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.

Refact this

Resumé

Table des matières

Présentation du stage

Présentation d'Open Data Event

Deuxième partie

Développement

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 réprésentants 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 résumé, on pourrait regrouper la définition en trois points :

- **Disponibilité et Accès** : Les données doivent être disponibles dans leurs ensembles et gratuitement. De plus, les données doivent être disponible 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 popularisé en France. Ainsi, depuis quelques années, nous pouvons voir l'apparition de plusieurs portails de données de villes, de département ou de régions (tels que **Paris Data**, **Data Pays de la Loire**, etc...)

LiberTIC est une association nantaise, fondé 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[1] 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 simplifié, on peux 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 première question que l'on se pose quand on entend parler de logiciel libre fait par une entreprise est la suivante :

"Mais si c'est gratuit, comment ils font pour gagner de l'argent ? ?"

Il s'agit la plupart du temps d'une confusion entre le développement d'un logiciel libre par une entreprise et du service proposé sur ce service par cette entreprise. Un des exemple intéressant 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 lance 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.

Les Polypodes fait partie de ceux qui contribue 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 la seule activité de l'agence. 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)*

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$^{\text{ème}}$ 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.

FIGURE 1 – Partie des bureaux chez Les Polypodes

1.3 L'aggrégateur d'événement (ODE)

L'aggrégateur d'événement, appellé **Open Data Event** (ODE), est une plateforme permettant de relier des fournisseurs de d'événements (les organisateurs de festivals, concerts, conférence, 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 étais 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 plusieurs phases d'analyses, de conceptions et de développements.

La première réunion

L'élément déclencheur de ce projet est 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 aggrégateur d'événement permettant de relier les fournisseurs d'événements (les organisateurs de festivals, concerts, conférence, 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 addresse : `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 responsable 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 développé par Makina Corpus est programmé en Python. Il s'agit d'une API REST qui permet aux clients d'intéragir 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 maitrise 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 accueillir 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 la même fin 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 bibliothèrques open-source permettant de manipuler des données au format iCalendar avec le language 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 sur le 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 assignés.

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'établissais, avec Ronan, des " Sprints " d'un semaine décrivant la liste des tâches à effectuer, et je préparerai 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 effectuions des " Daily scrum " tout les matins chez Les Polypodes. Il s'agit globalement d'une petite réunion entre tous les employés où nous répondons aux trois questions suivantes : " *Qu'est ce que j'ai fait hier pour attendre les objectifs ?* ", " *Que vais-je faire aujourd'hui pour attendre les objectifs ?* " et " *Est ce que je voi des problèmes qui pourraient m'empécher d'attendre 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 disponible sur le Github aux côtés de la documentation du projet.

2 Analyse et Conception

Comme tous 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 étais très importante.

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 tout les logiciels nécessaire à la réalisation du projet. Pour des raisons personnels, 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 maitrisais pas. C'est pourquoi il a été convenu que la première semaine de mon stage devais me servir pour effectuer une mise à niveau pour différentes technologies.

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

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

Enfin, j'ai lu et appliqué la totalité du **Symfony Book** [6].

2.3 Analyse des besoins

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 de ce que je proposais comme architecture du projet suite à l'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 maîtrisé par l'équipe, cela était le choix le plus logique pour pouvoir être accompagné par toute l'équipe en cas de besoin.

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

CalDAV et WebDAV

Dès la première réunion, il était clair qu'il fallais se baser sur un protocole déjà en place, pour ne pas " réinventer la roue ". Le seul protocole actuellement en place répondant à nos besoin est **CalDAV**. Il s'agit d'une extension du protocole WebDAV, lui même une extension du protocole HTTP (plus d'information 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[11] datant de 2007. Il existe plusieurs implémentation de cette RFC. La première, historiquement parlant, est CalendarServer, en python.

3 Développement et Tests

Après la phase de Conception, viens la phase de développement, suivi 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 composant pour accélérer et faciliter le développement d'une application PHP. De plus, Symfony possède une communauté très impliqué, ce qui permet de trouver de très nombreuses documentation très bien rédigé.

Symfony est sponsorisé par SensioLabs, une entreprise française. Sponsorisé car, même si cette entreprise a développé la première version du framework, elle est maintenant entretenu 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 adaptés à nos besoin. 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
                           dont Symfony                      */

- 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 2 – Architecture proposé 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 3 – Code simplifié avec exemple d'HTTPFoundation

Comme on peux 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 privé, nous pourions 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 proposé par Symfony (Figure 1), 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 lié à SabreDAV, je vous en parlerais dans la section suivante. La dernière, **UserManager**, est, quant à elle, lié à **FOSUSerBundle** ; il s'agit d'un bundle permettant une gestion simplifié des utilisateurs. Il gère, entre autres, l'inscription, la connexion, le chiffrage des mots de passes, les sessions liés à l'utilisateurs, les rôles, etc...

