/Lecture sur les documents**

**Utilisateurs Lecture sur les documents**

[Créer](#) [Créer et retourner à la liste](#) [Créer et ajouter](#)

### Annexe 4 : Détails d'un projet

Une poule chanteuse (Google) : [Terminer](#)

Répartition du temps par tâche



#### Informations

Description : The voice

Prix à l'heure : 80€

TEMPS FACTURABLE  
31:30

74% du temps passé est facturable

PRIX FACTURÉ  
2800€

109% du prix facturable a été facturé

Date de création : 17/04/15

Createur : aucun

#### Todo (1)

Priorité	Nom	Terminé	Actions
●	<a href="#">Modifier le machin</a>	<a href="#">Terminer</a>	<a href="#"></a> <a href="#"></a>

[Ajouter un todo](#)

[Voir tous les todos](#)

Tâches en cours

Toutes les tâches

Tout les todos

Tout les documents

#### Liste tâches en cours (2)

Nom	Utilisateur	Description	Ajoutée le	Facturable	Temps total	Terminée	Actions
Crée le site	corentin	Faire le site	18/05/15	<input checked="" type="checkbox"/>	07:45	<a href="#">Terminer</a>	<a href="#"></a> <a href="#"></a>
faire la laverie	corentin	azerty	19/05/15	<input checked="" type="checkbox"/>	03:45	<a href="#">Terminer</a>	<a href="#"></a> <a href="#"></a>

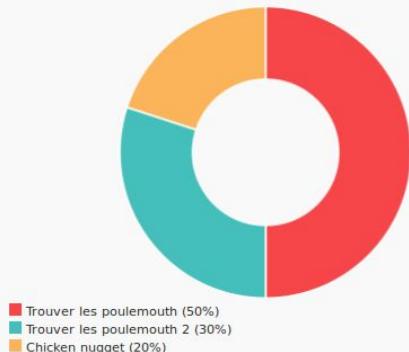
[Ajouter une tâche](#)

[Voir toutes les tâches](#)

## Annexe 5 : Détails d'un client

### Chicken Run (#506) : Actif

Répartition du temps par projet



### Informations

**Prix à l'heure :** 35€  
**Adresse :** 17 rue des poules  
**Ville :** QUIMPER (29000)  
**Tel :** 444719  
**Portable :** 444719  
**Email :** roule@mapoule.com  
**Crée le :** 17/04/15  
**Technicien :** Jean Jacque  
**Notes :** Une poule vend des oeufs

**TEMPS FACTURÉ**  
**32:00**

85% du temps passé facturable a été facturé

Projets en cours    Tout les projets

### Liste projets en cours (2)

Nom	Temps passé	Temps facturable	Prix facturé	Date de création	Terminé	Actions
Trouver les poulemouth	13:30	13:30	831€	30/04/15	<input checked="" type="checkbox"/> Terminer	
Construire un kfc	00:00	00:00	180€	06/05/15	<input checked="" type="checkbox"/> Terminer	

Ajouter un projet

Voir tous les projets

## Annexe 6 : Espace document d'un projet

### Liste des documents de "Faire une choucroute" (Total : 12)

Liste d'actions ▾

Nom d'un document..

42-Un-fichier-d... 4 KB	35-padaboum.p... 4 KB	30-coucou.zip 7 KB	28-Un-autre-tru... 129 KB
27-un-pdf.pdf 96 KB	26-plus-quun.txt 95 Bytes	25-encore-duex... 604 Bytes	24-Des-docume... 61 KB
22-fiml.txt 604 Bytes	21-Un-autre-do... 95 Bytes	19-une-autre-r... 95 Bytes	17-Recette-de-c... 95 Bytes

