



ASSOCIATION  
LIBERTIC



ENTREPRISE  
LES POLYPODES

STAGE DE FIN D'ÉTUDE

---

## Rapport de Stage de fin de DUT

---

*Auteur :*  
Thibaud COURTOISON

*Responsables :*  
M. Ronan GUILLOUX  
M. Nicolas HERNANDEZ



UNIVERSITÉ DE NANTES



IUT DE NANTES

Du 13 avril 2015 au 19 juin 2015

## Première partie

# Introduction

## Remerciements

Je tiens tout d'abord à remercier les personnes qui m'ont accompagnées durant ces deux années de DUT.

*M. Ronan Guilloux* et toute l'équipe des Polypodes, pour m'avoir accordé leur confiance en m'acceptant dans leur équipe et pour m'avoir accompagné durant ces deux mois et demi. Leur patience et leur aide ont fait de ce stage une expérience particulièrement constructive.

Et bien sûr, toute l'équipe pédagogique du département informatique de l'IUT de Nantes, qui m'ont fait découvrir les mondes passionnants de l'informatique, de la gestion, du droit et de la communication. Si ces deux années peuvent s'assimiler à une réussite, c'est en très grande partie grâce à eux.

# Resumé du rapport

## Résumé français

Ce document est le rapport de mon expérience de deux mois et demi dans l'élaboration du projet Open Data Event en collaboration avec LiberTIC, une association nantaise militant dans l'Open Data, et Les Polypodes, une agence web promouvant le Logiciel Libre, située sur l'Île de Nantes. Vous trouverez dans un premier temps la présentation du stage et du projet. La suite vous présentera la phase d'Analyse et de Conception que j'ai effectuée. Puis, je décrirai la phase de Développement avec les différentes technologies que j'ai pu aborder. Enfin, une partie présentera les possibilités de suite du projet. Et pour finir, je ferai un bilan du projet, ainsi qu'un bilan personnel.

## English abstract

This document is a report of my two month and a half experience in the making of the Open Data Event project in collaboration with LiberTIC, a Nantes-based organization activist in the Open Data field, and Les Polypodes, a web agency promoting Free Software, located at L'Île de Nantes. First, you will find a presentation of the internship and the projet. The following will present the analysis and design phases I made. Then, I will describe the development stage with different technologies I had the possibility to tackle. Finally, a section will present the probable future of the project. And at the end, I will do a review of the project and a personnal assessment.

## Table des matières

<b>I</b>	<b>Introduction</b>	<b>1</b>
	Remerciements	1
	Resumé	2
	Sommaire	3
<b>1</b>	<b>Le projet</b>	<b>4</b>
1.1	Données Ouvertes et LiberTIC . . . . .	4
1.2	Logiciel Libre et Les Polypodes . . . . .	4
1.3	L'aggrégateur d'événements (ODE) . . . . .	6
1.4	Historique des versions . . . . .	6
1.5	Ma place au sein de ce projet . . . . .	8
<b>II</b>	<b>Développement</b>	<b>9</b>
<b>2</b>	<b>Analyse et Conception</b>	<b>9</b>
2.1	Mise en place de mon environnement de travail . . . . .	9
2.2	Mise à niveau préliminaire . . . . .	9
2.3	Analyse des besoins . . . . .	10
2.4	Conception et première réunion . . . . .	10
<b>3</b>	<b>Développement et Tests</b>	<b>13</b>
3.1	Symfony 2 . . . . .	13
3.2	Intégration de SabreDAV . . . . .	17
3.3	Implémentation avec Elasticsearch . . . . .	19
3.4	Implémentation avec PostgreSQL . . . . .	21
3.5	Fondation d'une API REST . . . . .	21
3.6	Tests unitaires . . . . .	23
3.7	Tests de comportement . . . . .	23

<b>III Conclusion</b>	<b>25</b>
<b>4 En marche vers ODE v2.1</b>	<b>25</b>
4.1 Finir l'API . . . . .	25
4.2 Import / Export . . . . .	25
4.3 Formulaire . . . . .	25
4.4 Brouillons . . . . .	25
4.5 Autres fonctionnalités . . . . .	25
<b>Conclusion</b>	<b>26</b>
Bilan du projet . . . . .	26
Bilan personnel . . . . .	26
<b>Références / Webographie</b>	<b>27</b>
<b>Annexes</b>	<b>28</b>

# 1 Le projet

Une des particularités de mon stage est le fait que je travaillais en collaboration direct non seulement avec l'entreprise qui accompagnait mon stage, mais aussi avec des représentants de l'association LiberTIC, clients du projet. Cette section présentera donc Les Polypodes et LiberTIC, puis, je parlerai du projet, de ses premières versions et enfin de ma place sur ce projet.

## 1.1 Données Ouvertes et LiberTIC

*“L’ouverture des données (en anglais open data) représente à la fois un mouvement, une philosophie d’accès à l’information et une pratique de publication de données librement accessibles et exploitable.”*

*(Wikipédia)*

Il s’agit là d’une définition plutôt abstraite de l’ouverture des données. Mais pour résumer, on pourrait regrouper la définition en trois points :

- **Disponibilité et Accès** : Les données doivent être disponibles dans leurs ensembles et gratuitement et dans un format libre et modifiable.
- **Réutilisation et Redistribution** : Les données doivent être disponibles sous une licence autorisant la réutilisation et la redistribution des données
- **Participation universelle** : Tout le monde doit être en mesure d’utiliser, de réutiliser et de redistribuer sans discrimination contre des personnes ou des groupes. (Exemple : restriction non-commerciale ne fait pas partie de l’open data)

On pourrait aussi définir l’Open Data comme la tendance à considérer l’information publique comme un bien commun dont la diffusion est d’intérêt public et général.

Grâce à des mouvements citoyens comme **Wikimédia France**, **Open Street Map France** ou **LiberTIC**, l’ouverture des données s’est popularisée en France. Ainsi, depuis quelques années, nous pouvons voir l’apparition de plusieurs portails de données de villes, de départements ou de régions (tels que **Paris Data**, **Data Pays de la Loire**, etc...)

**LiberTIC** est une association nantaise, fondée en 2009, de promotion de la Culture Libre et des Données Ouvertes dans l’espace francophone. Elle est particulièrement identifiée comme l’une des principales associations françaises œuvrant pour la promotion de l’Open Data.

LiberTIC publie notamment une carte de France de l’Open Data<sup>[1]</sup> identifiant les projets de collectivités publiques, en cours ou déjà réalisés, de mise à disposition de données publiques transversales.

Durant mon stage, j’ai pu rencontrer plusieurs personnes de l’association, tels que :

**Parler de Loïc et Aleth**

## 1.2 Logiciel Libre et Les Polypodes

*“Un logiciel libre est un logiciel dont l’utilisation, l’étude, la modification et la duplication en vue de sa diffusion sont permises, techniquement et légalement.”*

(Wikipédia)

D'une manière un peu simplifiée, on peut dire qu'un logiciel libre est un logiciel qui peut être **utilisé**, **modifié** et **redistribué** sans restriction par la personne à qui il a été distribué. Un tel logiciel est ainsi susceptible d'être soumis à l'étude, critique et correction. C'est pourquoi les logiciels libres sont d'une certaine fiabilité et réactivité.

Des exemples de logiciels libres célèbres sont **Mozilla Firefox**, **Mozilla Thunderbird**, **OpenOffice** et **VLC**.

La plupart du temps, il existe une confusion entre le développement d'un logiciel libre par une entreprise et le service proposé pour ce logiciel par cette entreprise, ce qui amène de nombreuses personnes à se questionner sur la viabilité d'un Business Model inhérent à la contribution aux logiciels libres. Un des exemples intéressants est celui de **Canonical Ltd.**, le créateur d'Ubuntu. En effet, avec un système d'exploitation libre, mis à jour régulièrement, Canonical propose ses services aux entreprises pour l'installation, la customisation et l'utilisation de leur système d'exploitation. Ainsi, Ubuntu en tant que logiciel libre leur donne une visibilité sur le plan international et une crédibilité dans le milieu des systèmes d'exploitation.