Controller

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

La première classe, **DefaultController** est le controlleur le plus léger (2 fonctions) : Index du site et page de test.

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

La troisième, **BrowserController** est le controlleur définissant le Front-End de l'application. C'est lui qui gère les pages et les formulaires proposé à l'utilisateur. Il est notemment en lien avec FOSUserBundle pour intégrer la connexion et l'inscription des utilisateurs.

Enfin, **APIController** est le controlleur 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, notemment dans la création des champs des formulaires.

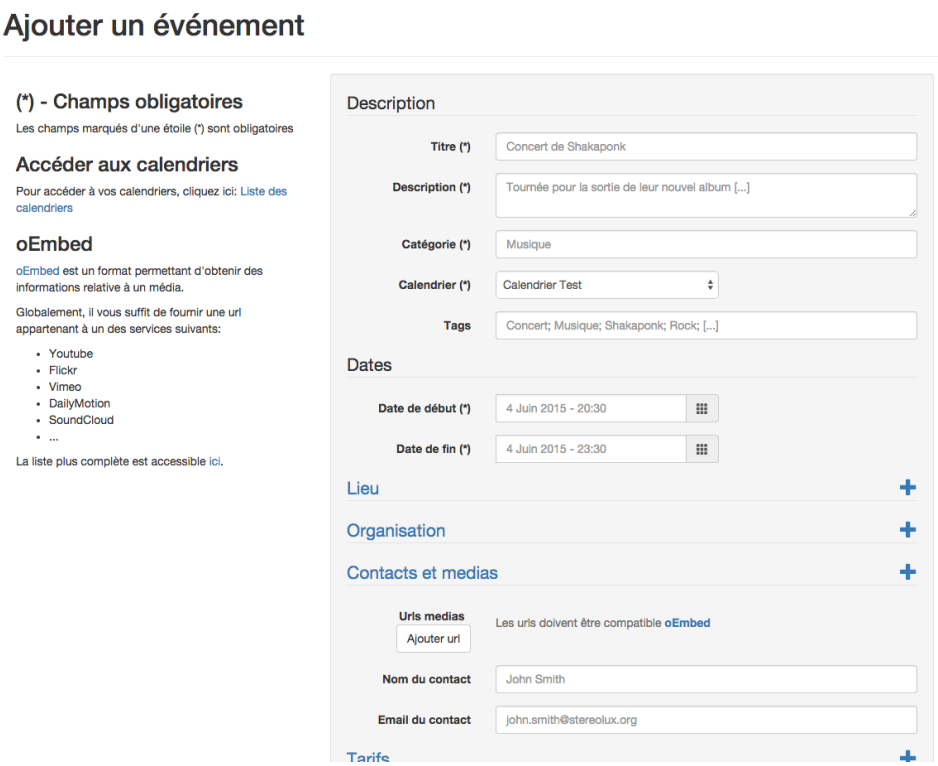

FIGURE 4 – Formulaire de création d'un événement

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'événements.

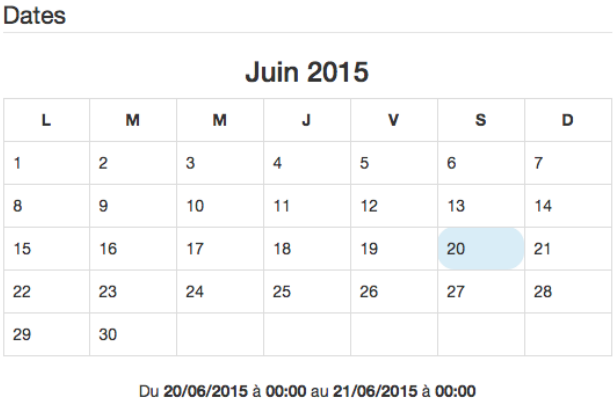

FIGURE 5 – Calendrier généré avec javascript

Service

Un service (dans Symfony) est un simple objet PHP qui remplit une fonction. 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'accèder aux services depuis les controlleurs 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 6 – 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 protocol 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 rêquetes "PROPPATCH", "MKCALENDAR", etc. propres à CalDAV.

