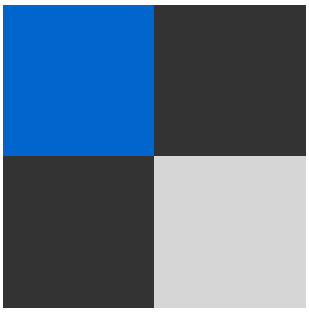
Ecole Polytechnique de l'Université François Rabelais de Tours
Département Informatique
64 avenue Jean Portalis
37200 Tours, France
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Projet Recherche & Développement
2015-2016

VRP-REP



Tuteurs académiques
Jorge Mendoza

Étudiants
Valentin Maerten (DI5)

26 septembre 2016

Liste des intervenants

Nom	Mail	Qualité
Valentin Maerten	valentin.maerten@etu.univ-tours.fr	Étudiant DI5
Jorge Mendoza	jorge.mendoza@univ-tours.fr	Tuteur académique, Département informatique - équipe OC

Avertissement

Ce document a été rédigé par Valentin Maerten susnommé l'auteur.

L'école polytechnique de l'université François Rabelais de Tours est représentée par Jorge Mendoza susnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnait assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnait qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.

Pour citer ce document :
Valentin Maerten, *VRP-REP*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2015-2016.

```
@mastersthesis{
    author={Maerten, Valentin},
    title={VRP-REP},
    type={Projet Recherche \& Développement},
    school={Ecole Polytechnique de l'Université François Rabelais de Tours},
    address={Tours, France},
    year={2015-2016}
}
```

Document mis en forme sous LaTeX
grâce à la classe de document `polytech.cls` V1.0 (2016/02/01)

Table des matières

Table des figures

Conclusion

Liste des tableaux

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon projet et qui m'ont aidé lors de la rédaction de ce rapport. Je souhaite remercier en particulier M. Mendoza, enseignant chercheur au département informatique de Polytech Tours, qui m'a accompagné tout au long de ce projet.

Je souhaite pour finir, remercier M. Aupetit et M. Monmarché pour l'encadrement général des projets de recherche et développement.

Introduction

Ce projet s'inscrit dans le cadre des Projets de Recherche & Développement (PRD) que les élèves de cinquième année sont amenés à réaliser à l'École Polytechnique Universitaire de Tours.

Ce projet s'étale sur toute l'année à raison de deux jours par semaine. Les objectifs principaux sont de mettre en pratique les connaissances théoriques et techniques acquises au cours des années d'études précédentes grâce à une partie recherche avec un état de l'art et une partie développement.

Durant ce projet, j'ai eu un contact régulier avec mon encadrant afin de partir dans la bonne direction.
De plus, cette année nous devons faire un compte rendu écrit à notre encadrant à la fin de chaque semaine. Ces comptes rendus figureront aussi dans ce rapport.

Première partie

Partie recherche

1 Le projet

1 Contexte du projet

1.1 Objectif du site

M. Mendoza avec des enseignants-chercheurs de l'université d'Angers, a créé le site vrp-rep.org (que j'appellerai VRP-REP par la suite) qui a pour principal objectif de rassembler le maximum d'instances pour le problème du VRP ainsi que les différentes solutions proposées afin de pouvoir comparer les algorithmes entre eux.

Avant le lancement du site, chaque personne créait ses instances avec son propre format, c'est pour cela que le comité du site a créé un format pour la description d'une instance afin d'essayer d'uniformiser les différents formats présents sur internet.

1.2 Terminologies

Il est important de faire un point sur les termes qui sont utilisés sur le site.

- **References** : une référence représente un article dans lequel une méthode de résolution est proposée pour une ou plusieurs variante(s)
- **Problem variants** : Il existe plusieurs variantes du problème VRP, un *problem variant* représente une variante du problème
- **Instance** : représente une définition du problème avec les différents nœuds, arcs ainsi que la fonction objectif
- **Dataset** : représente un ensemble de solutions. Un dataset est lié à une variante ainsi qu'a une référence
- **Solution** : représente la liste des fonctions objectives pour chaque instance d'un dataset. Une solution est donc liée à un dataset, à une variante, et pour finir à une référence
- **BSK tables** : *BSK* pour *Best Known Solutions* est la table des meilleures solutions pour le couple dataset et variante

1.3 Fonctionnalités

VRP-REP propose plusieurs fonctionnalités qui sont les suivantes :

- Consulter la table BSK
- Consulter les différentes variantes ainsi que les datasets et références associées.
- Consulter les différentes références avec leurs datasets et variantes associées.
- Consulter tous les datasets ainsi que les différentes solutions proposées.
- Télécharger les datasets
- Déposer un dataset
- Déposer une solution

Pour déposer un dataset ou une solution, il est nécessaire d'être inscrit. Pour les opérations de consultation ou de téléchargement, il n'est pas nécessaire d'être inscrit.

[WWW6] Le format créé est un format reposant sur le XML. La spécification du format est disponible ici sur le GitHub du site.
Lorsque l'on dépose un dataset au format VRP-REP, il y a un vérificateur qui vérifie que toutes les instances du dataset sont bien conformes aux normes VRP-REP.

S'il y a un problème, l'erreur est explicitée (s'il manque une balise ou si une balise présente n'est pas autorisée).

2 Problématique

Sur certaines variantes du VRP (Vehicle Routing Problem), il est nécessaire d'avoir plusieurs critères pour la fonction objectif.
Actuellement, il n'est possible de n'avoir qu'un seul critère sur la fonction objectif par problème. Ce qui peut poser problème sur certains problèmes, par exemple une variante où l'on peut envoyer plusieurs camions en même temps.
Sur ce type de problème, il est surement utile d'avoir deux critères : un qui va compter le temps total et un qui va compter le nombre de camions.

Cela permettra d'être plus juste sur la comparaison des algorithmes, si quelqu'un a fait un temps minimal, mais a utilisé un grand nombre de camions ce n'est pas la même chose que quelqu'un qui fait un temps un peu plus élevé, mais qui divise par deux le nombre de camion.

Avec cette solution, nous pourrons afficher les deux (ou plusieurs) critères, et même trier les solutions en fonction de tels ou tels critères.

D'autre part, certains chercheurs voudraient pouvoir mettre à disposition leurs datasets mais de manière privée. C'est-à-dire, non visible par tout le monde, mais téléchargeable par certaines personnes suivant un lien mis à disposition.

Cela leur permettra d'avoir une manière centralisée de stocker leurs dataset et pouvoir les partager uniquement avec les personnes en ayant l'autorisation.

Le dataset devra donc apparaitre sur la liste des datasets uniquement pour son auteur.

3 Le problème du Vehicle Routing Problem

Le problème du VRP [1], en français, problème de tournées de véhicules est le grand frère du TSP [2], en français, problème du voyageur de commerce.
Je vais donc commencer par définir le problème du voyageur de commerce.

3.1 Traveling Salesman Problem

D'après [1] , le TSP possède en entrée un ensemble de villes avec le coût de voyage entre chaque paire. Toutes les villes ne sont pas forcement reliées entre-elles.
Le but du TSP est de trouver le chemin le moins couteux pour visiter toutes les villes et revenir au point de départ.

L'ordre dans lequel les villes sont visitées s'appelle un *tour* ou un *circuit*.

Le TSP a des applications dans la logistique, les télécommunications, les neurosciences, etc.

En mathématique, le TSP est modélisé par un graphe complet et le but est de trouver le plus court cycle hamiltonien dans le graphe. Le TSP est un problème NP-Complet.

3.2 Vehicle Routing Problem

[3] Une définition générique du Vehicle Routing Problem peut-être la suivante :
On possède comme pour le TSP, un ensemble de villes avec le coût de voyage entre chaque paire. De plus, en entrée on a une liste de demande de transport et une liste de véhicules.

La problème est le suivante : il faut trouver la liste des itinéraires pour tous les véhicules afin de satisfaire toutes les requêtes avec un coût minimum.
En particulier, décider quel véhicule va traiter quelles requêtes et dans quel ordre pour que les itinéraires soient faisables.

Tout comme le TSP, le VRP est un problème NP-Complet.

3.3 Variantes

Il existe de multiples variantes du problème général du VRP. La variante la plus étudiée du VRP est "Capacitated Vehicle Routing Problem" (CVRP).

Dans la littérature, les variantes sont très nombreuses, par exemple sur le site vrp-rep il y en 14 de répertoriés.
J'ai choisi de décrire succinctement et arbitrairement 3 variantes qui sont :

- Capacitated Vehicle Routing Problem (CVRP)
- The Vehicle Routing Problem with Time Windows
- The Vehicle Routing Problem with Stochastic Demands

1. Vehicle Routing Problem
2. Traveling Salesman Problem

3.3.1 Capacitated Vehicle Routing Problem (CVRP)

[3] Dans le CVRP, la demande de transport consiste en la distribution de biens depuis un unique dépôt vers n autres points, qu'on appellera des clients.
Chaque véhicule a la même capacité de transport.

L'objectif est de minimiser le nombre de véhicules ainsi que la somme de temps de trajet. La demande totale de chaque route ne doit pas être plus grande que la capacité du véhicule.

Le CVRP est comme le VRP, mais avec une contrainte en plus qui est que tous les véhicules doivent avoir la même capacité.

3.3.2 The Vehicle Routing Problem with Time Windows

[3] The Vehicle Routing Problem with Time Windows (VRPTW) est une extension du CVRP où chaque client possède une fenêtre de temps dans laquelle il doit être livré.
Cette fenêtre peut-être dite dure ou souple. Si la fenêtre est dure, alors si le véhicule arrive plus tôt alors il devra attendre (cela n'engendre aucun coût supplémentaire). Si la fenêtre est souple, si le véhicule arrive plus tôt, il peut alors violer la fenêtre de temps avec des pénalités.
Dans tous les cas, le véhicule ne peut pas arriver après la fenêtre de temps.

Les applications dans la vie réelle peuvent être par exemple, la livraison de banque, parcours de bus scolaire, etc.

3.3.3 The Vehicle Routing Problem with Stochastic Demands

[2, 3] Ici les demandes des clients sont des variables aléatoires avec une loi de probabilité connue à l'avance.
Il en résulte que la capacité du véhicule ne sera peut-être pas suffisante pour répondre à toutes les demandes des clients pendant son chemin. En cas d'échec, le chauffeur doit faire un réapprovisionnement au dépôt.

2 Cahier des charges

L'introduction et le contexte de la réalisation ont été définis plus tôt dans ce rapport.

M. Mendoza et le comité gérant du site seront la MOA [1] pour ce projet et je serais la MOE [2]

1 Cas d'utilisations

1.1 Dataset privé

L'utilisateur veut pouvoir rendre un dataset privé. De cela en découle un certain nombre de cas d'utilisation à prévoir :

- Possibilité de passer un dataset en privé (mais il ne doit pas être possible de passer de public à privé).
- Générer un lien permettant de télécharger le dataset
- Le dataset ne doit pas être visible par les autres, mais uniquement par son auteur.

1.2 Variante du problème multicritères

Comme expliqué dans la problématique (Section 2 (Chapitre 1)), le but est que l'utilisateur puisse envoyer des instances d'un problème avec potentiellement plusieurs critères à prendre en compte.

Il y a deux cas à prendre en compte :

- Soit le format de l'instance correspond au format du VRP-REP, dans ce cas il faudra générer automatiquement le fichier .xsd qui permettra de vérifier la validité de l'instance.
- Soit le format de l'instance ne correspond pas au format du VRP-REP, ici on ne pourra pas vérifier l'instance.

Dans tous les cas, il faudra vérifier les solutions pour vérifier que tous les critères sont bien présents.

L'utilisateur peut vouloir aussi visualiser la table BSK, dans ce cas on affichera tous les critères dans cette table avec la possibilité de trier par rapport à un certain critère.

1. Maitrise d'ouvrage
2. Maitrise d'œuvre

2 Travail à réaliser

Dans cette partie, je vais détailler le travail que je vais réaliser durant la partie développement de ce projet.

2.1 Réalisation de la migration du site sur un VPS

Pour répondre au besoin technique du site, j'ai proposé à M. Mendoza de migrer le site sur un VPS. En effet, l'hébergement mutualisé commençait à montrer ses limites en particulier avec l'upload et la vérification des grosses instances.

Une première phase, faite en partie recherche, qui est l'étude des différents VPS et le choix de l'un d'entre eux.
Mon travail dans cette partie est en premier lieu, d'installer l'environnement technique, c'est-à-dire, PHP, Apache2, MySQL.
Dans un deuxième temps, je vais installer l'application VRP-REP tournant sous Symfony2 avec toutes les dépendances (Composer).

Pour finir, je vais installer Git et mettre en place le hook afin de pouvoir récupérer l'application avec un *clone* et de déployer sur les différents serveurs.

Tout ceci sera réalisé avec le guide de migration que j'aurais réalisé durant le premier semestre.

2.2 Petites améliorations / résolution de bug

Une partie de ma tâche va être de faire de petites améliorations et des résolutions de bug afin de prendre en main le code.
Après la première réunion que j'ai eu avec M. Mendoza et Hubert Lobit, nous avons défini une liste de de petits développements à faire :

- Faire que l'affiliation soit obligatoire lors de l'inscription
- Lors que l'on télécharge un dataset, les instances sont forcement téléchargées avec l'extension .xml, il faudra charger l'extension dynamiquement.
- Envoyer un email au comité lorsqu'un utilisateur s'inscrit
- Mettre une majuscule automatiquement sur le prénom et sur le nom des utilisateurs.