## Annexe 7 : Statistique des projets

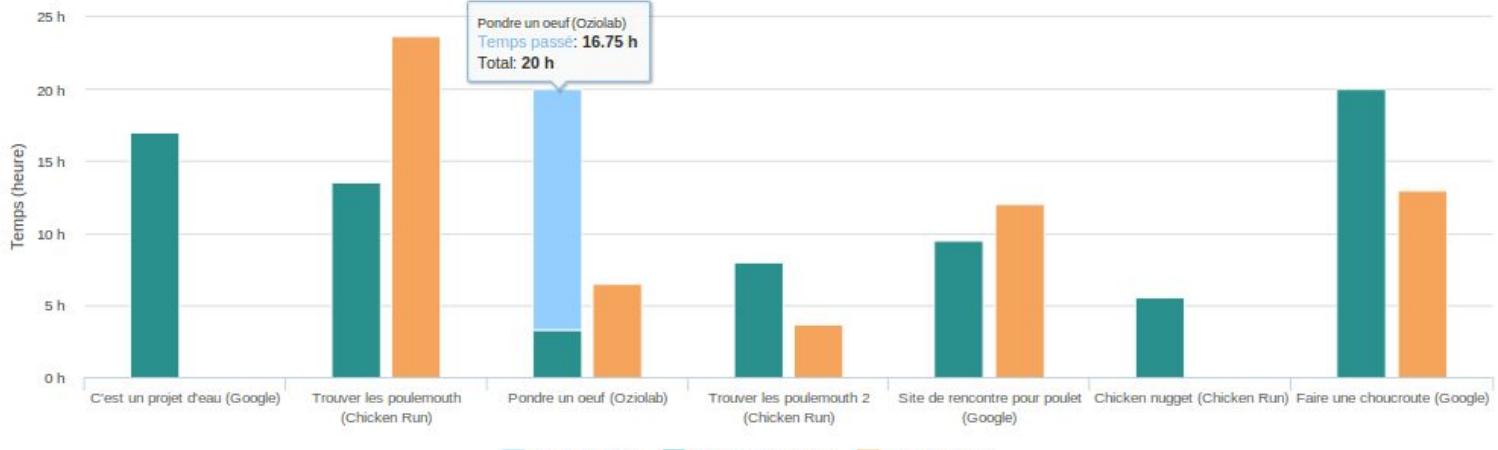
### Statistiques général

17 mai, 2015 - 15 juin, 2015 ▾

Statistiques des projets

Statistiques des utilisateurs

Résumé des projets\*



## Annexe 8 : Statistique d'un utilisateur

### Statistiques général

9 juin, 2015 - 15 juin, 2015 ▾

Statistiques des projets

Statistiques des utilisateurs

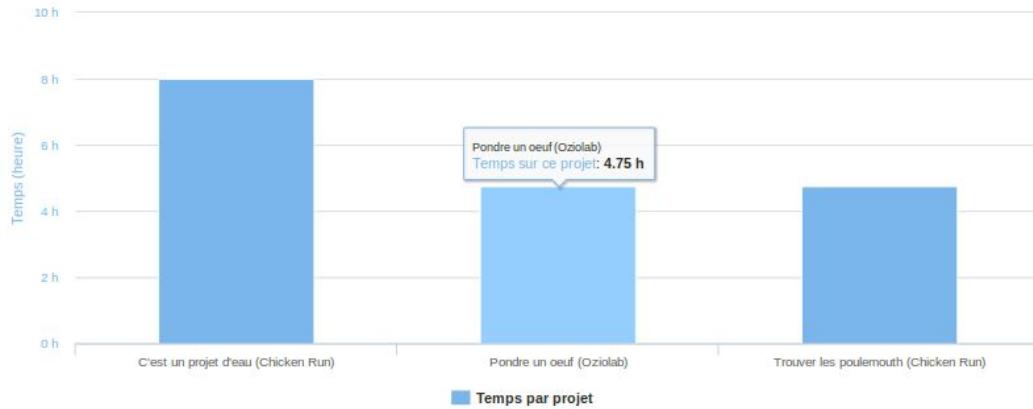
brewal

corentin

michel

unAdmin

Résumé des temps passés sur les projets (17:30)\*



\*Affichage des sept derniers projets avec un temps ajouté lors de cette période

## Annexe 9 : Modification de son compte (Utilisateur)

<b>Informations de connexion</b>		<b>Changer votre mot de passe</b>	
<b>Nom d'utilisateur :</b> **	corentin	<b>Mot de passe actuel :</b> **	*****
<b>Adresse e-mail :</b> **	cq@oziolab.fr	<b>Nouveau mot de passe :</b> **	
<b>Mot de passe actuel :</b> **		<b>Vérification :</b> **	
		<a href="#">Modifier le mot de passe</a>	
<a href="#">Envoyer</a>			