C'est pourquoi de nombreuses entreprises se lancent dans le logiciel libre, que cela soit par la diffusion de leurs propres logiciels sous licences libres ou par le maintien de d'autres logiciels libres.

Enfin, le Logiciel Libre est aussi une philosophie considérant que le meilleur moyen de bénéficier de logiciel à la pointe de la technologie est d'apporter chacun sa pierre à l'édifice du Logiciel Libre.

L'agence " Les Polypodes " fait partie de ceux qui participent au Logiciel Libre, en y contribuant, mais aussi en l'utilisant. Même si l'agence intègre le Logiciel Libre dans son *Business Model*, il ne s'agit pas non plus de sa seule activité. En effet, elle produit aussi des applications pour des clients.

*“Le Polypode commun (*Polypodium vulgare* L.) est une fougère de la famille des Polypodiaceae. Il est parfois appelé réglisse des bois ou réglisse sauvage. En effet, son rhizome a été utilisé à des fins médicinales, mais aussi gastronomiques.”*  
(Wikipédia)



FIGURE 1 – Partie des bureaux chez Les Polypodes

**Les Polypodes** est une agence web créé en 2005 par *M. Antonio-Manuel FIDALGO* et qui emploie actuellement 8 salariés. Les bureaux de l'entreprise sont situés au 7<sup>ème</sup> étage de l'immeuble *Sigma 2000*.

Chez Les Polypodes, le Logiciel Libre est le choix par défaut, et les développements sont créés et livrés à priori avec une licence libre MIT ou GNU/GPL.

De plus, l'équipe essaye de privilégier les outils et les frameworks libres, sans pour autant se fermer aux opportunités.

### 1.3 L'aggrégateur d'événements (ODE)

L'aggrégateur d'événement, appelé **Open Data Event** (ODE), est une plateforme permettant de relier des fournisseurs de d'événements (les organisateurs de festivals, concerts, conférences, etc...) avec les réutilisateurs d'événements (les utilisateurs de l'API, les journaux, etc...). Le but principal de cette association est d'améliorer la diffusion des données.

Sachant que j'ai déjà parlé de ce qu'était ODE dans l'introduction, je devrais plutôt parler d'où vient l'idée et quel était la première vision d'ODE

### 1.4 Historique des versions

ODE est un projet mené par LiberTIC commencé au début de 2014. Il est passé par plusieurs versions, traversant différentes phases d'analyses, de conceptions et de développements.

#### La première réunion

L'élément déclencheur de ce projet fut la réunion du 11 juin 2014 à Stéréolux avec l'association LiberTIC et quelques acteurs de l'événementiel à Nantes. Il est apparu qu'il y avait une nécessité d'améliorer la diffusion des données, librement et gratuitement.



C'est ainsi qu'est né le projet ODE (Open Data Event). Il s'agit d'un agrégateur d'événements permettant de relier les fournisseurs d'événements (les organisateurs de festivals, concerts, conférences, etc...) avec les réutilisateurs d'événements (les utilisateurs de l'API, les journaux, etc...).

Cette réunion a permis de produire un cahier des charges pour le projet (disponible à cette adresse : [https://github.com/LiberTIC/ODEV2/blob/master/doc/Documents/120622\\_ODE\\_cahierDesCharges\\_MakinaCorpus.pdf](https://github.com/LiberTIC/ODEV2/blob/master/doc/Documents/120622_ODE_cahierDesCharges_MakinaCorpus.pdf) )

## Version 1 - Makina Corpus

Makina Corpus ( <http://makina-corpus.com/> ), entreprise de développement de logiciels libres, a répondu au cahier des charges en proposant de réaliser le projet. Plusieurs mois plus tard, la première version était finie et fut présentée à des responsables de l'association LiberTIC.

Cependant, durant la réunion, il est apparu de nombreuses différences dans la direction du projet mené par Makina Corpus avec la direction voulu par l'association. A la fin de la réunion, il a été décidé de ne pas continuer la relation entre l'association et l'entreprise.

L'application développée par Makina Corpus est programmée en Python. Il s'agit d'une API REST qui permet aux clients d'interagir avec les événements. Le code est basé sur le **Pyramid web framework** et **Cornice**.

Le problème de cette application est qu'elle ne répond pas aux attentes de LiberTIC par rapport à l'accessibilité et la facilité d'utilisation.

De plus, il manquait à LiberTIC une maîtrise d'ouvrage technique pour assurer le succès du projet tout au long de sa réalisation.

## La reprise par Les Polypodes

Suite à une réunion de l'association LiberTIC abordant l'échec de la première version, Ronan Guilloux de l'entreprise Les Polypodes, présent à cette réunion, fit la proposition suivante : Les Polypodes serait prêt à encadrer des stagiaires pour relancer le projet. Les stagiaires seraient accueillis par l'entreprise et l'association s'engagerait à faire un suivi plus approfondi du projet pour éviter le même résultat que la version précédente.

## Version 2 - Les premiers stagiaires

En janvier 2015, deux stagiaires de BTS chez Les Polypodes ont commencé à travailler sur ODE version 2.

Leur travail a été de produire un prototype d'une application sous Symfony2 permettant une gestion des événements avec un serveur CalDAV. Ils ont fait un comparatif des serveurs CalDAV et fait une analyse des sémantiques disponibles pour les événements.

Sachant que la durée de leur stage n'était que de six semaines, ils n'ont pas pu faire évoluer le prototype ( disponible à cette adresse : <https://github.com/polypodes/CalDAVClientPrototype> )

Les stagiaires de BTS avaient pour mission de réaliser un panel comparatif des serveurs CalDAV existant, et de faire un " proof of concept " validant le choix d'un serveur CalDAV pour stocker les données calendaires (date de début/fin, description, etc.) mais aussi " métiers " (lieu, prix, organisateur, etc.), notamment les méta-données quit étendent le format

iCalendar (les champs non-standard **X-ODE-**). Ils ont également exploré et évalué les différentes bibliothèques open-source permettant de manipuler des données au format iCalendar avec le langage PHP.

## Version 2 - Mon stage

Depuis le 13 avril 2015, je travaille donc sur la version 2 du projet. Le but est de pouvoir réaliser le projet dans le temps qui m'est imparti ( 10 semaines ).

### 1.5 Ma place au sein de ce projet

Avant de commencer mon stage, le projet avait déjà évolué plusieurs fois et il m'a fallu analyser les précédentes versions, par le code, les documentations et les comptes-rendu de réunion, pour comprendre les choix techniques précédents et pouvoir effectuer au mieux les tâches qui m'ont été assignées.

Pour mener à bien mon stage, j'ai pu profiter du cadre de l'entreprise des Polypodes pour effectuer une gestion de projet. En effet, j'ai établi, avec M.Guilloux, des " Sprints " d'une semaine décrivant la liste des tâches à effectuer, et j'ai préparé les réunions avec LiberTIC en fin de sprint avec démo de l'avancement du Sprint et définition de la roadmap pour la suite. De plus, nous avons effectué des " Daily scrum " tout les matins chez Les Polypodes. Il s'agit globalement d'une petite réunion entre toute l'équipe où nous répondons aux trois questions suivantes : " *Qu'est ce que j'ai fait hier pour atteindre les objectifs ?* ", " *Que vais-je faire aujourd'hui pour atteindre les objectifs ?* " et " *Est ce que je vois des problèmes qui pourraient m'empêcher d'atteindre mes objectifs ?* ".

De plus, pour la gestion de projet, j'ai utilisé **Trello** qui permet de gérer toutes les tâches à effectuer et **Framadate** pour l'organisation des réunions.

Enfin, après chaque réunion avec LiberTIC, j'ai rédigé un rapport de réunion dans lequel je précisais les éléments évoqués et la roadmap pour la suite. Pour suivre la voie du Logiciel Libre, les rapports sont disponibles sur le Github aux côtés de la documentation du projet.