Par la suite, j'ai découvert quelques bugs à résoudre et des améliorations du système que je pourrais proposer :

- Il n'y a pas de validation côté serveur, que ce soit à l'inscription ou à la modification du profil. Ce qui veut dire que si le navigateur ne supporte pas HTML5 ou alors modifie le code HTML grâce par exemple à Firebug, il peut soumettre le formulaire sans les valeurs requises et faire planter le système. De plus il n'y pas non plus de vérification de l'unicité de la clé primaire des utilisateurs (adresse email).
- Actuellement, il y a une barre de progression durant l'upload, mais aucun loader durant la vérification des instances, processus qui peut être long suivant le nombre et la taille des fichiers. Je vais donc ajouter un loader durant ce processus afin d'avertir l'utilisateur du traitement en cours.
- Dans le même contexte que l'item du dessus, si le système crash pendant la vérification des instances, l'utilisateur n'est pas averti. Je vais donc afficher un cadre rouge avec l'erreur afin que l'utilisateur puisse la faire remonter aux administrateurs du site.

2.3 Modification pour avoir des datasets privés

2.3.1 Système existant

Actuellement, le système ne prévoit pas la possibilité d'avoir un dataset privé. Lorsque l'on upload un dataset, il est automatiquement référencé sur le site ainsi que visible et téléchargeable par tout le monde. Cette modification est prioritaire sur la précédente. Je ferai donc cette tâche en priorité.

2.3.2 Amélioration prévue

Durant la partie développement, je vais modifier le schéma de la base de données afin de pouvoir avoir un dataset privé ainsi que la possibilité de stocker un token pour un dataset privé.

Il me faudra adapter le système en conséquence :

- Sur la page du dataset, lorsque l'on est son auteur, il y aura la possibilité de changer la visibilité du dataset (privé => public) avec la génération du token associé
- Re-générer un token à la demande
- Avoir une page de téléchargement pour le dataset lorsque le bon token est donné
- Ne pas afficher les datasets privés dans la liste des datasets.

2.4 Changement de type d'URL pour les datasets

Actuellement, les URL des datasets sont de la forme : /datasets/item/slug.html avec le slug qui est créé à partir de l'identifiant du dataset. Cependant, l'identifiant et donc le slug peuvent être modifiés par le créateur du dataset.
M. Mendoza souhaite avoir une URL du type /datasets/item/id.html avec un identifiant unique généré à l'upload du dataset qui ne changera pas.

Pour comprendre ce besoin prenons un exemple, M. Mendoza a soumis un papier sur le VRP fin 2014. Il a donc aussi créé des datasets illustrant ce papier, il a naturellement daté ces datasets de 2014. La réponse positive de la publication de son papier lui est revenue en 2015, mais par manque de place son papier ne sera publié qu'en 2016.
Donc lors de la publication de son papier, M. Mendoza changera la date de ses dataset (donc les identifiants de ses datasets) par 2016 afin de rester cohérent avec la date de publication de son papier. En effet, cela serait bizarre de voir des datasets de 2014 sur un papier publié en 2016.

Or, des utilisateurs du site auront peut-être en 2014 (ou 2015) téléchargé les datasets afin d'essayer de les résoudre. S'ils veulent en 2016, déposer leurs solutions ils ne trouveront pas les datasets correspondants. Grâce à cet identifiant unique, ce problème sera évité.

Ce développement sera effectué en même temps que l'implémentation des datasets privés, afin de n'avoir qu'une seule migration à faire sur la base de données.

2.5 Modification du site pour accepter plusieurs fonctions objectifs pour certains problèmes

2.5.1 Système existant

Pour l'instant dans l'application, pour vérifier les instances d'un dataset et vérifier qu'ils sont corrects le système utilise un fichier .xsd et vérifie que tous les nœuds XML requis sont présents dans chacune des

instances.

Cette modification n'est pas prioritaire et mon encadrant préfère que je me concentre sur l'implémentation des datasets privés.
Actuellement, le diagramme de classe de l'application est le suivant :

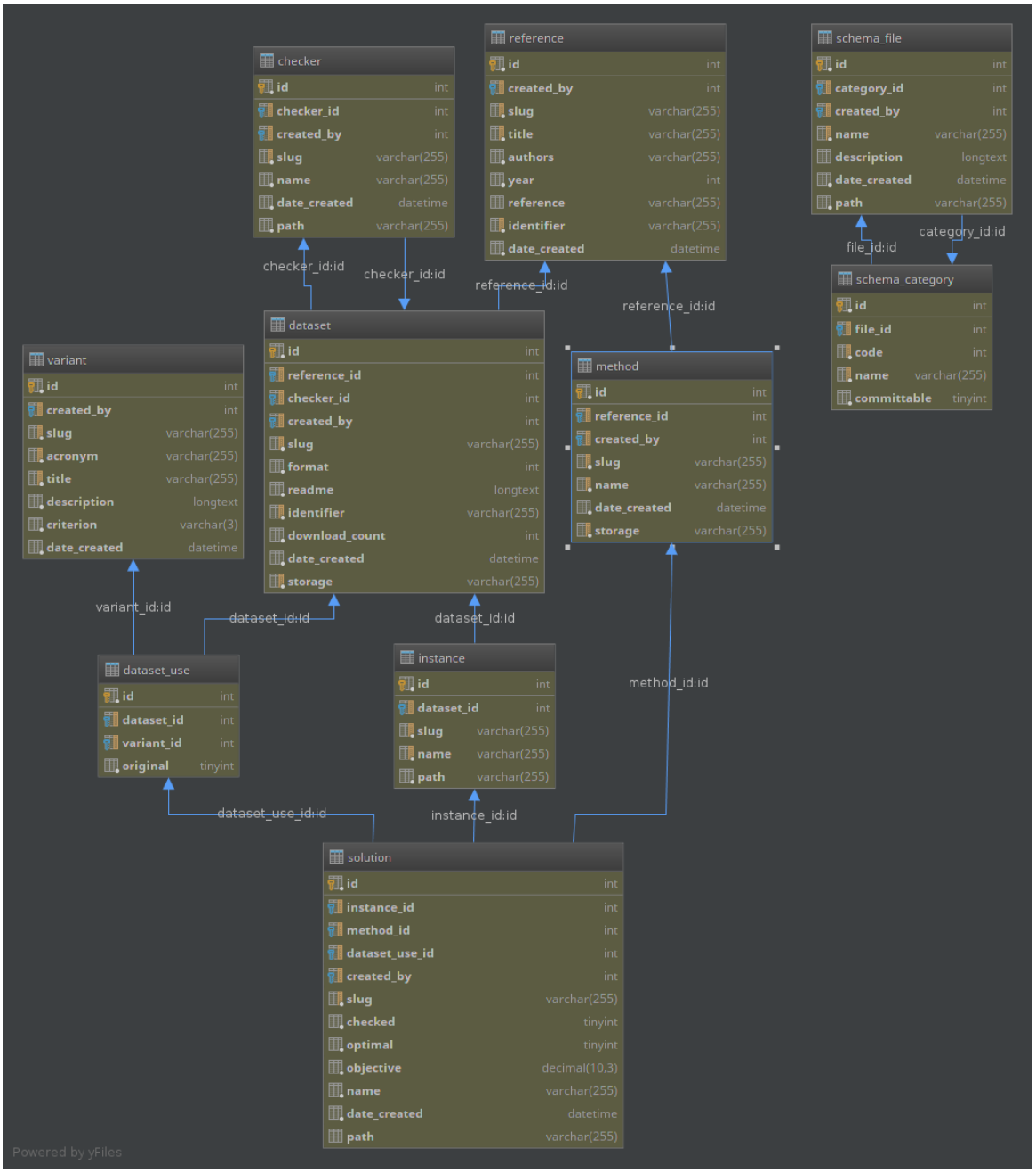


Figure 1 – *Diagramme de classe réduit*

J'ai volontaire épuré le diagramme pour garder uniquement les classes qui nous intéressent.

Sur ce diagramme, nous pouvons remarquer que l'objectif est un attribut de la classe *Solution*. Or avec ce système, il n'est pas possible d'avoir plusieurs fonctions objectives.

Il va falloir adapter le schéma de la base de données pour pouvoir avoir la possibilité de choisir entre 1 et n critères et leurs valeurs.

2.5.2 Amélioration prévue

Durant la partie développement, je vais modifier le schéma de la base de données afin de pouvoir avoir entre 1 et n critères pour un problème donné.

Il me faudra adapter le système en conséquence :

- Dans la BSK (Best Solution Known) table, il faudra afficher tous les critères ainsi que leurs valeurs, puis pouvoir trier en fonction du critère choisi.
- Générer les fichiers de vérification de solution en fonction des critères du problème.
- Autoriser les utilisateurs à uploader des datasets avec des instances qui possède plusieurs critères.

3 Symfony2

1 Qu'est-ce que Symfony ?

[WWW12] Symfony est un framework MVC [1] PHP développé par l'agence web française SensioLabs.
D'après SensioLabs :

> Symfony est un ensemble de composants PHP, un framework pour application web, une philosophie, ainsi qu'une communauté - le tout fonctionnant en harmonie.

[WWW4]Symfony utilise Doctrine. Doctrine est un ORM (Object Relational Mapper).
Un ORM, Mapping objet-relationnel en français, permet de créer l'illusion d'avoir une base de données orientée objet à partir d'une base de données relationnelle. Cela permet le plus souvent une abstraction du langage SQL.

[WWW5]Doctrine propose aussi outre l'ORM, ce qu'ils appellent DBAL pour The Doctrine database abstraction & access layer. DBAL est une puissante couche d'abstraction de base de données avec de nombreuses fonctionnalités pour la gestion des schémas et l'abstraction de PDO.
Doctrine est vraiment très puissant, il permet notamment de gérer la création de tables, les migrations et les clés primaires sans écrire de requêtes SQL, par exemple avec des annotations.

Symfony est utilisé par de nombreux projets qu'ils soient petits, moyens ou même grands.
Voici quelques projets utilisant Symfony :

- Drupal qui est un CMS open-source
- phpBB qui est un forum open-source
- Laravel qui est un framework PHP qui utilise des composants de Symfony
- Magento qui est une plateforme de commerce électronique libre

2 Les composants Symfony

Comme dit précédemment Symfony est un ensemble de composants. Il est possible d'utiliser ces composants séparément dans des projets externes, c'est ce qui fait la force de Symfony et le rend extrêmement modulable.

1. Modèle Vue Controlleur

[WWW10]Voici une liste non exhaustive des composants les plus importants ainsi que leurs descriptions :

Asset Prend en charge la génération d'URL et les versions des assets (ressources) telles que les feuilles de style CSS, les fichiers JavaScript et les images.

Debug Fournit des outils complémentaires d'aide au débogage en PHP. Ce qui permet par exemple d'afficher le contenu d'un objet, d'un tableau ou d'une variable dans une barre de débuggage.

EventDispatcher Le composant EventDispatcher fournit des outils qui permettent à vos composants de l'application de communiquer avec les autres par la distribution d'événements et en les écoutant.

Form Fournit les outils pour concevoir, traiter et réutiliser des formulaires HTML.

Templating Fournit les outils nécessaires pour concevoir des systèmes de rendus (templates). Symfony permet d'utiliser le moteur de template Twig.

Validator Fournit les outils pour valider des classes, des objets et des données. Des règles de base sont disponibles, mais il est possible d'en ajouter d'autres.

HttpFoundation Contient les classes *Request* et *Response*.

3 Organisation générale de Symfony

[WWW11]Symfony se base sur le design pattern MVC, et il utilise le moteur de template Twig bien qu'il ne soit pas obligatoire. Il est possible d'utiliser une page PHP simple mais cela sera moins performant et moins intuitif.

Symfony a créé des classes pour gérer les requêtes et réponses HTTP, qui sont *Request* et *Response*. Il y a aussi une classe permettant de gérer le Routing.

D'après [WWW14], voici un schéma du déroulement d'une application Symfony :

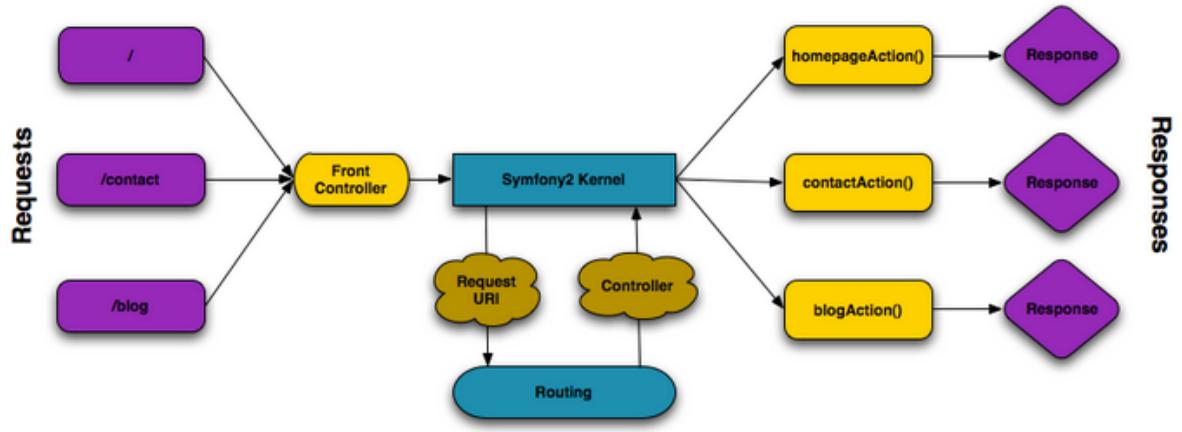


Figure 1 – *Déroulement d'une application Symfony*

Chaque URL est associée à une méthode d'un contrôleur qui doit retourner un objet de type *Response*.