SabreDAV un projet Open Source développé par Fruux. Fruux fait parti 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 Controlleur spécialement pour la partie CalDAV (CalDAVController.php). Au niveau des routes, j'ai les ai toutes redirigés vers la méthode " indexAction " qui controlleur. 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 constuction et l'envoi de la réponse, ce qui perturbais le fonction 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 envoyé par Sabre-

DAV sera attrapé par Symfony puis renvoyé vers l'utilisateur. Cela permet aussi de logger les messages renvoyé, ce qui m'a été très utile tout au long du développement de l'application.

```php
<?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 7 – 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 implémenté sont les suivantes :

Auth.php : Celle ci sert à récupérer le mot de passe chiffré en fonction du nom d'utilisateur.

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 pourrais 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 mes controlleurs, car même si elle est utilisé par SabreDAV, elle s'adapte très facilement à n'importe quelle utilisation.

3.3 Implémentation avec ElasticSearch

ElasticSearch est un moteur de 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'ElasticSeach requiert de plus la version 1.8 de la Java Virtual Machine. ElasticSearch n'étant pas adapté à OSX, j'ai du installer une machine virtuelle sur mon Mac. Pour cela, j'ai utilisé **Vagrant**.

Après avoir installé Vagrant, il suffit d'éxécuter 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 controlleurs. De plus, je l'ai ajouté au ServiceContainer de Symfony pour l'utiliser facilement.

Un des problèmes rencontré avec ElasticSearch était le fait que 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 veux 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 apprécié dans l'utilisation d'ElasticSearch dans la plupart des cas. Cependant, jCal suit une architecture différente de JSON comme le montre la figure suivante :

```
// JSON Classique

{
    "glossary": {
        "title": "example␣glossary",
        "GlossDiv": {
            "title": "S",
            "GlossList": {
                "GlossEntry": {
                    "ID": "SGML",
                    "SortAs": "SGML",
                    "GlossTerm": "Standard␣Generalized␣Markup␣Language",
                    "Acronym": "SGML",

[...]
```

FIGURE 8 – 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 9 – Exemple jCal

3.4 Implémentation avec PostgreSQL

azfezf

efz

zef

zef

fze

fze

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 protocol. 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 10 – 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)*. C'est une amélioration de la philosophie lié à REST. Il s'agit de 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.

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 bon 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
<?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 11 – Code simplifié de Test avec Symfony

3.7 Tests de comportement

En plus des tests unitaires, j'ai effectué des tests de comportants grâce à Behat. Pour cela, j'ai d'abord écris des fichiers de " Scénarios " pour décrirer le comportement attendu de mon 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, adapté à notre application.

Login.feature

```
# language: fr
Fonctionnalité: Accéder au Front End
        Afin de se connecter
        En tant qu'utilisateur
        Si je n'arrive pas à me connecter, je devrais pouvoir avoir des choix alternatifs
        Tels que, réessayer, se créer un compte, réinitialiser son mot de passe

    Scénario: Accéder à l'interface de connexion par clic
        Étant donné que je suis à "/"
        Quand je clique sur le lien "Se connecter"
        Alors je devrais être sur "/login"
        Et je devrais voir le bouton "Se connecter"
        Et je devrais voir "Nom d'utilisateur"
        Et je devrais voir "Mot de passe"
        Et le code de status de la réponse ne devrait pas être 500

    Scénario: Accéder à l'interface de connexion par url
        Étant donné que je suis à "/"
        Quand je vais sur "/login"
        Alors je devrais voir "Se connecter"
        Et je devrais voir "Se connecter"
        Et je devrais voir "Nom d'utilisateur"
        Et je devrais voir "Mot de passe"
        Et le code de status de la réponse ne devrait pas être 500

    Scénario: Accéder à l'interface de création de compte par clic
        Étant donné que je suis à "/"
        Quand je clique sur le lien "Inscription"
        Alors je devrais voir "Inscription"
        Et je devrais voir "Nom d'utilisateur"
        Et je devrais voir "Mot de passe"
        Et je devrais voir "Vérification"
        Et je devrais voir "Adresse e-mail"
        Et le code de status de la réponse ne devrait pas être 500

    Scénario: Accéder à l'interface de création de compte par url
        Étant donné que je suis à "/"
        Quand je vais sur "/register"
        Alors je devrais voir "Inscription"
        Et je devrais voir "Nom d'utilisateur"
        Et je devrais voir "Mot de passe"
        Et je devrais voir "Vérification"
        Et je devrais voir "Adresse e-mail"
        Et le code de status de la réponse ne devrait pas être 500

    Scénario: Accéder à l'interface de réinitialisation de mot de passe par clic
        Étant donné que je suis à "/"
        Quand je vais sur "/login"
        Et que je clique sur le lien "Mot de passe oublié"
        Alors je devrais voir "Réinitialiser un mot de passe"
        Et je devrais voir "Adresse e-mail"
        Et je ne devrais pas voir "Nom d'utilisateur"
        Et le code de status de la réponse ne devrait pas être 500

    Scénario: Accéder à l'interface de réinitialisation de mot de passe par url
        Étant donné que je suis à "/"
        Quand je vais sur "/lostpassword"
        Alors je devrais voir "Adresse e-mail"
        Et je devrais voir "Réinitialiser un mot de passe"
        Et je ne devrais pas voir "Nom d'utilisateur"
        Et le code de status de la réponse ne devrait pas être 500