## Deuxième partie

# Développement

## 2 Analyse et Conception

Comme tout projets de développement, il faut commencer par une phase d'analyse et de conception. Le projet Open Data Event a comme particularité d'avoir déjà eu plusieurs versions avant que je ne commence à travailler dessus. C'est pourquoi la partie d'analyse des versions précédentes était très importante, comme vous pourrez le lire par la suite.

### 2.1 Mise en place de mon environnement de travail

Il a été mis à ma disposition un **iMac 21"** avec un second écran de la même taille. J'ai donc pu découvrir l'environnement de développement offert par **OS X**. Globalement, la connaissance de l'environnement **Linux** m'aura aidé tout au long de l'utilisation d'**OS X**.

Après m'avoir créé un compte administrateur sur l'ordinateur fourni, j'ai pu installer tous les logiciels nécessaires à la réalisation du projet. Pour des raisons personnelles, j'ai choisi d'utiliser les outils suivants :

- **iTerm** (Version 2.0.0)
- **Sublime Text 2** (Version 2.0.2)
- **Google Chrome** (Version 43)
- **Wireshark** (Version 1.12.5)
- **Apple Calendar** (Version 7.0)
- **PDFLatex** (Version 3.14)

Pour l'utilisation de serveurs SabreDAV, Baïkal, Elasticsearch et d'autres, j'ai installé une machine virtuelle avec **Vagrant**.

**LiberTIC** étant une association militant pour les licences libres, il était évident que la majorité de mon travail (qu'il s'agisse de développement ou de compte-rendus) soit mis lui aussi sous licence libre. Ainsi, il est possible de retrouver mon travail sur **GitHub** à l'adresse suivante : <https://github.com/LiberTIC/ODEV2>.

En plus de mon environnement local, j'ai pu accéder à un serveur de pré-production hébergé par OVH.

### 2.2 Mise à niveau préliminaire

Dès le début de mon stage, il a été défini que le projet allait se construire sur des technologies que je ne connaissais pas et/ou ne maîtrisais pas. C'est pourquoi il a été convenu que la première semaine de mon stage serait consacrée à une mise à niveau pour différentes technologies.

Premièrement, j'ai suivi un récapitulatif des commandes **Git** <sup>[2]</sup> puis lu un article sur les bonnes pratiques de l'utilisation de Git et de ses branches <sup>[3]</sup>.

Ensuite, j'ai lu plusieurs articles sur les bonnes pratiques du développement **PHP** <sup>[4]</sup> <sup>[5]</sup>.

Enfin, j'ai lu et appliqué la totalité du **Symfony Book** <sup>[6]</sup>. Il s'agit du document de référence dans l'apprentissage du Framework. Ce document est mis à jour très régulièrement et permet d'obtenir une grande partie des informations nécessaires pour se lancer dans le

développement d'une application Symfony 2.

## 2.3 Analyse des besoins

### Service en ligne

Une volonté de la part de LiberTIC était de faire d'ODE un service disponible en ligne. Sachant que j'ai effectué mon stage en partenariat chez Les Polypodes et que Symfony 2 est la Stack la mieux maîtrisée par l'équipe, c'était le choix le plus logique pour pouvoir être accompagné par toute l'équipe en cas de besoins.

Symfony 2 est un framework PHP proposant un grand nombre de composants facilitant et accélérant le travail du développeur PHP.

### CMS adapté

Un des besoins les plus importants de la version 2 était de pouvoir proposer un Content Manager System accessible, dont même une personne ne connaissant rien à l'informatique pourrait ajouter du contenu.

### API REST

Sachant qu'une partie importante de l'Open Data consiste en la réutilisation des données, il parut importante de proposer une API REST pour récupérer les événements.

### Compatibilité

Un des souhaits de la part d'ODE était de rendre compatible l'application avec d'autres clients calendriers. J'en parlerai plus tard avec l'introduction de CalDAV.

## 2.4 Conception et première réunion

Une semaine après le début de mon stage, j'ai pu rencontrer deux personnes de l'association durant une réunion où nous avons discuté ensemble du périmètre du projet ainsi que ce que je proposais pour l'architecture.

### Diagramme de cas d'utilisation

Suite à la conception, j'ai fait un diagramme de cas d'utilisation de l'application. Il est possible de le retrouver dans les Annexes à la dernière page de ce rapport.

### CalDAV et WebDAV

Dès la première réunion, il était clair qu'il fallait se baser sur un protocole déjà en place, pour ne pas "réinventer la roue". Le seul protocole actuellement en place répondant à nos besoins est **CalDAV**. Il s'agit d'une extension du protocole WebDAV, lui-même une extension du protocole HTTP (plus d'informations en Annexe 1).

CalDAV définit la gestion des calendriers au sein d'un serveur. Il décrit aussi la façon dont les serveurs CalDAV doivent gérer les événements et les utilisateurs.

CalDAV est défini par la RFC 4791<sup>[12]</sup> datant de 2007. Il existe plusieurs implémentations de cette RFC. La première, historiquement parlant, est CalendarServer, en python.


### **oEmbed**

Durant la réunion, nous avons discuté de la façon dont ODE devrait gérer l'ajout de médias à un événement. Après s'être mis d'accord sur le fait que gérer l'upload d'images, de vidéos ou autres, ne serait pas intéressant, M. Ségalou a évoqué l'utilisation d'oEmbed (<http://www.oembed.com/>).


oEmbed est un format permettant d'obtenir des informations relatives à un média. Globalement, il suffit de fournir une URL appartenant à un service compatible oEmbed (Youtube, Flickr, Vimeo, DailyMotion, SoundCloud, etc.) (Figure X). Ensuite, grâce à l'URL, il sera possible de récupérer des informations le concernant (Figure X). Pour enfin l'afficher directement sur le site (Figure X).

## Contacts et medias


Ajouter url

<https://youtu.be/aRljVackZ08>

Supprimer

<https://soundcloud.com/shakaponkofficial>

Supprimer

<https://www.flickr.com/photos/zloran/6916140000/>

Supprimer

Les urls doivent être compatible **oEmbed**

FIGURE 2 – Partie du formulaire pour ajouter des urls de médias

```
{
  "title": "Shaka_Ponk_-_Wanna_Get_Free",
  "width": 480,
  "height": 270,
  "thumbnail_height": 360,
  "html": "\u003ciframe_width=\"480\"_height=\"270\"_src=\"https://\nwww.youtube.com/embed/aRljVackZ08?feature=oembed\"_frameborder=\"0\" allowfullscreen\u003e\u003c/iframe\u003e",
  "thumbnail_width": 480,
  "author_url": "http://www.youtube.com/user/ShakaPonkVEVO",
  "provider_name": "YouTube",
  "version": "1.0",
  "type": "video",
  "provider_url": "http://www.youtube.com/",
  "author_name": "ShakaPonkVEVO",
  "thumbnail_url": "https://i.ytimg.com/vi/aRljVackZ08/hqdefault.jpg"
}
```

FIGURE 3 – Données récupérées depuis youtube grâce au format oEmbed

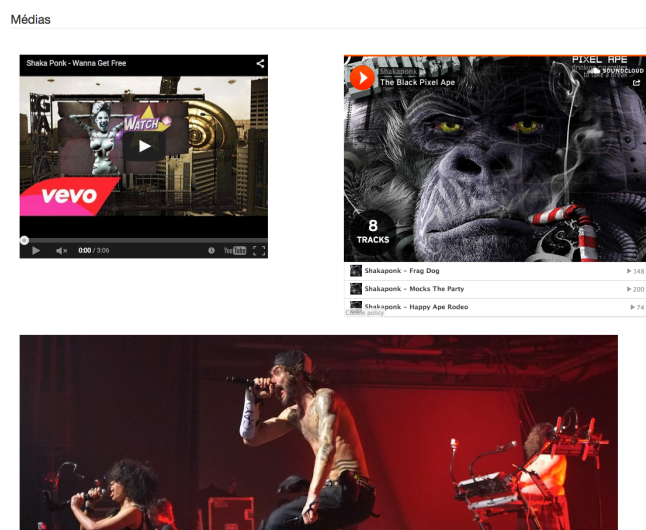


FIGURE 4 – Médias affichés directement sur le site

### 3 Développement et Tests

Après la phase de Conception, vient la phase de développement, suivie de la phase de test.