3.1 Les bundles

[WWW17]Symfony est organisé en bundles. Un bundle est similaire aux plugins que l'on peut trouver dans d'autres logiciels, mais la différence est que dans Symfony tout est bundle.

Un bundle est simplement un ensemble structuré de fichiers au sein d'un répertoire et qui implémentent une fonctionnalité unique.
Chaque répertoire contient tout ce qui est lié à cette fonctionnalité incluant les fichiers PHP, les templates, les feuilles de style, le JavaScript.

Les bundles ont une structure des répertoires similaires pour garder le code homogène.
Voici cette structure :

- **Controller/** : contient les contrôleurs du bundle ;
- **DependencyInjection/** : contient certaines classes d'extension d'injection de dépendances ;
- **Resources/config/** : contient la configuration, notamment la configuration de routage (ex routing.yml) ;
- **Resources/views/** : contient les templates organisés par nom de contrôleur ;
- **Resources/public/** : contient les ressources web (images, feuilles de style, etc.) et sont copiées ou liées par un lien symbolique dans le répertoire de projet web/ grâce à la commande assets :install ;
- **Tests/** contient tous les tests du bundle.

3.2 La structure des répertoires

Les répertoires sont entièrement flexibles, mais chaque application a par défaut la même structure de répertoire :

```
app/ ..... Ce répertoire contient la configuration de l'application
    config/ ..... Contient les fichiers de configuration
        config.yml ..... Paramètres de configuration
        parameter.yml ..... Paramètres spécifiques à une installation (non versionné)
        route.yml ..... Fichier de route principal
    Resources/
        views/ ..... Contient les vues (Twig) non spécifiques à un bundle
src/ ..... Tout le code PHP du projet est stocké ici
vendor/ ..... Par convention, toutes les bibliothèques tierces (additionnelles) sont placées ici
web/ ..... Le répertoire racine web qui contient tous les fichiers publiquement accessibles
    css/ ..... Contient toutes les feuilles de style CSS
    images/ ..... Contient toutes les images
    js/ ..... Contient tous les scripts JS
composer.json ..... Liste des dépendances externes
```

3.3 Composer

[WWW3]Symfony utilise Composer comme gestionnaire de dépendances. Composer est open-source et écrit en PHP.
Pour ajouter une bibliothèque ou un bundle, il faut l'ajouter au fichier *composer.json*. Il est possible de choisir la version, ou même de choisir la dernière version stable.

Les deux principales commandes sont :

```
composer install
```

Cette commande permet de télécharger toutes les bibliothèques contenues dans le fichier *composer.json*. Ainsi que

```
composer update
```

Cette commande permet de mettre à jour les bibliothèques (par exemple, un changement de version).

4 Les contrôleurs

Symfony étant un framework MVC, il implémente ses propres contrôleurs.
D'après [WWW15], un contrôleur est un fichier PHP prenant les informations de la requête HTTP et retourne une réponse HTTP (grâce à l'objet **Response** de Symfony).
Le contrôleur contient toute la logique dite métier de la page demandée. Sur Symfony, chaque méthode du contrôleur doit retourner un objet **Response**.
Les contrôleurs ainsi que leurs méthodes doivent avoir un nom formaté comme ceci : xController (avec x le nom du contrôleur souhaité) et yAction (avec y le nom de l'action souhaité).

Il bien évidemment possible d'afficher une vue avec des paramètres à la fin d'une méthode.
Il est possible de faire comme ceci (cite) :

```
// renders app/Resources/views/hello/index.html.twig
return $this->render('hello/index.html.twig', array('name' => $name));
```

Les vues Symfony sont soit en .php soit en .twig, Twig est privilégié, car plus puissant.

5 Les vues

Nous allons ici détailler les vues au format Twig. Twig est un moteur de template destiné uniquement à l'affichage de page HTML, il ne permet pas de faire de code métier comme le PHP. L'avantage est par exemple, pour les web designer qui auront plus de facilité à écrire des templates Twig que du PHP.

D'après [WWW16], il existe trois types de syntaxe :

{{ maVariable }} : affiche le contenu de la variable nommée maVariable

{% ... %} : est utilisé pour la logique du template (par exemple les boucles for, les instructions if/else, etc)

{# ... #} : permet d'inclure des commentaires qui ne seront pas affichés

Twig est également capable de gérer les objets, par exemple {{monObjet.id}} va afficher la valeur de l'attribut **id** de l'objet **monObjet**.

Twig permet de gérer facilement l'inclusion de fichier CSS et JavaScript.
Il est possible de faire comme ça pour le CSS :

```
{% block stylesheets %}
    <link rel="stylesheet" href=" {{  asset('css/pages/stylesheets/x.css')}}"/>
{% endblock %}
```

Et comme ça pour le JavaScript :

```
{% block javascripts %}
  <script type='text/javascript' src='{{ 'js/pages/js/y.js' }}'></script>
{% endblock %}
```

4 Étude de l'application

L'application a été créée sur Symfony2 que j'ai détaillé dans ce chapitre : Chapitre 3.
L'application utilise des bundles externes comme Sonata User Bundle et Sonata Admin Bundle .

Les autres parties ont été développées à la main grâce a tout l'éco système Symfony.
Dans Symfony, il est possible d'utiliser plusieurs techniques pour configurer le comportement de l'application.

[WWW13]Par exemple le routing, on peut associer une méthode d'un certain contrôleur à une route donc une vue selon plusieurs moyens qui sont :

- Annotations
- Fichier YAML [1]
- Fichier XML [2]
- Fichier PHP

Dans cette application, très souvent les annotations sont utilisées. Si cela n'est pas possible, on utilisera le *YAML* ou le *XML*.
Pour le routing, uniquement les annotations sont utilisées, donc pour retrouver la vue associée à une méthode, il faudra regarder les annotations au-dessus de celle-ci.

L'application utilise la version 2.5.7 de Symfony. Ce qui est dommage, car à partir de la version 2.7, une méthode **dump()** du bundle **Debug** de Symfony est apparu.
Cette méthode est une alternative à **var_dump()** et est vraiment utile pour faire du débogage, car affiche toutes les informations dans la barre de débug même s'il y a eu un rechargement de la page entre les deux.

Malgré cette méthode, cela ne vaut pas le coup d'effectuer la migration. En effet, cela entrainerait trop de problèmes pour un gain faible, d'autant plus que cette fonction serait utile uniquement pour la phase de développement.

1 Diagramme de la base de données

1. YAML Ain't Markup Language
2. eXtensible Markup Language

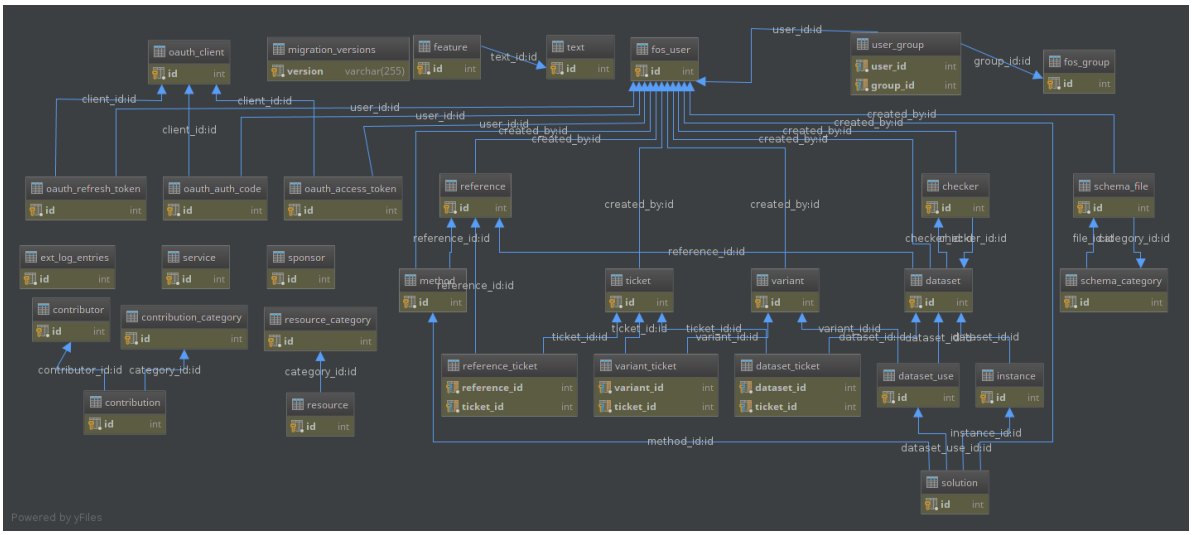


Figure 1 – *Diagramme de classe de l'application VRP-REP*

Certaines classes ne sont pas utilisées, mais ont été créées en prévision d'une intégration future.
Je vais ici détailler certaines classes afin d'expliquer à quoi elles correspondent :

Solution : représente une solution. Une solution est définie pour une instance, un dataset use, et une méthode. La table permet entre autres de savoir si la solution est validée, ainsi que si elle est optimale. On peut aussi connaitre la valeur de la fonction objectif.

Instance : une instance est un problème qui appartient à un dataset. Une instance possède un fichier qui décrira les différents noeuds et arcs du problème.

Dataset : représente un ensemble d'instances qui sont sur un problème similaire. Un dataset est forcement lié à une méthode.

Variante : pour un problème donné, il peut y avoir plusieurs variantes. Une variante est liée à un dataset use.

Method : ce sont les différents algorithmes de résolution de problème. Pour chaque méthode, on connait sa *Reference* ainsi que la liste des *Solution* trouvées grâce cette méthode.

Reference : représente un article sur des algorithmes de résolution du problème. Grâce a une *Reference*, on peut avoir les dataset et méthodes associés.

Ticket : ce sont les objets qui représenteront les tickets envoyés par les utilisateurs au support.

Sponsor : représente un sponsor, une personne ou une organisation qui soutient le projet

Ressource : ce sont les différentes ressources mises à disposition par le comité pour utiliser le site (en particulier ici des tutoriels vidéos)

Contributor : représente un contributeur, un contributeur est quelqu'un qui a participé au développement du site (membre du comité, spécification des instances, développeur du site, etc)

Contribution : représente une contribution, sauvegarde la personne à l'origine de la contribution, la catégorie (site web, comité, etc.) et éventuellement une mention (Lead Developper)

2 Sonata

2.1 UserBundle

[WWW18]Le bundle User de Sonata s'appuie sur le FOSUserBundle tout en ajoutant d'autres fonctionnalités.

FOSUserBundle a été développé par FOS [3] qui est une communauté de développeur PHP qui développe des bundles pour Symfony2.

3. FriendsOfSymfony

[WWW9, WWW19]FOSUserBundle permet une gestion poussée des utilisateurs.
Ce bundle propose un certain nombre de fonctionnalités ce qui fait qu'il est très flexible :

— Possibilité d'utiliser Doctrine ORM ou MongoDB

— Possibilité d'enregistrer les utilisateurs avec éventuellement une confirmation par email

— Possibilité de réinitialiser le mot de passe

— Définition des profils de validation avec des règles pour chaque champ du formulaire

Il est possible de redéfinir les configurations par défaut grâce au fichier *config.yml*. Si on souhaite redéfinir un fichier en particulier, il faut recréer le fichier dans la même architecture que le bundle (du dossier *vendor/*) dans le dossier *src/*.

2.2 AdminBundle

Sonata Admin bundle est un bundle permettant une administration poussée de la base de données grâce à une interface graphique.

Pour le VRP-REP, Sonata ressemble à ça :

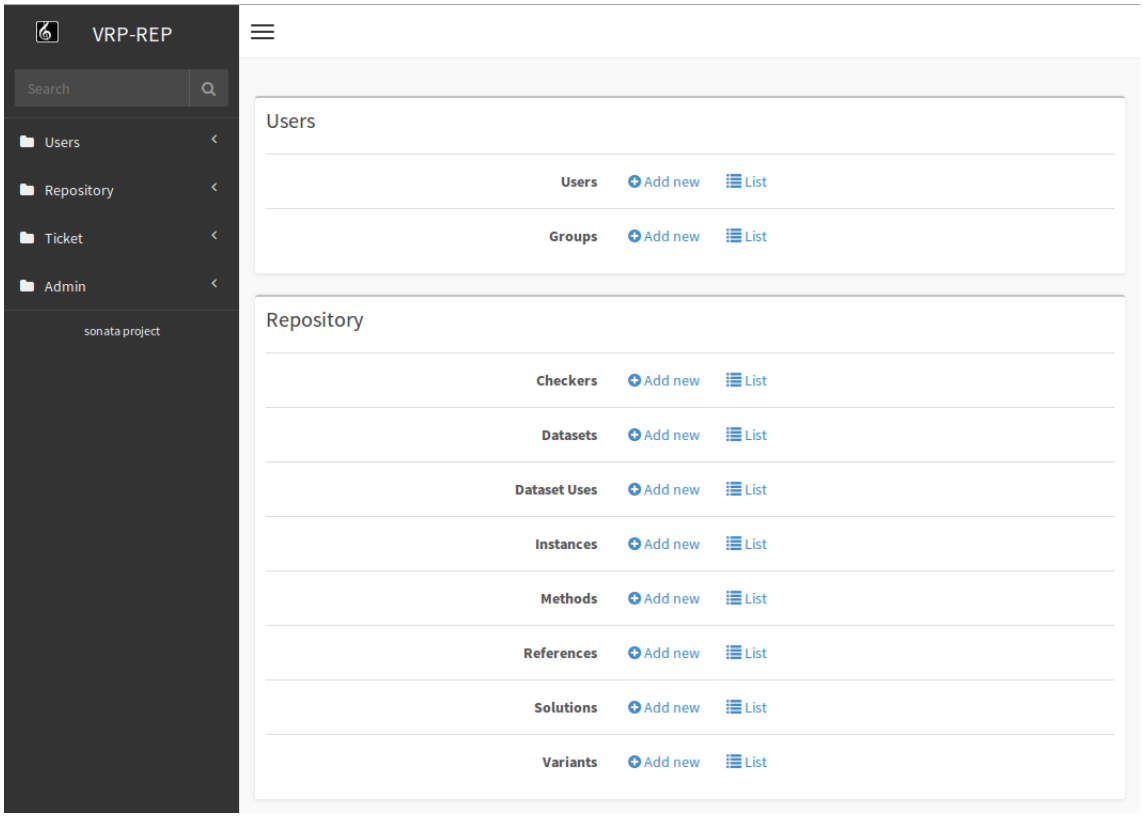


Figure 2 – *Interface de l'administration de Sonata*

3 FileSetBundle

Le FileSetBundle a été créé par Hubert Lobit qui est l'ancien développeur du projet. Ce bundle permet la gestion d'ensemble de fichiers avec Symfony2.