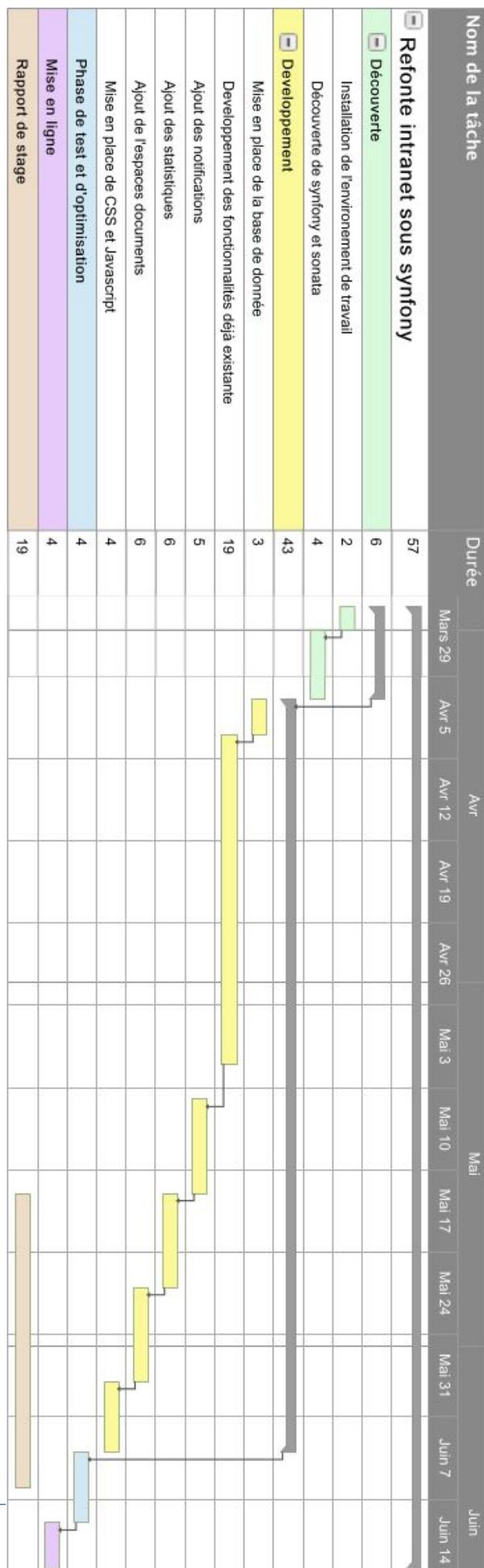
## Annexe 10 : Liste des utilisateurs

<input type="checkbox"/>	<b>Nom d'utilisateur</b>	<b>Adresse email</b>	<b>Groupes</b>	<b>Activé</b>	<b>Verrouillé</b>	<b>Créé le</b>	<b>Changer d'identité</b>
<input type="checkbox"/>	corentin	cq@oziolab.fr	Super_Admin	oui	non	2 avr. 2015 09:50:26	-
<input type="checkbox"/>	Jean Jacque	JJ@oziolab.fr		oui	non	8 avr. 2015 15:33:34	
<input type="checkbox"/>	brewal	br@oziolab.fr	Super_Admin	oui	non	6 mai 2015 16:40:42	
<input type="checkbox"/>	mickey	mickey@oziolab.fr		non	non	7 mai 2015 09:52:57	
<input type="checkbox"/>	michel	mg@oziolab.fr	Super_Admin	oui	non	11 mai 2015 11:06:04	
<input type="checkbox"/>	test	test@test.fr		oui	non	11 mai 2015 11:13:33	
<input type="checkbox"/>	test2	tes@tes.fr		oui	non	11 mai 2015 11:29:02	
<input type="checkbox"/>	unUtilisateur	u@oziolab.fr	Observateur	oui	non	12 mai 2015 09:45:00	
<input type="checkbox"/>	unAdmin	a@oziolab.fr	Administrateur	oui	non	12 mai 2015 17:15:33	
<input type="checkbox"/>	unTicket	t@oziolab.fr	Ticket, Client, Tache, Facturation	oui	non	12 mai 2015 17:39:39	
<input type="checkbox"/> <a href="#">Tous les éléments (10)</a>				<a href="#">Supprimer</a>	<a href="#">OK</a>	<a href="#">Export</a>	- 1 / 1 - 10 résultats - Par page <a href="#">25</a>

## Annexe 11 : Planning Gantt réel



## Annexe 12 : Planning Gantt prévisionnel



## Annexe 15 : UML de la base de donnée