#### 3.1 Symfony 2

Comme expliqué précédemment, pour ce projet, j'ai utilisé **Symfony 2**. Il s'agit d'un framework PHP suivant le pattern MVC proposant de nombreux composants pour accélérer et faciliter le développement d'une application PHP. De plus, Symfony possède une communauté très impliquée, ce qui permet de trouver de très nombreuses documentations très bien rédigées.

Symfony est sponsorisé par SensioLabs, une entreprise française. Sponsorisé car, même si cette entreprise a développé la première version du framework, elle est maintenant entretenue par la communauté, en tant que projet Open-Source (*MIT License*). Il est actuellement dans sa version 2.7.

Symfony offre la possibilité d'ajouter de nombreux "Bundles" indépendants permettant d'ajouter des fonctionnalités adaptées à nos besoins. De plus, Symfony utilise **Composer** qui est un "Dependency Manager" (ou gestionnaire de dépendance en français) qui permet d'ajouter un Bundle au projet en une seule commande : `composer require "genemu/form-bundle"`

Symfony propose de construire une application selon la structure suivante :

```
- app/
  - cache/
  - config/          /* Tout les fichiers de configuration
                      avec les paramètres et les routes */
  - logs/
  - Resources/       // Les fichiers de ressources (Templates html)
  - AppKernel.php    // Base de l'application, enregistre les bundles
  - ...

- bin/              // Géré par Symfony, pas toucher

- doc/              // La documentation de l'application

- src/              // La plupart de notre code est ici
  - AppBundle       // Bundle principal de l'application

  - [...]           // Plus de détails à suivre

- vendor/           /* Ici se trouve les bundles externes */

- web/              /* Ici, nous trouvons tout les fichiers css,
                      js, les images, etc... Il s'agit du dossier
                      qui sera accessible aux utilisateurs
*/

- composer.json     /* Ce fichier liste tous les bundles que
                      nous utilisons */
```

FIGURE 5 – Architecture proposée par le framework Symfony

## Les composants Symfony

Un des composants que j'ai le plus utilisé dans Symfony est : " HTTPFoundation ". En effet, il gère de lui même les requêtes récupérer pour les offrir dans une classe **Request** et il est possible de créer une réponse très simplement en utilisant la classe **Response**.

```
<?php
[...]
```

```
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpFoundation\Request;
[...]
```

```
function indexAction(Request $request) {

    $name = $request->query->get('name');

    return new Response("Hello".$name);
}
[...]
```

FIGURE 6 – Code simplifié avec exemple d'HTTPFoundation

Comme on peut le voir sur les quelques lignes de code précédentes, il est facile de saluer un utilisateur nous donnant son nom. Si nous voulions faire une page privée, nous pourrions renvoyer un code **403 Forbidden** simplement en ajoutant 403 en argument du constructeur de Response.

Durant le développement d'ODE, j'ai utilisé de nombreux autres composants de Symfony : *Controller*, *FormBuilder*, *Routing*, etc...

Pour revenir sur l'architecture proposée par Symfony (Figure 5), nous pouvions trouver le dossier AppBundle qui est prévu pour y déposer la majorité de notre code.

Change number figure

Il est conseillé de mettre son code selon l'architecture suivante :

## Backend

Dans le dossier **Backend**/, j'ai déposé les quelques classes permettant de stocker et de retrouver des données dans la base données : *CalDAV/Auth.php*, *CalDAV/Principals.php*, *CalDAV/Calendar.php* et *Users/UserManager.php*.

Les trois premières classes étant liées à SabreDAV, j'en parlerai dans la section suivante. La dernière, **UserManager**, est, quant à elle, liée à **FOSUserBundle** ; il s'agit d'un bundle permettant une gestion simplifiée des utilisateurs. Il gère, entre autres, l'inscription, la connexion, le chiffage des mots de passes, les sessions liées à l'utilisateur, les rôles, etc...



## Controller

Dans ce dossier, j'ai déposé les classes de mes Contrôleurs (*sachant que Symfony respecte le design pattern MVC*).

La première classe, **DefaultController** est le contrôleur le plus léger (2 fonctions) : `indexAction` pour afficher la page d'accueil du site et `testAction` qui m'a servi à écrire quelques lignes de php pour tester rapidement mon code.

La seconde classe, **CalDAVController** est le contrôleur permettant de faire le lien avec SabreDAV (plus d'explication dans la section 3.2).

La troisième, **BrowserController** est le contrôleur définissant le Front-End de l'application. Il gère les pages et les formulaires proposés à l'utilisateur. Il est notamment en lien avec `FOSUserBundle` pour intégrer la connexion et l'inscription des utilisateurs.

Enfin, **ApiController** est le contrôleur délivrant l'API. Il s'agit là de la partie de l'application délivrant la promesse de l'**Open Data**.

## Entity

Le dossier **entity/** est celui qui contient le **modèle** de l'application.

Plus de truc

## Form

Ce dossier permet de gérer la création des formulaires en lien avec le modèle. Pour proposer des formulaires ergonomiques, j'ai utilisé un bundle, **MopaBootstrapBundle**, qui permettait d'inclure **Bootstrap** dans l'application, notamment dans la création des champs des formulaires.

## Ajouter un événement

### (\*) - Champs obligatoires

Les champs marqués d'une étoile (\*) sont obligatoires

### Accéder aux calendriers

Pour accéder à vos calendriers, cliquez ici: [Liste des calendriers](#)

### oEmbed

[oEmbed](#) est un format permettant d'obtenir des informations relative à un média.

Globalement, il vous suffit de fournir une url appartenant à un des services suivants:

- Youtube
- Flickr
- Vimeo
- DailyMotion
- SoundCloud
- ...

La liste plus complète est accessible [ici](#).

#### Description

**Titre (\*)**

**Description (\*)**

**Catégorie (\*)**

**Calendrier (\*)**

**Tags**

#### Dates

**Date de début (\*)**

**Date de fin (\*)**

**Lieu** +

**Organisation** +

**Contacts et medias** +

**Urls medias**  Les urls doivent être compatible [oEmbed](#)

**Nom du contact**

**Email du contact**

**Tarifs** +

FIGURE 7 – Formulaire de création d'un événement

## Resources

Ce dossier est en lien avec le dossier *app/Resource/*. En effet, il possède aussi des fichiers de configuration, de routes, mais aussi les librairies js et css à inclure dans le dossier *web/*. Pour les besoins de l'application, j'ai écrit un fichier Javascript simple pour afficher un calendrier avec une coloration sur les dates de l'événement.

### Dates

Juin 2015						
L	M	M	J	V	S	D
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					

Du 20/06/2015 à 00:00 au 21/06/2015 à 00:00

FIGURE 8 – Calendrier généré avec javascript

## Service

Un service (dans Symfony) est un composant, un module, qui remplit une fonction et qui peut être injecté dans le conteneur de dépendance pour être rendu accessible dans toute l'application. Cette fonction peut être très simple : envoyer des e-mails, vérifier qu'un texte n'est pas un spam, etc. Un service est donc un objet PHP qui a pour vocation d'être accessible depuis n'importe où dans le code. Pour chaque fonctionnalité dont je peux avoir besoin dans mon application, je peux créer un ou plusieurs services.

Symfony utilise un "ServiceContainer" qui enregistre tout les services. Il est ainsi possible d'accéder aux services depuis les contrôleurs facilement :

```
<?php
[...]
```

```
function indexAction(Request $request) {

    $slug = $this->get('slugify')->slugify("ExEmPlE_èéàó_à");

    return new Response($slug);

    /* Le code précédent renverra la chaine de caractère suivante:
       exemple-eeao-a
    */
}
```

```
[...]
```

FIGURE 9 – Code simplifié avec exemple d'un Service

Le code précédent est une démonstration de l'appel au service **Slugify**, permettant de générer des slug "URL Friendly".