Ses fonctionnalités sont :

— Gestion générale de l'ensemble (Ajouter des fichiers, supprimer des fichiers, etc)

— Dézipper une archive

— De vérifier les fichiers suivant des règles (par exemple typeMime)

Ce bundle est utilisé pour l'upload de dataset. Il permet de dézipper les archives si besoin et de créer un ensemble de fichiers validé par les règles du validateur.

4 AdminBundle

Le bundle AdminBundle sert à créer une zone admin à l'interface un peu plus cohérente avec celle du site.
Ce bundle sera à mettre à jour suivant les modifications de l'application afin de pouvoir l'administrer correctement.

L'interface est la suivante :

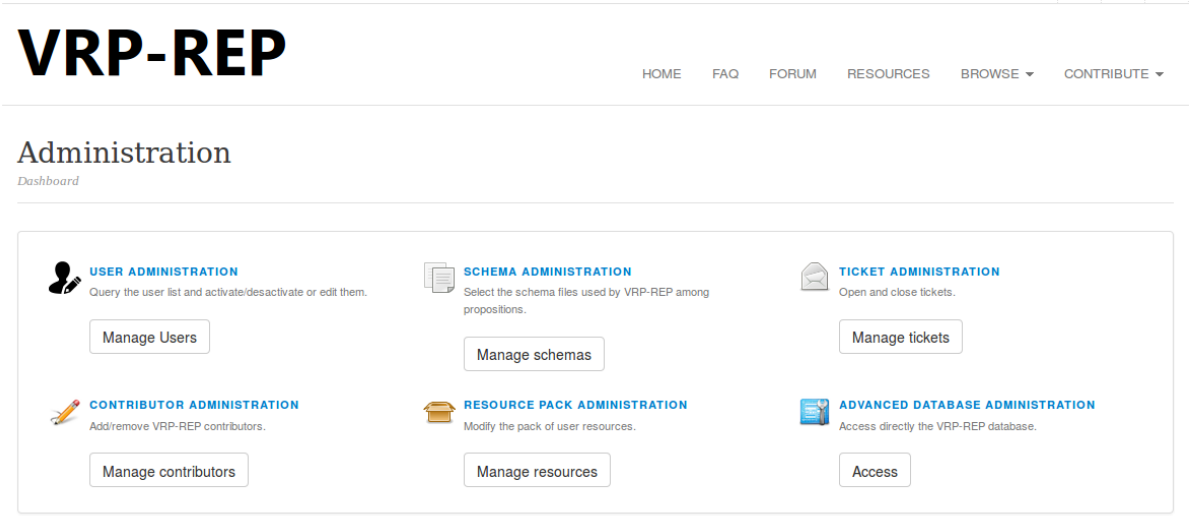


Figure 3 – *Interface de l'administration VRP-REP*

Ce bundle dispose de 4 contrôleurs :

— ContributorsController

— ResourcesController

— SchemasController

— UsersController

4.1 Controllers

Les contrôleurs suivent plus ou moins tous le même schéma. Ils disposent d'un certain nombre de méthodes avec leur vue associée :

dataAction : retourne tous les entités (ex : *Contributor*) sous la forme d'une *JsonReponse*

editAction : prend en paramètre une entité (exemple : *Contributor*) et la met à jour en base de données

addAction : ajoute une entité (exemple : *Contributor*) en base de données

removeAction : supprime une entité (exemple : *Contributor*) en base de données

4.2 Entity

Les entités de Doctrine utilisent elles aussi les annotations pour définir les différents champs ainsi que les caractéristiques des ceux-ci.
Dans l'application, en accord avec Doctrine, il y a les entités et les *Repository* qui eux feront toutes les requêtes de type *SELECT*.

Dans ce bundle, il y a les entités suivantes (cette liste n'est pas exhaustive, l'important est de montrer le raisonnement suivi) :

Contributor : représente un contributeur

Contribution : représente une contribution

ContributionRepository : contient une méthode : *findAllByName()* qui retourne un tableau de *Contribution*

Sponsor : représente un sponsor

SponsorRepository : contient actuellement une méthode : *findAllEnabledByPriority()* qui retourne tous les sponsors qui sont actifs par ordre de priorité. Un tableau de *Sponsor* trié est retourné

5 RepositoryBundle

Ce bundle est le bundle principal de l'application. Il contient tous les fichiers nécessaires au bon fonctionnement de l'application.

C'est principalement dans ce bundle que je serai amené à faire des modifications. Bien entendu, je ne vais pas décrire tous les fichiers ni tous les contrôleurs ou entités, mais je vais en décrire quelques-uns ainsi qu'une méthode efficace pour rechercher l'information dans le code.

J'aborderai plusieurs points dans cette partie :

- Les commandes
- Les contrôleurs
- Les entités
- Les évènements
- Les listeners
- Les ressources

5.1 Commandes

Comme expliqué dans le chapitre sur Symfony, les commandes sont des éléments utiles dans une application Symfony et peuvent être appelées depuis la console Symfony.
Par exemple comme ceci :

```
1 php app/console nom:de:la:commande
```

Le repository bundle définit deux commandes :

ChangePathsCommand : cette commande modifie tous les chemins de la base de données. Elle transforme les chemins absolus en chemins relatifs. Elle s'appelle avec le nom : *repository :paths :change*

CleanDatabaseCommand : utilise le service *DatabaseCleaner* pour supprimer toutes les données non utilisées de la base de données. Cette commande ne supprime que les datasets, variantes, références et méthodes. Elle s'utilise grâce au nom *repository :database :clean*

La commande *ChangePathsCommand* me servira lors de la migration de l'application vers le VPS.

5.2 Contrôleurs

Le contrôleur le plus gros en terme de méthode est le *DataSetController* car il doit gérer l'upload, la vérification des datasets, ajout/modification/suppression, affichage des datasets ainsi que des instances à l'intérieur d'un dataset.
Voici une liste des contrôleurs de ce bundle :

DatasetController comme dit ce contrôleur est le plus conséquent, il permet entre autres l'upload des datasets avec la vérification des instances à l'intérieur, l'ajout/la modification et la suppression de dataset. Il propose aussi le téléchargement. Ce contrôleur est en lien avec le bundle *FilSetBundle*.

MainController : ce contrôleur propose uniquement la récupération de données et l'affichage des pages "annexes" comme la liste des contributeurs (/contributors.html), la FAQ [4], la liste des sponsors (/sponsors.html), etc.

MethodController : permet de récupérer toutes les méthodes (*dataAction()*) ainsi que de les supprimer (*removeByIdAction(Method $method)* et *removeAction(Method $method)*)

ReferenceController : ce contrôleur propose la possibilité de récupérer les *Reference* sous la forme d'une *JsonResponse* grâce à la méthode *dataAction()*. Il permet aussi la modification d'une *Reference* et l'affiche d'une *Reference* avec les données associées (variantes, dataset résolus, etc)

RessourceController : permet uniquement le téléchargement de la *Ressource* grâce à la méthode *downloadAction(Ressource $ressource)*

SchemaController : la méthode *commitAction()* permet de proposer un schéma pour la spécification d'une instance. Cette méthode permet soit de créer le formulaire et de l'afficher ou alors si l'utilisateur a déjà soumis une requête, elle effectue le processus derrière. L'autre méthode (*downloadAction(SchemaFile $schema)*) permet de télécharger un schéma.

SolutionController : ce contrôleur est un autre contrôleur conséquent en termes d'action. Il fait la même chose que le *DatasetControlleur* mais pour les *Solutions*.

VariantController : permet de gérer tout ce qui est relatif au Variante.

5.3 Entités

Les entités suivantes sont présentes dans ce bundle :
- Checker
- Dataset
- DatasetUse
- Instance
- Method
- Reference
- Solution
- Variant

avec leurs *Repository* associées.

5.4 Événements

Ce bundle définit deux événements qui sont :

DatasetEvent : c'est l'objet qui sera envoyé lors d'un événement sur les datasets (création, modification, etc).

SchemaFileEvent : représente l'objet retourné lors d'un événement sur les schémas des fichiers.

4. Frequently Asked Questions

5.5 Listeners

Les listeners sont là pour exécuter des actions lorsque certains événements surviennent. Les listeners sont les suivants :

FlashMessageListener : permet d'afficher un message flash suivant plusieurs événements (onDatasetAdded, onSchemaCommitted, onSolutionsAdded, onContactSubmitted)

NotifyCommitteeListener : permet d'envoyer un mail au comité suivant plusieurs événements (onTicketSubmitted, onSchemaCommitted, onTicketClosed)

5.6 Ressources

Pour finir, les ressources de ce bundle.
Dans les ressources, il y a :

- les configs, en particulier la définition des listeners.
- les traductions pour ce bundle
- pour finir, les templates twigs rangés par contrôleur.

6 TicketBundle

Ce bundle sert à gérer les tickets que les utilisateurs peuvent envoyer au support de site VRP-REP.

7 Bonne pratique de développement

De mon expérience, afin de développer sur ce projet, il peut-être intéressant d'utiliser un IDE [5]. Je conseille PhpStorm de JetBrains, il est gratuit pour les étudiants ou les projets open-source.

PhpStorm couplé au plugin Symfony2 permet par exemple depuis une méthode d'un contrôleur de passer directement à la vue associée. Concernant Git, il peut-être intéressant d'utiliser *GitFlow* qui rend

le développement plus simple sur de petites fonctionnalités. Voici un site illustrant le workflow de Git Flow GitFlow

5. Integrated Development Environment

5 Modélisation, proposition et solution apportée

1 Proposition

Comme décrit plus haut, M. Mendoza m'a fait part de son problème d'upload et vérification de dataset. En effet, il n'était pas possible d'envoyer de gros dataset (150 Mo de XML Zipé), de plus il n'y avait pas de gestion des messages d'erreur, donc l'application tournait en boucle sans prévenir l'utilisateur. Cela est dû à des limitations de l'hébergement mutualisé de chez 1&1 qui bride les configurations du php.ini.

J'ai donc proposé à M. Mendoza de migrer le site sur un VPS, cela coûtera un peu plus cher, mais il aura le total contrôle sur le serveur. On pourra débrider PHP à souhait, il sera possible d'installer n'importe quelle application (Java, NodeJs, Ruby, etc.) si par la suite un changement de technologie est envisagé.
Afin de tester la viabilité de ce projet et être sûr que cela va résoudre les problèmes, j'ai proposé à M. Mendoza d'installer l'application sur mon VPS et de la tester dessus avant de commander le leur.
Une fois le fonctionnement validé, j'ai réalisé une étude des différents VPS sur le marché (Chapitre 6).

Concernant la gestion de l'information pour les utilisateurs, j'ai proposé d'inclure un loader pendant la validation des instances (cette validation peut prendre pas mal de temps suivant la taille des instances), ainsi que l'affichage des messages d'erreurs retournés par l'application en cas d'erreur.

J'ai également trouvé quelques bugs que je pourrais résoudre, et des améliorations qui me semblent intéressantes à mettre en place et que je pourrais proposer. J'ai détaillé ces tâches dans le cahier des charges ici : subsection 2.2 (Chapitre 2).