    Scénario: Se connecter avec de bons identifiants
        Étant donné que je suis à "/"
        Quand je vais sur "/login"
        Et que je saisis des identifiants de connexion
            | login      | password      |
```

FIGURE 12 – Scénarios de login sur l'application

4 En marche vers ODE v2.1

Même si j'ai rempli un partie importante du cahier des charges et ai effectué toutes les tâches qui m'ont été confié 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 intégrer à d'autres projets.

4.1 Finir l'API

Même si toutes les fonctionnalités demandées de l'API ont été développé, 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 simplifié 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 interessant 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 Categorie 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 Plein d'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.

Troisième partie

Conclusion

Conclusion

Bilan du projet

Bilan personnel

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.

RFCs

[10] RFC 4918. *HTTP Extensions for Web Distributed Authoring and Versioning*. URL : https://tools.ietf.org/html/rfc4918.

[11] RFC 4791. *Calendaring Extensions to WebDAV*. URL : https://tools.ietf.org/html/rfc4791.

[12] RFC 6352. *vCard Extensions to WebDAV*. URL : https://tools.ietf.org/html/rfc6352.

[13] RFC 5545. *Internet Calendaring and Scheduling Core Object Specification*. URL : https://tools.ietf.org/html/rfc5545

[14] RFC 6350. *vCard Format Specification*. URL : https://tools.ietf.org/html/rfc6350.

[15] RFC 7265. *jCal : The JSON Format for iCalendar*. URL : https://tools.ietf.org/html/rfc7265.

Annexes

Annexe 1 - Lexique

Disponible à l'addresse : `https://github.com/LiberTIC/ODEV2/blob/master/doc/Thibaud_Printemps2015/lexique.md`

Protocoles

WebDAV (ou **Web Distributed Authoring and Versioning**) est une extension du protocol **HTTP**. Il permet de récupérer, déposer, synchronier et publier des fichiers rapidement et facilement. WebDAV permet une édition de contenu simultanée avec plusieurs utilisateurs. WebDAV est décrit dans la RFC 4918[10].

Source : Wikipédia - WebDAV

CalDAV (ou **Calendar Extensions to WebDAV**) est 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[11].

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 disant. Il étend les spécifications de **WebDAV** et utilise le format **vCard** pour les données. CardDAV est décrit dans le RFC 6352[12].

Source : Wikipédia - CardDAV

Formats de fichiers

iCalendar est un format de fichier (.ical ; .ics ; .ifb ; .icalendar) permettant d'envoyer des événements entre utilisateur. iCalendar est indépendant du protocol de transport. En effet, le transport eut être effectué par mail, par partage sur un serveur **WebDAV** ou même par pigeon voyageur. iCalendar est décrit dans la RFC 5545[13].

Source : Wikipédia - iCalendar

vCard est un format de fichier (.vcf ; .vcard) permettant de 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[14]. La version actuelle est la 4.0, mais la 3.0 reste utilisé 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 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[15]. Cependant, cette RFC n'est qu'une proposition de standard, émise en mai 2014 : elle peux 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 peux entre autres, être client d'un serveur **WebDAV**.

Source : Wikipédia - Calendar

Serveurs

SabreDAV est un serveur **CardDAV**, **CalDAV** et **WebDAV**. Il implémente les recommendations 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 Evénement

Disponible à l'addresse : `https://github.com/LiberTIC/ODEV2/blob/master/doc/Thibaud_Printemps2015/Modele_Evenement.md`

Propriété	Type	Description	Source	Exemple
Base				
Nom	Texte	Nom court de l'événement	iCalendar	Concert Shakaponk
UID	Nombre	Identifiant Unique de l'event	iCalendar	SL-2015-XYZ-004
Description	Texte	Description de l'événement	iCalendar	Concert de rock avec [...]
Date et heure				
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'event	ODE_V2	2015-04-01 / 13 :37
Date modification	Date	Date de modif de l'événement	ODE_V2	2015-04-03 / 20 :15
Localisation				
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'event	ODE_V1	4000 (personnes)
Organisation				
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	Stéréolux
Sous-Evénement	UID	UID d'un sous-événement	schema.org	SL-2015-XYZ-009
Super-Evénement	UID	UID d'un sur-événement	schema.org	SL-2015-XYZ-001
URLs				
URL	URL	URL vers le site ODEV2	iCalendar	http ://ODEV2/event/XYZ004
URL orga	URL	URL de l'event 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
International				
Langue	Texte	Langue de l'événement	ODE_V1	FR
Tarifs				
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
Contacts				
Contact - Nom	Texte	Nom du contact	ODE_V1	John Smith
Contact - Email	Email	Email du contact	ODE_V1	john.smith@email.com
Catégorisation				
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