## 3.2 Intégration de SabreDAV

Après avoir décidé d'utiliser le protocole CalDAV pour le projet, j'ai cherché une librairie PHP permettant d'implémenter CalDAV dans Symfony. Malheureusement, il n'existe pas de Bundle tout prêt pour cette utilisation. Cependant, durant ma recherche, j'ai découvert **SabreDAV**. Il s'agit d'un serveur php capable de gérer les requêtes "PROPPATCH", "MK-CALENDAR", etc. propres à CalDAV.

SabreDAV est un projet Open Source développé par Fruux. Fruux est une entreprise éditrice qui fait partie du **Calendaring and Scheduling Consortium**, discutant sur le standard CalDAV, au côté d'Apple, Google, IBM et d'autres.

SabreDAV possède sa propre architecture, il a donc fallu l'intégrer dans l'application. Pour cela, j'ai créé un Contrôleur spécialement pour la partie CalDAV (CalDAVController.php). Au niveau des routes, j'ai les ai toutes redirigées vers la méthode "indexAction" du contrôleur. Ainsi, j'ai pu transmettre la requête directement au système de SabreDAV.

Le problème que j'ai rencontré par la suite était le fait que SabreDAV gérait lui-même la construction et l'envoi de la réponse, ce qui perturbait le fonctionnement de Symfony.

Grâce à de nombreuses recherches sur le sujet, j'ai découvert que Symfony proposait

une classe, **StreamedResponse**, permettant de gérer ce cas de figure. En effet, **StreamedResponse** prend en argument une fonction de callback dans laquelle la réponse envoyée par SabreDAV sera attrapée par Symfony puis renvoyée vers l'utilisateur. Cela permet aussi de logger les messages renvoyés, ce qui m'a été très utile tout au long du développement de l'application.

```
<?php
[...]
```

```
    public function indexAction(Request $request)
    {
        [...]

        $server = new CalDAVServer();

        $callback = function () use ($server, $request) {

            [...]

            $server->exec();

            /* These two lines log the request and the response */
            $responseBody = $server->httpResponse->getBodyAsString();
            $this->logIt($request, $server->httpResponse, $responseBody);
        };

        return new StreamedResponse($callback);
    }
[...]
```

FIGURE 10 – Code simplifié avec exemple de **StreamedResponse**

Niveau Backend, SabreDAV a été conçu pour laisser le choix aux utilisateurs. En effet, la librairie utilise des interfaces pour les classes de Backend, ainsi, j'ai pu implémenter moi-même les classes, d'abord avec ElasticSearch, puis avec PostgreSQL.

Les trois classes de Backend que j'ai implémentées sont les suivantes :

**Auth.php** : Celle-ci permet d'authentifier l'utilisateur se connectant depuis un client compatible CalDAV.

**Principals.php** : Cette classe permet de gérer l'accès, la création et la suppression des "Principals". Il s'agit d'un concept propre à CalDAV, que l'on pourrait résumer par un dossier pour gérer les différents calendriers.

**Calendar.php** : C'est la classe la plus importante du Backend. En effet, elle rassemble toutes les fonctions pour créer, récupérer, modifier et supprimer un calendrier ou un événement. Elle s'occupe aussi des souscriptions et de lister les calendriers présents sur le serveur. J'utilise cette classe de nombreuses fois dans les contrôleurs, car même si elle est utilisée par SabreDAV, elle s'adapte très facilement à n'importe quelle utilisation.

### 3.3 Implémentation avec Elasticsearch

**ElasticSearch** est un moteur d'indexation et de recherche. Les principaux atouts d'ElasticSearch sont sa capacité à stocker les données sous le format JSON, ainsi que sa capacité à offrir de nombreux graphiques et statistiques sur l'utilisation du service.

ElasticSearch se base sur Apache Lucene, il est donc propulsé par Java. L'installation d'ElasticSearch requiert de plus la version 1.8 de la Java Virtual Machine. ElasticSearch n'étant pas adapté à OSX, j'ai dû installer une machine virtuelle sur mon Mac. Pour cela, j'ai utilisé **Vagrant**.

**Vagrant** est un outil permettant de gérer des machines virtuelles simplement. Il utilise lui même le logiciel très connu : **VirtualBox**. Pour configurer la machine virtuelle, il suffit de modifier un fichier de configuration simple.

Après avoir installé Vagrant, il suffit d'exécuter les commandes suivantes :

```
vagrant init hashicorp/precise32
vagrant up
vagrant ssh
```

Et on se retrouve dans une machine virtuelle avec Ubuntu 14.04.

En ce qui concerne l'installation d'ElasticSearch sur la machine virtuelle, j'ai produit la [Documentation de l'installation](#) sur le Github pour que d'autres puissent le refaire facilement.

Pour faire le lien entre Symfony et le serveur ElasticSearch, j'ai créé la classe ESManager permettant de faire des requêtes de façon simple depuis les contrôleurs. De plus, je l'ai ajouté au ServiceContainer de Symfony pour l'utiliser facilement.

Un des problèmes rencontrés avec ElasticSearch est le fait qu'ElasticSearch est fait pour l'indexation de données, mais il ne donne pas suffisamment de garantie en tant que moteur de persistance de données.

Un autre problème venait l'utilisation que j'en ai fait : le stockage complet des données, posait des problèmes notamment pour l'indexation du jCal d'un événement.

En effet, ElasticSearch se décrit comme " SchemaLess ". Cela veut dire que quel que soit le **document** (*structure JSON*) que nous allons **indexer**, ElasticSearch va produire un document de **Mapping** qui décrit le Schema du document que nous venons de lui donner sous la forme : " Nom => Type ". Il s'agit là d'une fonctionnalité très appréciée dans l'utilisation d'ElasticSearch dans la plupart des cas. Cependant, jCal suit une architecture différente de JSON comme le montre la figure suivante, et je devais donc modifier la façon dont je stockais le jCal pour l'adapter à ElasticSearch.

```
// JSON Classique

{
  "glossary": {
    "title": "example_glossary",
    "GlossDiv": {
      "title": "S",
      "GlossList": {
        "GlossEntry": {
          "ID": "SGML",
          "SortAs": "SGML",
          "GlossTerm": "Standard_Generalized_Markup_Language",
          "Acronym": "SGML",
        }
      }
    }
  }
  [...]
}
```

FIGURE 11 – Exemple JSON

```
// jCal

[ "vevent",
  [
    [ "dtstamp", {}, "date-time", "2006-02-06T00:11:21Z" ],
    [ "dtstart",
      { "tzid": "US/Eastern" },
      "date-time",
      "2006-01-02T12:00:00"
    ],
    [ "duration", {}, "duration", "PT1H" ],
    [ "rrule", {}, "recur", { "freq": "DAILY", "count": 5 } ],
    [ "rdate",
      { "tzid": "US/Eastern" },
      "period",
      "2006-01-02T15:00:00/PT2H"
    ],
    [ "summary", {}, "text", "Event_#2" ],
    [ "description",
      [...]
    ]
  ]
]
```

FIGURE 12 – Exemple jCal

### 3.4 Implémentation avec PostgreSQL

Finalement, durant une réunion hebdomadaire, je me suis rendu compte que le projet avait légèrement évolué en ce qui concerne son utilisation. En effet, la façon dont j'avais conçu ODE était la suivante : Une instance unique du projet avec un grand nombre de données. Or, petit à petit, ODE s'adaptait mieux au format : Un grand nombre de petites instances du projet.

J'ai donc ré-évalué la question du stockage des données, et je me suis rendu compte qu'ElasticSearch, bien qu'adapté pour l'instance unique, n'était pas du tout adapté pour la façon dont nous envisagions ODE à ce moment. Ainsi, j'ai décidé, avec l'accord de mon maître de stage, de remplacer l'utilisation d'ElasticSearch pour le remplacer par une base de données PostgreSQL.

À partir de là, j'ai ré-implémenté les classes d'implémentation du BackEnd SabreDAV pour les adapter à PostgreSQL.