2 Modélisation

2.1 Multi critère

Lors de mon rendez-vous avec M. Mendoza et Hubert Lobit, nous avons réfléchi à 3 afin de faire un diagramme de classe pour répondre à la problématique du multicritère.
Le diagramme résultant est le suivant :

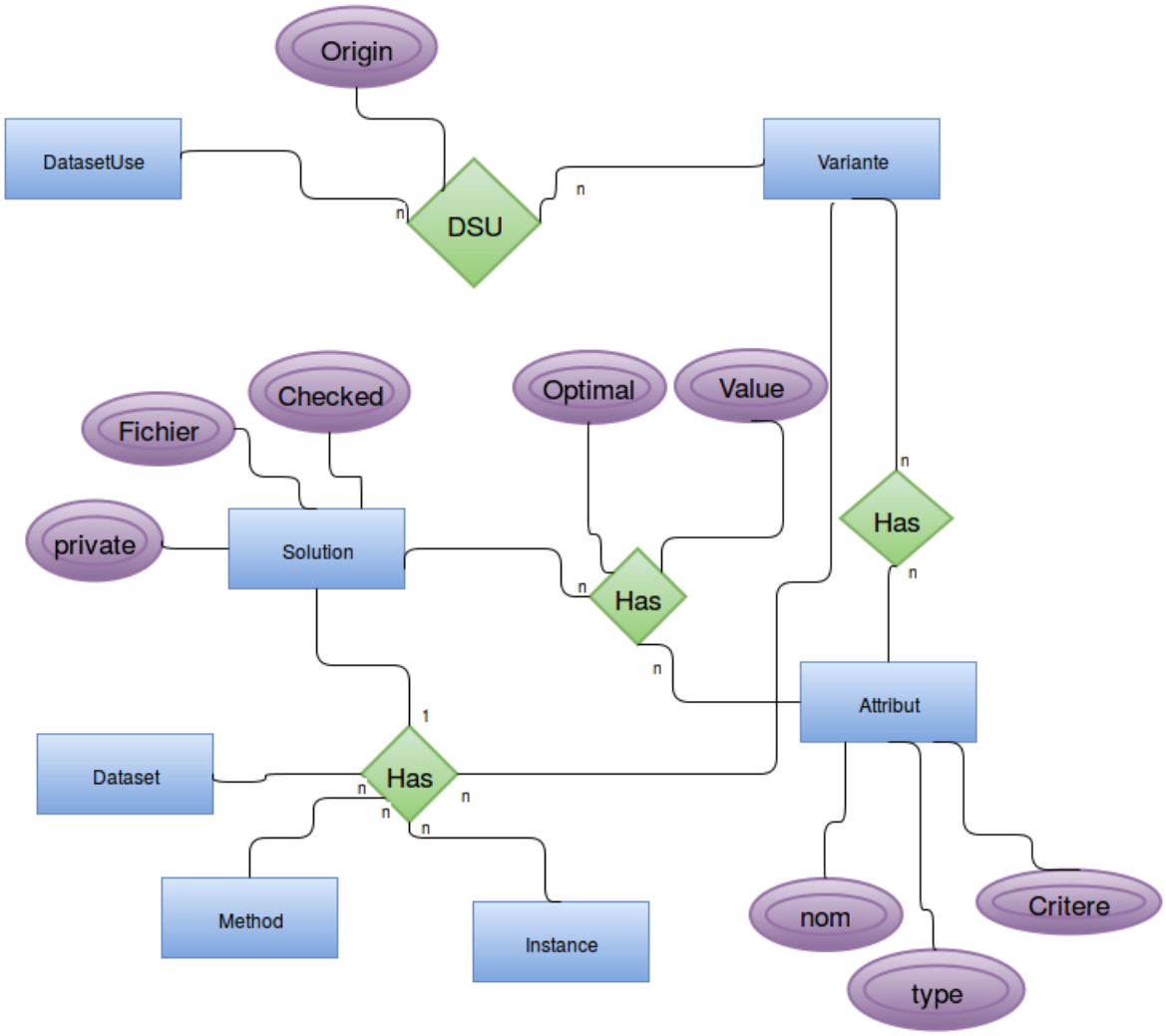


Figure 1 – *Diagramme de classe pour le multi-objectif*

Il faudra donc créer une table **Attribut** avec les champs nom, critère, et type. Le nom va permettre de l'identifier, le critère est est le critère du problème et le type est par exemple *MIN* ou *MAX*.
Un **Attribut** est lié avec une solution de façon n-n, le résultat de cette jointure permettra de connaitre la valeur de l'attribut et s'il est optimal ou non. Il peut donc y avoir plusieurs attributs par solution. Un **Attribut** est aussi lié à une variante.

La grosse jointure qui est Instance - Method - Dataset - Variante possède une Solution est la même qu'avant. Ce qui change surtout c'est l'ajout de la table Attribut avec toutes les modifications qui en découlent.

Il faudra faire plusieurs migrations pour adapter les données au nouveau modèle de la base de données.

6 Étude des différents VPS

Une de mes tâches durant cette phase de recherche était de faire une étude des différentes offres de VPS disponibles sur le marché se basant sur plusieurs critères.
Je devais faire une recommandation à mon encadrant quant à l'offre qui me paraissait la plus adaptée à notre cas.

1 Comparatif

1.1 1&1

[WWW1]J'ai ciblé 3 offres chez 1&1 car les autres sont hors budget :

- 1 vCore, 1Go RAM, 50 Go HDD, Trafic illimité : 5,99 TTC avec réduction sinon 10.79 TTC
- 2 vCore, 2Go RAM, 150 Go HDD, Trafic illimité : 9.59 TTC avec réduction sinon 15.59 TTC
- 4 vCore, 4Go RAM, 300 Go HDD, Trafic illimité : 11.99 TTC avec réduction sinon 23.99 TTC

Les réductions s'appliquent sous la condition suivante : engagement de 6 mois . À l'issue de ces 6 mois, les tarifs habituels s'appliquent. Il est important de noter qu'il y a des frais de mise en service de 9,99 € HT (11,99 € TTC).

Il y a le nom de domaine ainsi que 250 comptes email inclus.

1.1.1 Avantages

- Espace disque important (HDD)
- Nom de domaine inclus
- Email inclus

1.1.2 Inconvénients

- Prix élevé
- Pas de SSD
- Peu de RAM pour le prix
- Réduction : Engagement de 6 mois, donc prix réduit pour 6 mois puis prix normal

1.2 OVH

[WWW7]J'ai ciblé aussi 3 offres chez OVH car les autres sont hors budget :

- 1 vCore (2,4GHz), 2 Go RAM, SSD 10 Go : 2,99 HT (3,59 TTC)
- 1 vCore (2,4 GHz), 4 Go RAM, SSD 20 Go : 5,99 HT (7,19 TTC)
- 2 vCore (2,4 GHz), 8 Go RAM, SSD 40 Go : 11,99 HT (14,39 TTC)

1.2.1 Avantages

- Pas cher
- Présence d'un SSD
- Possibilité de prendre au mois sans changement de tarif
- Possibilité de prendre le nom de domaine chez eux

1.2.2 Inconvénient

- Pas beaucoup de stockage

1.3 BeHost

[WWW2]J'ai ciblé 4 offres chez BeHost :

- LVM01 : 2 vCore, 2 Go RAM, 20 Go d'espace disque, 100 Mbps : 1,39 HT (1.67 TTC)
- LVM02 : 3 vCore, 2 Go RAM, 30 Go d'espace disque, 100 Mbps : 1,99 HT (2.39 TTC)
- LVM03 : 3 vCore, 3 Go RAM, 40 Go d'espace disque, 100 Mbps : 2,99 HT (3.59 TTC)
- LVM04 : 3 vCore, 4 Go RAM, 50 Go SSD, 100 Mbps : 3,99 HT (4.79 TTC)

1.3.1 Avantages

- Pas chère (moins chers qu'OVH)
- Présence d'un SSD
- A priori bon rapport qualité-prix
- Pas mal de stockage (LVM03)

1.3.2 Inconvénient

- Peu connu (Downtime ? Maintenance ? Support ?)
- Pas possible de prendre le nom de domaine chez eux.
- Graphique des performances pas terrible (Simple image)

1.4 Pulseheberg

[WWW8]J'ai ciblé 3 offres chez Pulseheberg :

- VPS S : 1 vCore (2,4 GHz), 1 Go RAM, 50 Go HDD OU 16 Go SSD, 100 Mbps, 5Tb Trafic : 1,49 TTC / mois

— VPS M : 4 vCore (2,4 GHz), 4 Go RAM, 100 Go HDD OU 64 Go SSD, 1 Gbps (Connectivité 1Gbps en "best-effort" limitation à 100Mbps après 10Tb de données échangées) , 10Tb Trafic : 2,99 TTC / mois

— VPS L : 6 vCore (2,4 GHz), 8 Go RAM, 200 Go HDD OU 128 Go SSD, 1 Gbps (Connectivité 1Gbps en "best-effort" limitation à 100Mbps après 10Tb de données échangées) , 10Tb Trafic : 5,99 TTC / mois

1.4.1 Avantages

— Bon rapport qualité / prix
— Très bonne connexion (VPS M & L)
— Beaucoup de stockage

1.4.2 Inconvénient

— Peu connu (Downtime ? Maintenance ? Support ?)
— Pas possible de prendre le nom de domaine chez eux.
— Mauvaise réputation sur internet

2 Tableau comparatif

Table 1 – *Comparatif des différents VPS*

Nom	Processeur	RAM	Disque	SSD	Prix
OVH 1	1 vCore 2,4 GHz	2 Go	10 Go	Oui	3,59 TTC
OVH 2	1 vCore 2,4 GHz	4 Go	20 Go	Oui	7,19 TTC
OVH 3	2 vCore 2,4 GHz	8 Go	40 Go	Oui	14,39 TTC
1&1 1	1 vCore	1 Go	50 Go	Non	10.79 TTC
1&1 2	2 vCore	2 Go	150 Go	Non	5.59 TTC
1&1 3	4 vCore	4 Go	300 Go	Non	23.99 TTC
PH : S	1 vCore 2,4 GHz	1 Go	50 Go HDD / 16 Go SSD	Oui	1,49 TTC
PH : M	4 vCore 2,4 GHz	4 Go	100 Go HDD / 64 SSD	Oui	2,99 TTC
PH : L	6 vCore 2,4 GHz	8 Go	200 Go HDD / 128 Go SSD	Oui	5,99 TTC
BH LVM01	2 vCore	2 Go	20 Go HDD	NC	1.67 TTC
BH LVM02	3 vCore	2 Go	30 Go HDD	NC	3.59 TTC
BH LVM03	3 vCore	3 Go	40 Go HDD	NC	3.59 TTC
BH LVM04	3 vCore	4 Go	50 Go SSD	NC	4.79 TTC

Ayant moi-même un VPS chez OVH et n'ayant pas eu de problème, j'ai donc conseillé à M. Mendoza de

prendre un VPS chez OVH.
De plus, certains hébergeurs low-cost décrits au-dessus ont une très mauvaise réputation sur internet (beaucoup de personnes ont eu des problèmes de downtime par exemple).
Pour l'instant la première offre semblant suffisante, il est possible d'améliorer l'offre tout en gardant l'installation du VPS donc pas de soucis de ce côté.

7 Gestion de projet

Dans cette partie, je vais effectuer un planning prévisionnel, un point sur les méthodologies et outils pour le suivi .

1 Planning prévisionnel

Je vais rappeler les différentes tâches à faire pour ce projet :

- Migration de l'application sur le VPS
- Résolution de bug
- Amélioration de l'application (Extension dynamique des instances, etc)
- Dataset privé
- Changement d'URL des datasets
- Multi-objectif

Je vais maintenant détailler chacune des tâches :

Migration de l'application sur le VPS

Cette tâche est référencée ici : subsection 2.1 (Chapitre 2).
Cette tâche est délicate, car elle va rendre le site indisponible pendant certains temps. Je vais devoir faire attention à reproduire exactement le même comportement sur le VPS, cela passera par une phase de test.

Pour cette tâche, j'utiliserai le guide de migration effectuée Chapitre 11. Je vais donc faire l'installation des technologies nécessaires (Apache, PHP, MySQL, Git, Composer) puis migrer l'application, restaurer les backups. Pour finir, il me faudra faire une crontab pour automatiser les backups et déplacer les logs d'apache dans un dossier plus accessible que "/var/log/apache/".

J'estime que cette tache me prendra 1 semaine.

Résolution de bug

Cette tâche est une tâche que j'ai proposée à mon encadrant (subsection 2.2 (Chapitre 2)). Lorsque j'ai testé l'application, j'ai découvert quelques bugs restants, j'ai donc proposé à mon encadrant des solutions pour les résoudre.

Les objectifs de cette tâche sont susceptibles d'évoluer dans le temps si d'autres comportements anormaux surviennent. Pour l'instant, je vais ajouter de la validation côté serveur, ajouter une gestion des erreurs lors de l'upload de dataset. Pour finir, lorsqu'une instance est envoyée sur le serveur, l'application sauvegardait le chemin absolu, je vais donc le changer pour avoir le chemin relatif.

Cette tâche est actuellement estimée à 2 semaines . Si d'autres bugs sont trouvés en accord avec mon encadrant nous ajusterons le planning.

Amélioration de l'application

Cette tâche est référencée ici : subsection 2.2 (Chapitre 2).
Mon encadrant m'a demandé de faire quelques petites améliorations dans l'application. Par petite, j'entends des modifications qui n'entrainent pas de modification majeure de la structure de l'application.
Comme pour la résolution de bug, cette liste est susceptible d'évoluer dans le temps.

Pour l'instant, la liste des modifications est la suivante :
- Envoyer un email au comité lorsqu'un utilisateur s'inscrit
- Changer le texte de l'email qui est envoyé lors d'une demande de mot de passe oublié
- Empêcher les utilisateurs de soumettre des fichiers autres que des ZIP.
- Changer les noms d'utilisateur suivant cette nouvelle convention : première lettre de chaque mot en majuscule et le reste en minuscule
- Rendre l'affiliation obligatoire lors de l'inscription
- Charger dynamiquement l'extension de l'instance lors du téléchargement (à l'heure actuelle, l'extension est toujours xml)

Cette tâche est actuelle à 2 semaines, cependant cette durée peut-être ajustée en fonction des besoins de mon encadrant.

Datasets privé

Cette tâche est ma tâche principale (subsection 2.3 (Chapitre 2)), c'est la tâche qui devrait me prendre le plus de temps.
J'ai fait une liste de ce que j'ai prévu de modifier pour mon encadrant afin de ne rien avoir oublié.

Je vais donc :
- Ajouter les champs nécessaires dans la table **Dataset**
- Faire une migration pour les anciens datasets
- Modification de la liste des datasets pour afficher uniquement ceux publics
- Ajout d'un champ sur le formulaire de l'upload pour choisir la visibilité (ainsi que le traitement métier derrière)
- Ajout d'un bouton pour changer la visibilité d'un dataset dans la liste des dataset de l'auteur (passage de privé à public)
- Ajout d'un cadre pour afficher le token et le lien pour télécharger le dataset
- Ajout d'un bouton pour réinitialiser le token
- Modification de la page d'affichage des datasets pour accepter les datasets privé (avec le bon token)

Cette tâche est estimée à 8 semaines. Durant ces 8 semaines, j'effectuerai aussi des tests afin de vérifier le bon comportement de l'application.