Pour communiquer entre l'application et PostgreSQL, j'ai utilisé **Pomm**. Il s'agit d'un framework open source d'accès à la base de données lié avec un Object Model Manager dédié à PostgreSQL. Il offre une approche alternative à ORM pour l'utilisation des bases de données dans le développement web. Pomm est développé par Grégoire Hubert sous licence Creative Commons BY-SA.

À l'instar de la création d'un service ESManager pour utiliser facilement ElasticSearch, j'ai créé le service PommManager possédant des fonctions telles que : createOne, deleteOne, findOne, findById, etc.

Le problème principal de mon utilisation de PostgreSQL fut le manque du module PostgreSQL de PHP natif sur OSX. Cependant, je n'ai pas rencontré d'autre problème particulier.

### 3.5 Fondation d'une API REST

Créer une API **REST** n'est pas simple. En effet, il ne s'agit ni d'un standard, ni d'un protocole. C'est pourquoi je me suis longuement documenté sur les principes de REST, sur ses bonnes pratiques, mais aussi sur ses limites. [\[7\]](#)[\[8\]](#)[\[9\]](#)

Un des points à savoir, est la progression constante de la part d'API JSON face aux API XML, comme le montre la figure suivante :

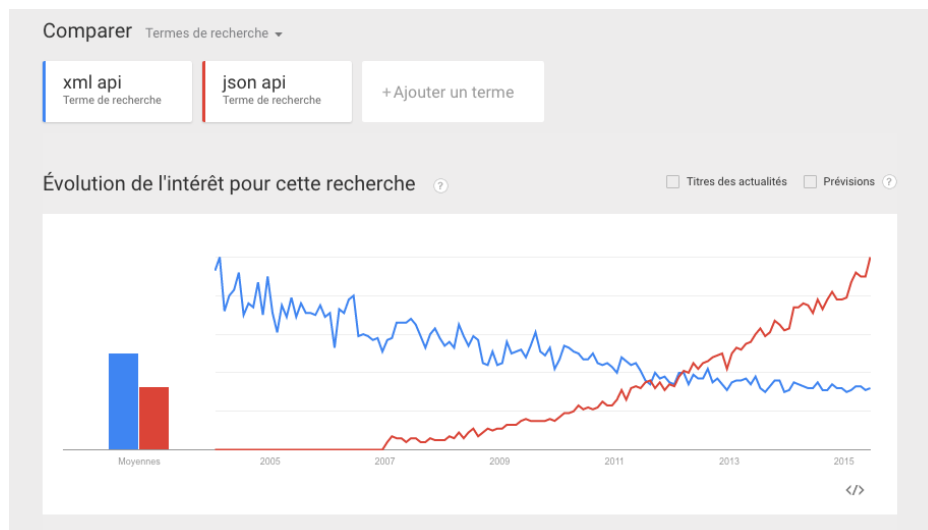


FIGURE 13 – Graphique de recherche des termes XML API et JSON API

Source : *Google Trends*

Durant cette recherche, j'ai découvert la notion de HATEOAS (*Hypermedia As The Engine Of Application State*). Il s'agit d'un concept d'architecture de l'application pour rendre l'API " Explorable ", c'est-à-dire qu'il est possible de se déplacer dans l'API avec des " liens " pour découvrir les opérations disponibles.

HATEOAS est décrit dans le Modèle de Maturité de Richardson <sup>[10]</sup> comme étant le troisième niveau de maturité d'une application REST, les deux premiers étant les Ressources et les Verbes HTTP. Il apporte une dimension sémantique aux API.

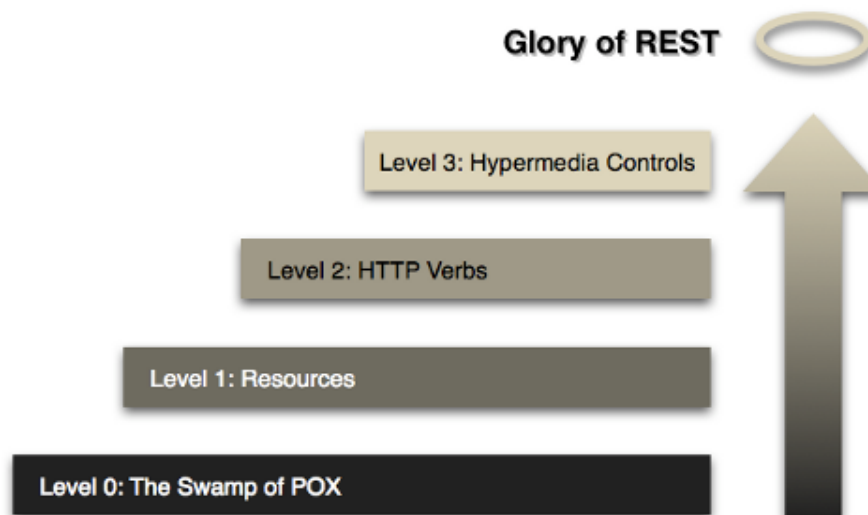


FIGURE 14 – Graphique de recherche des termes XML API et JSON API

Sachant qu'une des composantes principales de ce projet est l'Open Data, l'API REST doit permettre de fournir toutes les données de l'application.

J'ai documenté toutes les opérations possibles de l'API sur Github à l'adresse suivante :



<https://github.com/LiberTIC/ODEV2/blob/master/doc/RestAPI.md>.

J'ai essayé de retourner le plus souvent les bons codes de status HTTP. (*Même si je n'ai pas réussi à mettre le 418 - I'm a teapot...*)

### 3.6 Tests unitaires

Comme pour toutes applications, il est important d'effectuer des tests pour s'assurer de la fiabilité de notre code. Symfony 2 intègre un composant de Tests, c'est donc facile d'écrire les tests unitaires pour l'application.

```
<?php

namespace AppBundle\Tests\Controller;

use Symfony\Bundle\FrameworkBundle\Test\WebTestCase;

class DefaultControllerTest extends WebTestCase
{
    public function testIndex()
    {
        $client = static::createClient();

        $crawler = $client->request('GET', '/app/example');

        $this->assertEquals(200, $client->getResponse()->getStatusCode());
        $this->assertTrue(
            $crawler->filter('html:contains("Homepage")')->count() > 0);
    }
}
```

FIGURE 15 – Code simplifié de Test avec Symfony

Cependant, à cause de la courte durée de mon stage, je n'ai pas pu écrire la plupart des tests unitaires de l'application.