Changement d'URL des datasets

Cette tâche est référencée ici : subsection 2.4 (Chapitre 2) dans le cahier des charges.
Cette tâche sera effectuée en même temps que la tâche sur les datasets privés.

Les modifications prévues sont :
- Modifier la base de données pour accueillir cet identifiant unique

— Effectuer une migration afin de générer les identifiants pour les datasets déjà présents dans le système.

— Générer les identifiants pour chaque nouveau dataset

— Modifier le contrôleur et la vue pour charger le dataset en fonction de l'identifiant et non du slug

Multi-objectif

Cette tâche est référencée ici : subsection 2.5 (Chapitre 2).
En accord avec mon encadrant, cette tâche est la moins prioritaire. Je commencerai cette tâche uniquement si j'ai de l'avance sur le planning.

J'ai quand même listé les différentes modifications à faire pour celui qui reprendra le projet.

— Modification de la base de données (suivant le diagramme ici : Figure 1 (Chapitre 5))
— Migration pour affecter des valeurs par défaut
— Modification de l'upload : Si ce n'est pas le format VRP-REP, on ne fait rien, sinon il faut vérifier le fichier en fonction du nombre de critères
— Modification de l'upload pour prendre en compte la vérification des Solutions en fonction des critères
— Modification de la table BSK pour avoir tous les critères et pouvoirs trier en fonction de tel ou tel critère
— Générer les templates dynamiquement pour les dépôts de solution.

Planning prévisionnel

Voici le planning prévisionnel :

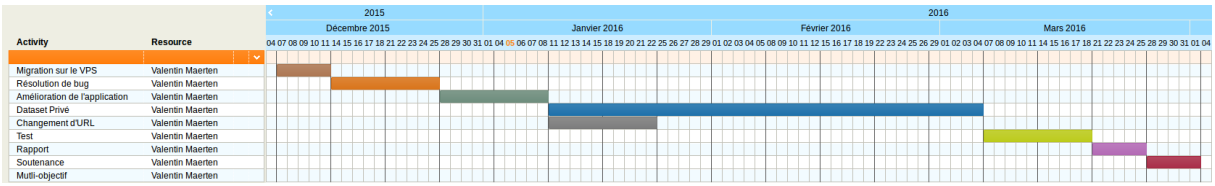


Figure 1 – *Planning prévisionnel*

La partie multi-objectif sera commencée si j'ai fini en avance la partie dataset privé.

2 Méthodologies et outils de suivi de projet

Concernant, la méthodologie nous avons choisi de partir sur une méthode agile. Cela permet une meilleure flexibilité de projet, et de coller au mieux à ce que désire l'encadrant.

Comme outils de gestion, nous utilisons Trello. Le Trello est déjà présent pour la gestion de ce projet avant que je ne commence ce PRD.
Nous effectuons des réunions soit chaque semaine soit une semaine sur deux.

Deuxième partie

Développement

8 Méthodologie

1 Gestion du gestionnaire de version Git

J'ai utilisé un gestionnaire de version qui est Git. Quand j'ai récupéré le projet, il y avait une branche **master** qui contenait la production, ainsi qu'une branche **develop** qui elle contenait la beta (où l'on peut tester en condition réelle). Ces branches sont les branches sur le serveur.
Je n'ai pas touché à ces branches.

Pour le développement, j'ai décidé de créer une branche par fonctionnalité. Ce qui m'a permis de travailler sur plusieurs fonctionnalités en même temps, par exemple dans l'attente de validation de mon encadrant.

2 Assurance qualité

J'ai mis en place pour le projet un outil permettant de mesurer la qualité du code qui est SonarQube. Il supporte plusieurs langages, dont **Java** & **PHP**.
SonarQube permet d'évaluer certaines composantes comme :

- Duplication de code
- Identification de bug (variable non initialisée,etc)
- Respect des règles de programmation
- Complexité de l'application (ou de chaque fonction)
- Mesure de la quantité de documentation par rapport au nombre de lignes de code

Les règles précédemment citées existent de base dans **SonarQube** mais elles peuvent être bien entendu définies par l'utilisateur.

Grâce à SonarQube, il est possible de repérer des erreurs que l'on aurait pu ne pas voir lors de la conception. De plus, cela permet aussi de vérifier que les conventions de code sont respectées, comme on peut ajouter des règles il suffi d'ajouter toutes les règles correspondants à nos conventions.

Voici une capture d'écran du dashboard de **SonarQube** de l'application VRP-REP :

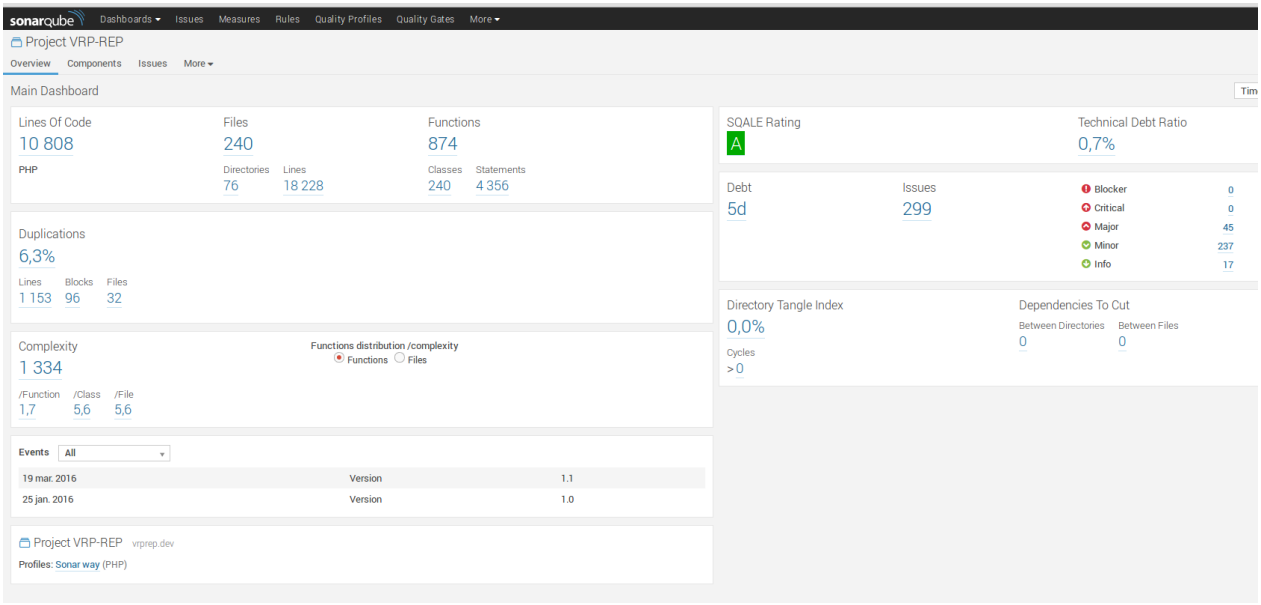


Figure 1 – *Dashboard SonarQube*

9 Mise en œuvre

Pour ce projet, j'ai développé plusieurs fonctionnalités demandées par mon encadrant.
Les fonctionnalités sont :

- Migration de l'application sur le VPS
- Envoyer un email aux administrateurs lorsqu'un utilisateur s'inscrit
- Empêcher les utilisateurs de soumettre des fichiers autres que des ZIP
- Changer les noms d'utilisateur suivant cette nouvelle convention : première lettre de chaque mot en majuscule et le reste en minuscule
- Rendre l'affiliation obligatoire lors de l'inscription
- Charger dynamiquement l'extension de l'instance lors du téléchargement (à l'heure actuelle, l'extension est toujours xml)
- Changement d'URL des datasets (ajout d'un ID unique compréhensible pour chaque dataset)
- Ajout des datasets privés

J'ai choisi ici de détailler le développement de la fonctionnalité permettant d'avoir des datasets privés ainsi que la possibilité de mettre une newsletter en place, car ces fonctionnalités sont à mon sens intéressantes.

Il est important de noter que chaque texte écrit dans l'application n'est pas écrit "en dur" mais dans un fichier de traduction ce qui permet de regrouper les textes de l'application.

1 Développement des datasets privés

L'objectif de cette fonctionnalité est d'offrir à l'utilisateur la possibilité d'avoir des datasets privés (donc pas visible par les autres) mais aussi de pouvoir les partager avec d'autres utilisateurs grâce à un token.

1.1 Mise en œuvre de la base de données

Pour gérer les datasets privés, je dois ajouter deux attributs dans la table **dataset** qui sont :

- un champ **private** qui sera un booléen (en réalité un tinyint(1)) qui décrira si le dataset est privé ou non
- un champ **token** qui sera un varchar(255) pouvant être null. Ce champ stockera le token pour accéder au dataset s'il est privé, sinon ce champ vaudra null.

Grâce à Doctrine, il est possible de faire des migrations. De plus, les migrations seront exécutées automatiquement lors du déploiement sur le serveur. Les migrations ont une fonction **up** et une fonction **down** ce qui permet de faire des reverts si nécessaire.
Au niveau du code, Doctrine donne accès à un environnement qui permet exécuter du SQL facilement.

```
1  public function up(Schema $schema)
2  {
3      $this->abortIf($this->connection->getDatabasePlatform()->getName() != 'mysql',
           'Migration can only be executed safely on \'mysql\'.');
4      $this->addSql('ALTER TABLE dataset ADD private TINYINT(1) DEFAULT \'0\' NOT NULL, ADD
           token VARCHAR(255) DEFAULT NULL');
5  }
6
7  public function down(Schema $schema)
8  {
9      $this->abortIf($this->connection->getDatabasePlatform()->getName() != 'mysql',
           'Migration can only be executed safely on \'mysql\'.');
10     $this->addSql('ALTER TABLE dataset DROP private, DROP token');
11 }
```

De base, tout les datasets existants sont publics et les tokens ont la valeur null.

1.2 Modification du formulaire d'upload de dataset

Maintenant que la base de données a été modifiée, il faut donner la possibilité aux utilisateurs lors de l'upload de mettre leurs datasets privés. La règle est la suivante : Tout dataset privé peut-être rendu public mais en aucun cas un dataset public peut devenir privé.

Le but est d'ajouter une partie dans le formulaire d'upload juste après le format du dataset. Le résultat est le suivant :

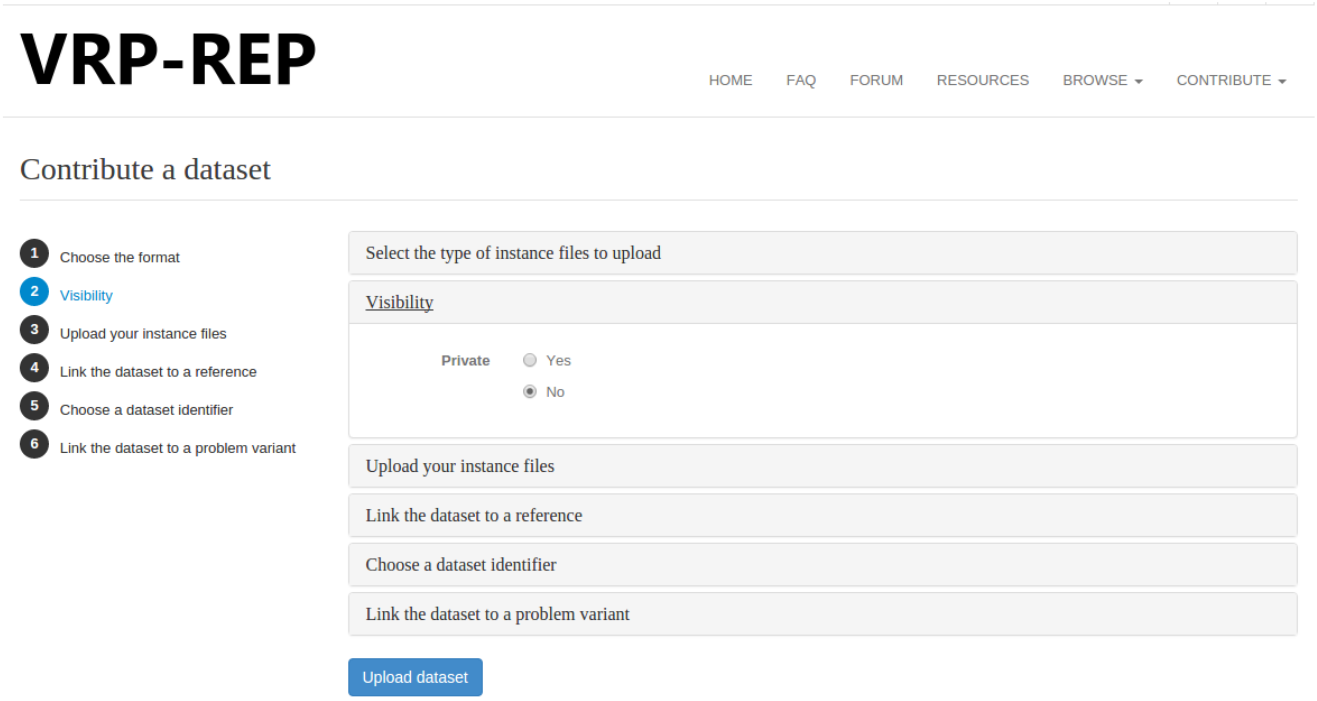


Figure 1 – *Upload d'un dataset avec la possibilité de choisir la visibilité*

Pour implémenter ça, il faut modifier la vue (le template Twig) et le contrôleur.
Je me suis inspiré du code existant pour re-créer une nouvelle box avec la visibilité. Les panels utilisent **Bootstrap** comme cet exemple le montre.

Maintenant que le côté front est fait, il faut modifier le contrôleur pour prendre en compte la visibilité. Le contrôleur est le **DatasetController** qui possède une méthode **addAction**.
Mais ce n'est pas cette méthode qu'il faut modifier enfin pas directement, cette méthode fait appel à la classe **DatasetFormHandler** qui va traiter le formulaire afin de renseigner les données dans un objet **Dataset**.
Symfony est capable de mapper directement certains inputs du formulaire aux attributs du modèle correspondant. C'est le cas pour l'attribut **private** donc nous n'aurons pas à nous en soucier.

Cependant, si le dataset est privé il faut générer le token. Pour cela, j'utilise la fonction openssl_random_pseudo_bytes qui permet de générer une chaine pseudo-aléatoire d'octets pour une certaine longueur. Cette fonction retourne des octets donc j'utilise la fonction bin2hex qui retourner est une chaîne ASCII.

La classe **DatasetFormHandler** est aussi utilisée par la fonctionnalité d'édition des datasets, il faut alors bien faire attention de générer le token uniquement lors de l'ajout d'un dataset, car sinon un token sera re-généré à chaque fois.

De plus, il faut qu'un utilisateur lambda ne puisse pas voir les références et variantes ne contenant que des datasets privés lors de l'upload par contre l'auteur d'une référence ou d'une variante dite privée peut sélectionner ces références / variantes.
Il faut donc avoir accès à l'utilisateur connecté. On est sûr que l'utilisateur existe, car il n'est pas possible d'accéder à l'upload sans un utilisateur valide.
Pour retrouver l'utilisateur connecté, Symfony nous donne accès à un service qui permet facilement de le retrouver.
La subtilité est que ce service n'est pas disponible dans la classe qui fait les requêtes, mais uniquement dans le contrôleur. Donc il faut récupérer l'utilisateur dans le contrôleur et le passer en paramètre la méthode qui fera la requête SQL.

Ce code permet de récupérer l'utilisateur connecté.

```
1 $user = $this->container->get('security.context')->getToken()->getUser();
```

Le raisonnement est le même pour les variantes et références : il faut sélectionner les variantes / références pour lesquelles il n'existe pas de datasets public ou que l'utilisateur connecté est l'auteur de la variante / référence.

1.3 Possibilité de passer les datasets en public

Une fois que l'utilisateur a uploadé son dataset en privé, il faut lui offrir la possibilité de passer ses datasets en public (l'inverse n'étant pas possible). Actuellement, l'interface permettant de consulter ses datasets est la suivante :

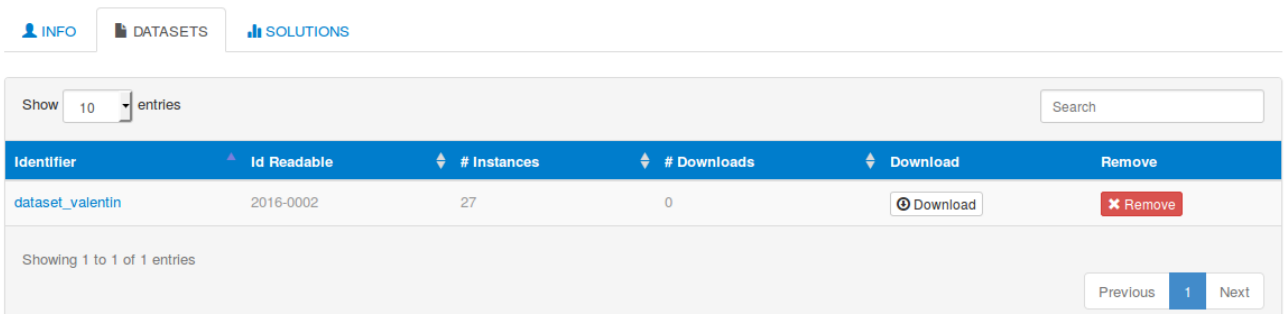


Figure 2 – *Liste des datasets à partir du profil utilisateur*

Le but ici est d'avoir une colonne en plus afin d'indiquer si le dataset est public ou privé, ainsi qu'une autre colonne avec bouton pour passer le dataset en public.

Cette page est générée avec une datatable de Jquery. A l'arrivé sur la page, un appel AJAX est effectué qui appelle une méthode qui fait une requête SQL sur la base afin de retourner toutes les informations sur les datasets.
J'ai donc du modifier cette requête afin de prendre en compte l'ajout d'information (privé ou non).

De plus, j'ai du indiquer à la datatable qu'il fallait afficher la visibilité (privé ou public) ainsi que le bouton si nécessaire. Â ce moment, j'ai une interface, mais le bouton n'est lié à aucune méthode back-end.
Grâce à Jquery, il est possible de générer des routes vers un contrôleur facilement. Si le dataset est privé, on affiche le bouton avec l'url vers le **DatasetController**.

Lorsque l'utilisateur clique sur le bouton, il lui est demandé de valider sachant qu'aucun retour en arrière ne sera possible. Après la validation, j'appelle une méthode dans ce contrôleur, je récupère les informations que m'a retournées la méthode et j'affiche un message de succès ou d'échec.

Côté backend, j'ai créé une méthode **renderPublicAction(Dataset $dataset)** qui vérifie que l'utilisateur a le droit de rendre ce dataset privé (qu'il est bien l'auteur de ce dataset) puis je passe le dataset en public.
Le code de la méthode est le suivant :

```php
public function renderPublicAction(Dataset $dataset)
{
    $response = function ($message) use ($dataset) {
        return new JsonResponse(
            array(
                'message' => $this->container->get('translator')->trans(
                    $message,
                    array('%dataset%' => $dataset)
                )
            )
        );
    };
    if (!$this->container->get('security.context')->isGranted('EDIT', $dataset)) {
        return $response('site.repository.datasets.edit.error.not-allowed');
    }
    $em = $this->container->get('doctrine.orm.entity\_manager');
    $dataset->setPrivate(false);
    $dataset->setToken(null);
    $em->persist($dataset);
    $em->flush();
    return $response('site.repository.datasets.edit.success');
}
```

Le résultat de cette fonctionnalité est le suivant :

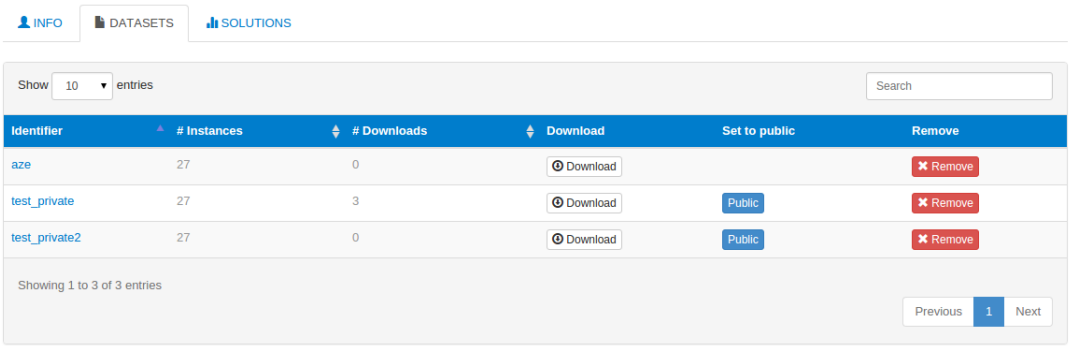


Figure 3 – *Liste des datasets*

1.4 Modification de la liste des datasets

Une des fonctionnalités demandées est que les datasets privés ne soient pas visibles par les utilisateurs lambda (utilisateurs inscrits ou non). J'ai donc modifié la page qui affiche les datasets.
La modification est essentiellement concentrée sur la requête qui est faite à la base de données, en effet je dois ajouter des conditions supplémentaires. Je dois afficher les datasets publics mais également les datasets privés si l'auteur de ces datasets est la personne connectée.

Nous avons vu comment récupérer les utilisateurs grâce à Symfony. La subtilité est que ce service n'est pas disponible dans la classe qui fait les requête mais uniquement dans le contrôleur. Donc il faut récupérer l'utilisateur dans le contrôleur et le passer en paramètre la méthode qui fera la requête SQL.

Concernant la requête, c'est simplement un filtrage sur le champ *private* en prenant en compte le cas de l'auteur du dataset.

1.5 Modification de la liste des références & variantes

Lorsque l'utilisateur upload un dataset privé, il peut choisir de créer une variante ou une référence. La variante ou la référence possède alors uniquement ce dataset qui est privé. L'entité (variante ou référence) ne doit donc pas apparaitre dans la liste correspondante.

Il faut plus ou moins faire la même chose que précédemment, c'est-à-dire, filtrer sur les entités ne contenant aucun dataset privé. Cependant, si l'utilisateur connecté est l'auteur de l'entité, il peut la voir et la consulter.

Cependant, il faut aussi penser à la sécurité ! En effet, il n'est pas possible de trouver l'entité dans la liste correspondante mais que se passe t-il s'il un utilisateur arrive à trouver l'url ? Il pourras la consulter.
J'ai donc corrigé ça, l'entité n'est consultable uniquement par son auteur si elle contient uniquement des datasets privés. Pour cela, il faut faire des vérifications : si l'entité contient au moins un dataset public, tout le monde peut y accéder sinon uniquement l'auteur peut y accéder (on peut récupérer l'utilisateur grâce à Symfony).

Lorsque l'utilisateur essaye de consulter une entité et qu'il n'a pas la permission de le faire, j'affiche une page 404 qui ressemble à cela :

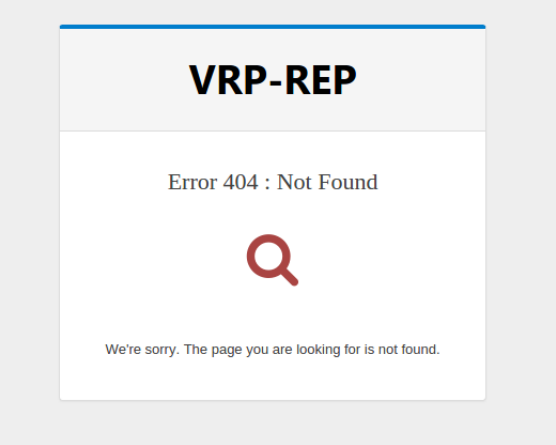


Figure 4 – *Page 404 Not Found*

1.6 Modification du formulaire d'upload de solution

Concernant l'upload de solution, la règle est un peu plus simple : même l'auteur ne peut pas déposer une solution pour un dataset privé. C'est la clé de voute de cette fonctionnalité.
Par contre, concernant les variantes / références, il n'est possible de les sélectionner que si elles possèdent au moins un dataset public. L'auteur pourra voir toutes les variantes / références mais pas forcement les sélectionner (voir la règle précédente). Un utilisateur lambda ne pourra pas voir les variantes / références qui n'ont que des datasets privés.

Pour cela, j'ai modifié la requête SQL qui remplit les select HTML. On est sur que l'utilisateur est connecté car c'est un pré-requis pour déposer un dataset.
Grâce à Doctrine, il est possible d'écrire beaucoup moins de SQL et de faire générer la requête en question, voici un exemple sur la requête modifiée :

```
$queryBuilderReference = function (EntityRepository $repository) use ($user) {
    $queryBuilder = $repository->createQueryBuilder('a');
    return $queryBuilder->where(
             $queryBuilder->expr()->not(
                    $queryBuilder->expr()->exists(
                        $this->em->createQueryBuilder()
                          ->select('1')
                            ->from('SiteRepositoryBundle:Dataset', 'd')
                            ->where('d.reference= a.id')
                            ->andWhere('d.private = 1')
                        )
                    )
                )
        ->orWhere('a.createdBy = :idUser')
        ->setParameter('idUser', $user->getId())
        ->orderBy('a.identifier', 'asc');
```

1.7 Modification de la page d'affichage et téléchargements des datasets

Les datasets privés ne sont pas visibles dans la liste des datasets (ou alors uniquement par leurs auteurs) mais si un utilisateur (connecté ou non) tombe sur l'url, il pourra le consulter or ce n'est pas le comportement souhaité. Mais alors l'auteur va aussi vouloir partager son dataset, c'est la qu'intervient le token stocké précédemment.
L'url pour un dataset privé sera de la forme : HOST/dataset/item/un-dataset.html ?token=xxxxxxxxx.

J'ai donc modifié les conditions de rendu du template. Je vérifie si le dataset est privé, si c'est le cas l'utilisateur peut consulter la page si l'un de ces deux conditions est rempli :

- l'utilisateur connecté (s'il existe) est l'auteur du dataset
- le token renseigné dans l'url (avec un passage en GET de la forme token=xxxxx) correspond au dataset demandé

Voici un extrait du contrôleur qui gère l'affichage d'un dataset.

```
if($dataset->isPrivate()) {
    if (false === $this->container->get('security.context')->isGranted('EDIT',$dataset) ↩
            && (null === $request->get('token') || $request->get('token') != $↩
            dataset->getToken()))
        return $this->render(
            '@Twig/Exception/error404.html.twig',
            array(
                'status_code' => 404,
                'status_text' => $this->container->get('translator')
                      ->trans('site.repository.datasets.item.not_found')
            )
```

```
        );
}
```

Si l'utilisateur n'a pas l'autorisation de le voir, je fais comme pour les variantes / références j'affiche une page 404 Not Found.