### 3.7 Tests de comportement

À l'instar des tests unitaires, je n'ai pas pu effectuer des tests de comportement. Cependant, je les ai préparé grâce à Behat. Pour cela, j'ai écrit des fichiers de " Scénarios " pour décrire le comportement attendu de mon application.

```

Login.feature
1 # language: fr
2 Fonctionnalité: Accéder au Front End
3   Afin de se connecter
4   En tant qu'utilisateur
5   Si je n'arrive pas à me connecter, je devrais pouvoir avoir des choix alternatifs
6   Tels que, réessayer, se créer un compte, réinitialiser son mot de passe
7
8   Scénario: Accéder à l'interface de connexion par clic
9     Étant donné que je suis à "/"
10    Quand je clique sur le lien "Se connecter"
11    Alors je devrais être sur "/login"
12    Et je devrais voir le bouton "Se connecter"
13    Et je devrais voir "Nom d'utilisateur"
14    Et je devrais voir "Mot de passe"
15    Et le code de status de la réponse ne devrait pas être 500
16
17   Scénario: Accéder à l'interface de connexion par url
18     Étant donné que je suis à "/"
19     Quand je vais sur "/login"
20     Alors je devrais voir "Se connecter"
21     Et je devrais voir "Se connecter"
22     Et je devrais voir "Nom d'utilisateur"
23     Et je devrais voir "Mot de passe"
24     Et le code de status de la réponse ne devrait pas être 500
25
26   Scénario: Accéder à l'interface de création de compte par clic
27     Étant donné que je suis à "/"
28     Quand je clique sur le lien "Inscription"
29     Alors je devrais voir "Inscription"
30     Et je devrais voir "Nom d'utilisateur"
31     Et je devrais voir "Mot de passe"
32     Et je devrais voir "Vérification"
33     Et je devrais voir "Adresse e-mail"
34     Et le code de status de la réponse ne devrait pas être 500
35
36   Scénario: Accéder à l'interface de création de compte par url
37     Étant donné que je suis à "/"
38     Quand je vais sur "/register"
39     Alors je devrais voir "Inscription"
40     Et je devrais voir "Nom d'utilisateur"
41     Et je devrais voir "Mot de passe"
42     Et je devrais voir "Vérification"
43     Et je devrais voir "Adresse e-mail"
44     Et le code de status de la réponse ne devrait pas être 500
45
46   Scénario: Accéder à l'interface de réinitialisation de mot de passe par clic
47     Étant donné que je suis à "/"
48     Quand je vais sur "/login"
49     Et que je clique sur le lien "Mot de passe oublié"
50     Alors je devrais voir "Réinitialiser un mot de passe"
51     Et je devrais voir "Adresse e-mail"
52     Et je ne devrais pas voir "Nom d'utilisateur"
53     Et le code de status de la réponse ne devrait pas être 500
54
55   Scénario: Accéder à l'interface de réinitialisation de mot de passe par url
56     Étant donné que je suis à "/"
57     Quand je vais sur "/lostpassword"
58     Alors je devrais voir "Adresse e-mail"
59     Et je devrais voir "Réinitialiser un mot de passe"
60     Et je ne devrais pas voir "Nom d'utilisateur"
61     Et le code de status de la réponse ne devrait pas être 500
62
63   Scénario: Se connecter avec de bons identifiants
64     Étant donné que je suis à "/"
65     Quand je vais sur "/login"
66     Et que je saisis des identifiants de connexion
67     | login | password |

```

FIGURE 16 – Scénarios de login sur l'application

L'avantage de Behat est de pouvoir écrire les scénarios en Français, pour pouvoir discuter des comportements attendu plus facilement avec les collaborateurs. En effet, il est possible de définir de nouvelles méthodes du Context pour créer de nouvelles phrases, adaptées à notre application.

## Troisième partie

# Conclusion

## 4 En marche vers ODE v2.1

Même si j'ai rempli une partie importante du cahier des charges et j'ai effectué toutes les tâches qui m'ont été confiées au début de mon stage, **Open Data Event** doit encore évoluer pour atteindre un niveau de maturité et de fonctionnalité suffisant pour être proposé au grand public ou pour être intégré à d'autres projets.

### 4.1 Finir l'API

Même si toutes les fonctionnalités demandées de l'API ont été développées, je n'ai pas eu le temps d'implémenter la gestion de l'authentification de l'API. Il n'est pas possible, en l'état, de proposer le service au grand public sans authentification.

### 4.2 Import / Export

Pour permettre à des organismes de faire une migration vers ODE, il faut pouvoir proposer un import simple des données, sous plusieurs formats (.ics,.xml,.csv). De plus, il est possible d'imaginer un import automatique des données depuis les sites des fournisseurs d'événements.

### 4.3 Formulaires

ODE v2.1 devrait proposer une customisation simplifiée du formulaire de création d'événement. En effet, chaque instance d'ODE aura des besoins spécifiques quant à sa description d'un événement. Une des possibilités serait de mettre à disposition un Schéma sémantique qui serait analysé pour créer un formulaire adapté.

De plus, il serait intéressant de pouvoir intégrer un service de géolocalisation permettant à chaque événement d'être situé sur une carte. Cependant, il serait important de laisser la possibilité le service de reverse-geocoding de son choix, pour promouvoir les services libres tels que Open Street Map et ne pas obliger l'utilisation de Google Map.

Enfin, utiliser une auto-complétion sur les champs de Catégorie et de Tags ne serait pas de trop pour ne pas multiplier les données inutilement.

### 4.4 Brouillons

Actuellement, dès la création d'un événement, il se retrouve en accès publique. Il serait intéressant de pouvoir créer des " brouillons " d'événements pour laisser le temps de remplir tous les champs de l'événement.

### 4.5 Autres fonctionnalités

Enfin, il y a de nombreuses autres fonctionnalités intéressantes à intégrer au projet, que cela soit l'authentification à OAuth2, l'utilisation de ReCaptcha et d'autres.

## Conclusion

### Bilan du projet

Grâce à toutes les analyses et le travail effectué précédemment, j'ai pu aborder avec facilité le développement d'**Open Data Event**. J'ai pu mettre les bases de l'application, notamment avec l'intégration de SabreDAV, ainsi que la mise en place du stockage des données. De plus, j'ai entamé la réflexion sur le modèle d'un événement, lié à sa représentation sous forme de formulaire. De plus, j'ai mis en place l'API qui pourra être facilement maintenable pour de futures évolutions.

Je pense que l'on peut considérer la version 2.0 comme un succès (tout relatif qu'il soit) et que l'on peut entamer la réflexion sur la version 2.1.

### Bilan personnel

J'ai pu lors de ce stage apprendre grâce à Ronan Guilloux de nombreuses techniques et appréhender de nouvelles technologies telles que Symfony, BootStrap, Composer, ... qui me seront utiles lors de ma carrière professionnelle. J'ai pu grâce au cours dispensés à l'IUT approcher la mission, la lecture, l'analyser et la résolution des problèmes liés aux différentes technologies sereinement, ce qui m'a permis de mener dans les meilleures conditions possibles cette mission. De plus, j'ai pu, grâce aux Polypodes, découvrir le milieu du travail en entreprise, dans un environnement calme et privilégié.

Enfin, je suis fier d'avoir pu contribuer d'une quelconque façon au Logiciel Libre et me sens prêt à continuer dans cette voie là.

## Webographie

- [1] LiberTIC. *OpenData Map*. URL : <http://www.opendata-map.org/>. Consulté le 1 juin 2015.
- [2] Gitimmersion. *Best GIT tuto ever -50 steps*. URL : <http://gitimmersion.com/>. Consulté le 13 avril 2015.
- [3] Vincent Driessen. *A successful Git branching model*. Janvier 2010. URL : <http://nvie.com/posts/a-successful-git-branching-model/>. Consulté le 13 avril 2015.
- [4] Brian Fenton. *Best Practices for Modern PHP Development*. URL : <https://www.airpair.com/php/posts/best-practices-for-modern-php-development>. Consulté le 14 avril 2015.
- [5] Hugo Hamon. *Votre code est STUPID ? Rendez le SOLID*. Décembre 2013. URL : <http://afsy.fr/avent/2013/02-principes-stupid-solid-poo>. Consulté le 14 avril 2015.
- [6] Sensio Labs. *The Symfony Book*. URL : <http://symfony.com/doc/current/book/index.html>. Consulté de nombreuses fois du 15 avril 2015 au 19 juin 2015.
- [7] William Durand. *REST APIs with Symfony2 : The Right Way*. Août 2012. URL : <http://williamdurand.fr/2012/08/02/rest-apis-with-symfony2-the-right-way/>. Consulté le 22 mai 2015.
- [8] The RESTful CookBook. *What is HATEOAS and why is it important for my REST API ?*. URL : <http://restcookbook.com/Basics/hateoas/>. Consulté le 12 juin 2015.
- [9] XEmacs Slartibarfast. *Representational State Transfer (REST) and HATEOAS*. Novembre 2013. URL : <http://fr.slideshare.net/XEmacs/representational-state-transfer-rest-and-hateoas>. Consulté le 12 juin 2015.
- [10] Leonard Richardson. *Act Three : The Maturity Heuristic*. Octobre 2008. URL : <http://www.crummy.com/writing/speaking/2008-QCon/act3.html>. Consulté le 12 juin 2015.

## RFCs

- [11] RFC 4918. *HTTP Extensions for Web Distributed Authoring and Versioning*. URL : <https://tools.ietf.org/html/rfc4918>.
- [12] RFC 4791. *Calendaring Extensions to WebDAV*. URL : <https://tools.ietf.org/html/rfc4791>.
- [13] RFC 6352. *vCard Extensions to WebDAV*. URL : <https://tools.ietf.org/html/rfc6352>.
- [14] RFC 5545. *Internet Calendaring and Scheduling Core Object Specification*. URL : <https://tools.ietf.org/html/rfc5545>.
- [15] RFC 6350. *vCard Format Specification*. URL : <https://tools.ietf.org/html/rfc6350>.
- [16] RFC 7265. *jCal : The JSON Format for iCalendar*. URL : <https://tools.ietf.org/html/rfc7265>.

## Annexes

### Annexe 1 - Lexique

Disponible à l'adresse : [https://github.com/LiberTIC/ODEV2/blob/master/doc/Thibaud\\_Printemps2015/lexique.md](https://github.com/LiberTIC/ODEV2/blob/master/doc/Thibaud_Printemps2015/lexique.md)

#### Protocoles

**WebDAV** (ou **Web Distributed Authoring and Versioning**) est une extension du protocole **HTTP**. Il permet de récupérer, déposer, synchroniser et publier des fichiers rapidement et facilement. WebDAV permet une édition de contenu simultanée avec plusieurs utilisateurs. WebDAV est décrit dans la RFC 4918<sup>[11]</sup>.

Source : [Wikipédia - WebDAV](#)

**CalDAV** (ou **Calendar Extensions to WebDAV**) est un standard internet permettant à un client de planifier des informations sur un serveur distant. Il étend les spécifications de **WebDAV** et utilise le format **iCalendar** pour les données. CalDAV est décrit dans la RFC 4791<sup>[12]</sup>.

Source : [Wikipédia - CalDAV](#)

**CardDAV** (ou **vCard Extensions to WebDAV**) est un standard internet permettant à un client d'accéder et de partager des données de contacts sur un serveur distant. Il étend les spécifications de **WebDAV** et utilise le format **vCard** pour les données. CardDAV est décrit dans la RFC 6352<sup>[13]</sup>.

Source : [Wikipédia - CardDAV](#)

#### Formats de fichiers

**iCalendar** est un format de fichier (.ical ; .ics ; .ifb ; .icalendar) permettant d'envoyer des événements entre utilisateurs. iCalendar est indépendant du protocole de transport. En effet, le transport peut être effectué par mail, par partage sur un serveur **WebDAV** ou même par pigeon voyageur. iCalendar est décrit dans la RFC 5545<sup>[14]</sup>.

Source : [Wikipédia - iCalendar](#)

**vCard** est un format de fichier (.vcf ; .vcard) permettant de transmettre des données personnelles (sous forme de Carte de visite). Le format vCal peut être utilisé en parallèle avec le format **iCalendar** pour lier des événements à des personnes. vCard est décrit dans la RFC 6350<sup>[15]</sup>. La version actuelle est la 4.0, mais la 3.0 reste utilisée par de nombreux clients et serveurs.

Source : [Wikipédia - vCard](#)

**hCalendar** (ou **HTML iCalendar**) est un microformat pour afficher une représentation HTML sémantique d'un calendrier au format iCalendar. L'avantage, outre un affichage personnalisé des événements, est la possibilité de donner des données à des outils de parsing d'extraire les

informations pour les stocker sous le format souhaité (iCalendar ou autre). hCalendar est utilisé entre autres par Facebook, Google et Wikipédia.

*Source : [Wikipédia - hCalendar](#)*

**jCal** est une spécification de format JSON pour iCalendar. Il permet de stocker les données décrit par iCalendar sous le format JSON. Il est défini par la RFC 7265<sup>[16]</sup>. Cependant, cette RFC n'est qu'une proposition de standard, émise en mai 2014 : elle peut donc évoluer.

*Source : [IETF - RFC 7265](#)*

## Clients

**iCal** (renommé **Apple Calendar** depuis OSX Mountain Lion) est l'application de gestion de calendrier fait par Apple. Il peut entre autres, être client d'un serveur **WebDAV**.

*Source : [Wikipédia - Calendar](#)*

## Serveurs

**SabreDAV** est un serveur **CardDAV**, **CalDAV** et **WebDAV**. Il implémente les recommandations RFC actuelles. Il est compatible avec toutes les plateformes majeures.

*Source : [Wikipédia - SabreDAV](#)*

**CalServ**, aussi appelé **Calendars and Contacts Server** ou **Calendar Server** est un projet d'Apple d'implémentation des protocoles CalDAV et CardDAV. Sortie en 2006 sous le nom de iCal Server and Address Book Server, il a été porté sur des plateformes non-Apple. Il est écrit en Python et utilise une base de données SQL pour stocker les données.

*Source : [Wikipédia - Calendar Server](#)*

## Mix

**Baïkal** est une surcouche de SabreDAV. Il propose entre autres une interface d'administration web permettant la gestion du serveur CalDAV et CardDAV. Il est donc serveur et client de son propre serveur.

*Source : [Site Baïkal](#)*

## Annexe 2 - Modèle Événement

Disponible à l'adresse : [https://github.com/LiberTIC/ODEV2/blob/master/doc/Thibaud\\_Printemps2015/Modele\\_Evenement.md](https://github.com/LiberTIC/ODEV2/blob/master/doc/Thibaud_Printemps2015/Modele_Evenement.md)

Propriété	Type	Description	Source	Exemple
<b>Base</b>				
Nom	Texte	Nom court de l'événement	iCalendar	Concert Shakaponk
UID	Nombre	Identifiant Unique de l'événement	iCalendar	SL-2015-XYZ-004
Description	Texte	Description de l'événement	iCalendar	Concert de rock avec [...]
<b>Date et heure</b>				
Date début	Date	Date et heure de début	iCalendar	2015-06-20 / 20 :00
Date fin	Date	Date et heure de fin	iCalendar	2015-06-20 / 23 :30
Date création	Date	Date de création de l'événement	ODE_V2	2015-04-01 / 13 :37
Date modification	Date	Date de modif de l'événement	ODE_V2	2015-04-03 / 20 :15
<b>Localisation</b>				
Lieu	Texte	Nom de l'endroit	iCalendar	Zénith Nantes
Emplacement	Texte	Nom de l'endroit	ODE_V2	Salle de concert #3
Géolocalisation	Geo	Géolocalisation de l'événement	iCalendar	47.229234, -1.628550
Capacité du lieu	Nombre	Capacité du lieu de l'événement	ODE_V1	4000 (personnes)
<b>Organisation</b>				
Participants	Texte	Participants à l'événement	schema.org	Shakaponk ; Tagada Jones
Durée	Durée	Durée de l'événement	schema.org	PT3H30M (3h30min)
Status	Texte	Status de l'événement	iCalendar	Annulé / Reporté
Organisateur	Texte	Organisateur de l'événement	schema.org	Séréolux
Sous-Événement	UID	UID d'un sous-événement	schema.org	SL-2015-XYZ-009
Super-Événement	UID	UID d'un sur-événement	schema.org	SL-2015-XYZ-001
<b>URLs</b>				
URL	URL	URL vers le site ODEV2	iCalendar	http ://ODEV2/event/XYZ004
URL orga	URL	URL de l'événement sur le site de l'orga	schema.org	http ://stereolux/event/XYZ004
URLs médias	URLs	URL de média compatible oEmbed	schema.org	http ://website/image.jpg
<b>International</b>				
Langue	Texte	Langue de l'événement	ODE_V1	FR
<b>Tarifs</b>				
Prix standard	Nombre	Prix au tarif normal	ODE_V2	10
Prix réduit	Nombre	Prix au tarif réduit	ODE_V2	7.5
Prix enfant	Nombre	Prix au tarif enfant	ODE_V2	5
<b>Contacts</b>				
Contact - Nom	Texte	Nom du contact	ODE_V1	John Smith
Contact - Email	Email	Email du contact	ODE_V1	john.smith@email.com
<b>Catégorisation</b>				
Catégorie	Texte	Catégorie de l'événement	ODE_V1	Concert
Tags	Texte	Tags de l'événement	ODE_V1	Rock ; Alternatif ; [...]



Annexe 3 - Diagramme de Cas d'Utilisation