Mais la page classique d'un dataset permet de consulter aussi les variantes / références associées grâce un lien sauf qu'ici nous donnerions accès à des informations potentiellement privées. Pour résoudre ce problème, dans la vue si l'utilisateur consulte la page depuis un token, il n'a pas accès aux liens. Niveau code, j'ai injecté le token (même s'il est null) dans la vue depuis le contrôleur ce qui me permettra de faire des tests dessus. Les datasets publics ne seront bien évidemment pas affectés par cette modification.

Dernier point, il faut aussi pouvoir télécharger le dataset lorsqu'on est en possession du token. Pour cela, j'ai modifié la méthode du contrôleur qui gère le téléchargement afin d'effectuer la même vérification que pour la consultation. J'ai aussi modifié le lien qui est présent sur la page de consultation pour y intégrer le token et donc pouvoir télécharger le dataset à partir de cette page.

1.7.1 Ajout d'un bouton "Copier vers le press papier"

Pour faciliter l'expérience utilisateur, mon encadrant m'a demandé d'ajouter un bouton "Copy to clipboard" disponible pour les datasets privés. Ce bouton sera disponible sur :

- La page d'un dataset privé
- La liste des datasets de l'utilisateur depuis son espace personnel

Pour ce faire, afin d'avoir la meilleure compatibilité j'ai utilisé une bibliothèque. J'ai utilisé la bibliothèque Clipboard.js.

Au lieu de télécharger la bibliothèque directement, je vais passer par un gestionnaire de dépendance frontend ici bower. J'ai donc ajouté dans le fichier **bower.json** une ligne pour ajouter la bibliothèque. Après j'ai installé la bibliothèque grâce à cette commande :

```
bower install
```

Au niveau du code, il y a un bouton placé dans le code HTML comme ceci :

```
<button id="copyButton" class="btn" data-clipboard-text="{{ ↩
        url('site_repository_dataset_itembyidreadable', {idReadable: dataset.idReadable, ↩
        token:dataset.token}) }}">
    Copy to clipboard
</button>
```

Je génère la route grâce à la fonction **url** de Twig. Il faut avoir un id sur le bouton afin de déclencher la copie.
Maintenant, il faut ajouter une partie javascript afin d'instancier la bibliothèque.

```
var clipboard = new Clipboard('#copyButton');
```

Maintenant, lorsque l'utilisateur clique sur le bouton le lien permettant de partager le dataset privé est copié dans son presse-papier. Le résultat en terme d'interface est le suivant :

Pour la liste des datasets dans l'espace utilisateur :

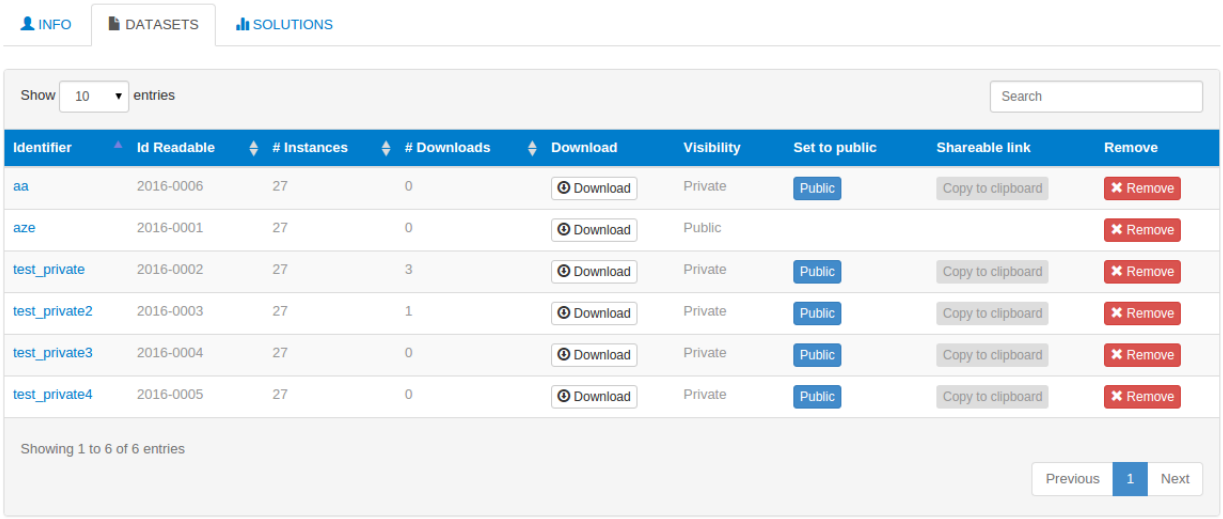


Figure 5 – *Visualisation des datasets d'un utilisateur avec le bouton Copy To Clipboard*

Pour la page de consultation d'un dataset :

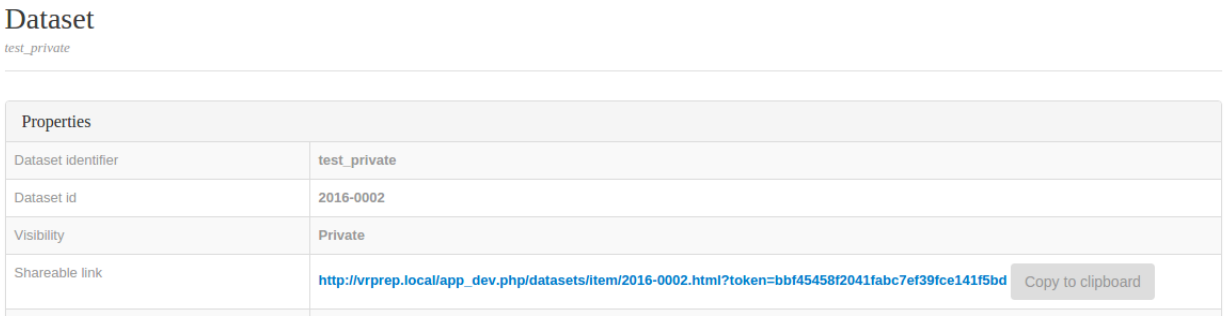


Figure 6 – *Visualisation d'un dataset privé avec le bouton Copy To Clipboard*

2 Mise en place d'une newsletter

L'objectif de cette fonctionnalité est de créer une newsletter. Chaque utilisateur est libre d'y souscrire ou non. Cette newsletter sera en grande partie, pour l'instant, utilisée pour prévenir de l'ajout de nouveaux datasets ou solutions.
Mon encadrant et moi avons décidé de ne pas séparer les différentes parties de la newsletter : l'utilisateur souscrit à toute la newsletter ou n'y souscrit pas du tout.
Cette fonctionnalité est légèrement moins complexe que la précédente.

Mise en œuvre de la base de données

Concernant la base de données, il faudra ajouter un attribut qui représentera la souscription ou non de l'utilisateur à la newsletter. J'appellerais cet attribut **subscription_newsletter** et il sera un tinyint(1).
Comme pour le développement des datasets privé, Doctrine permet de faire ce changement assez rapidement. Il faut bien noter que de base tous les utilisateurs auront souscrit à la newsletter.

J'ai donc créé une commande Symfony qui va permettre d'enlever tous les utilisateurs de la liste de souscription. Cela sera utile par exemple pour tester la fonctionnalité sur la beta. Le code de cette commande est le suivante :

```php
protected function execute(InputInterface $input, OutputInterface $output)
{
    $em = $this->getContainer()->get('doctrine.orm.entity_manager');

    $users = $em->getRepository('ApplicationSonataUserBundle:User')->findAll();

    foreach ($users as $user) {
        $user->setSubscriptionNewsletter(false);
        $em->persist($user);
    }
    $em->flush();
}
```

Cette commande Symfony peut être lancée depuis le shell grâce à :

```
php app/console repository:newsletter:remove
```

Modification du formulaire d'inscription d'un utilisateur

Il fallait également modifier le formulaire d'inscription d'un utilisateur afin d'ajouter la possibilité de souscrire à la newsletter.
Avec Symfony, pour les formulaires il faut créer des classes qui seront chargées de créer un formulaire. Il est donc nécessaire de modifier cette classe pour y ajouter le champ.
La classe en question est **RegistrationFormType**.
Il y a une fonction qui est **buildForm(FormBuilderInterface $builder, array $options)**, le builder va permettre de créer les différents champs.

J'ai ajouté au builder déjà existant mon champ comme ceci :

```php
$builder->add(
        'subscriptionNewsletter',
        'choice', array(
            'choices' => array(1 => 'Yes', 0 => 'No'),
            'expanded' => true,
            'multiple' => false,
            'label' => 'Newsletter',
            'data' => 1
      )
    )
```

Dans ce code, je déclare que mon champ s'appellera "subscriptionNewsletter", il possèdera 2 choix : 0 ou 1. Je défini un label "Newsletter" qui me permettra de choisir la traduction du label. Le code php est modifié, maintenant il faut modifier le côté frontend.

Pour ce faire, j'ai modifié le fichier **register.html.twig** qui est fichier étendu du fichier de base de **sonata**. Ce fichier contient entre autre le formulaire HTML de l'inscription. J'ai donc ajouté le nouveau champ avec la syntaxe Twig.
La syntaxe pour les formulaires avec **Twig** est un peu différentes qu'avec le HTML. La syntaxe est la suivante :

```
{{ form_row(form.subscriptionNewsletter)}}
```

L'instruction **form_row** permet de créer un champ dans le **form** HTML.
La dernière étape est d'ajouter le texte du label dans le fichier correspondant.

L'avantage de Symfony surtout avec Sonata est qu'il gère automatiquement la création de l'entité à partir du formulaire. Il n'y a donc pas de modification à faire de ce côté.

Modification de la page du profil d'un utilisateur

Il faut aussi que l'utilisateur puisse modifier son choix dans son profil. La technique qui était mise en place sur les champs déjà présent est la suivante. Chaque champ est sous la forme d'un lien (balise a) qui est éditable grâce au plugin x-editable. Puis avec le javascript et grâce au plugin, la modification est faite directement en base de données avec un appel sur une méthode du contrôleur.

De mon côté, je ne pouvais pas faire tout à fait comme ça, car je n'ai pas de champ texte mais un booléen. J'ai donc décidé de mettre un bouton qui au clique sur celui-ci changera la valeur pour l'utilisateur.
Voilà le code du bouton :

```
{% if user.subscriptionNewsletter %} {% set classCss = "btn btn-primary" %} {% else %} {% set classCss = "btn btn-primary" %}  {% endif %}

<span>
    <button id="buttonNewsletter" class="{{ classCss }}"
            data-name="subscriptionNewsletter" data-value="{{user.subscriptionNewsletter ? 0 : 1}}" data-pk="{{ user.id }}">
        <span id="spanNewsletter">
            {# if the user has suscribed to the newsletter, we display 'no' #}
            {% if user.subscriptionNewsletter %}
                No
            {% else %}
                Yes
            {% endif %}
        </span>
    </button>
</span>
```

J'utilise les data-values pour stocker le nom, la valeur (0 ou 1) ainsi que la clé primaire de l'utilisateur.
J'ai également ajouté du javascript afin de pouvoir faire une action lors d'un clique sur le bouton. Pour cela, je fais un appel AJAX sur une méthode du contrôleur puis je met à jour l'interface.
Voici la fonction javascript correspondante :

```
//When we click on the button to change the subscription of the newsletter
$("#buttonNewsletter").click(function () {
    var id = $(this).data('pk');
    var value = $(this).data("value");
    var name = $(this).data("name");
    $.ajax({
        url: Routing.generate('edit_user', {id: id}),
        data: {"pk": id, "value": value, "name": name},
        type: 'POST',
        success: function (data) {
            //Change the displaying text
            var valueText = !value ? 'Yes' : 'No';
            //Change the value
            $('#buttonNewsletter').data("value", +(!value));
            $('#spanNewsletter').html(valueText);
            $.gritter.add({
                text: data,
                sticky: false,
                time: 2000
            });
        }
    });
});
```

La fonction **edit_user** modifie l'utilisateur en fonction des paramètres.

10 Tests

Durant le développement, j'ai effectué des tests afin de valider le bon fonctionnement des fonctionnalités apportées à l'application.
Sur le projet que j'ai repris, il n'y avait pas de système d'automatisation de tests fonctionnels, par manque de temps je n'ai pas pu mettre en place un tel système.

J'ai cependant effectué ces tests à la main et je détaillerai dans ce chapitre les différents tests effectués.
Avec ces tests, j'ai mis l'accent sur la sécurité. L'application étant déjà en production, il faut absolument qu'une personne non autorisée ne puisse pas voir un dataset privé. Il en est de même pour les variantes / références car cela pourrait donner des informations sur les recherches d'une certaine personne alors qu'elle ne le souhaite pas.

Voici le tableau excel résumant les différents tests :

Numéro Test	Type Test	Fonctionnalités principalement testées	Nom (et description) du scénario	Données	Vérifications / résultats attendus
T01	Fonctionnel	Dataset privés	Uploader un dataset (le format, les données, les variantes et la référence n'ont aucun impact) en choissant de le rendre public	Un dataset quelconque, par exemple celui la : (Augerat 1995 - Set A)	- L'upload se passe bien - Il est possible de consulter le dataset grâce à sa fiche - Le dataset est présent dans la liste des datasets
T02	Fonctionnel	Dataset privés	Uploader un dataset (le format, les données, les variantes et la référence n'ont aucun impact) en choissant de le rendre privé	Un dataset quelconque, par exemple celui la : (Augerat 1995 - Set A)	- L'upload se passe bien - Il n'est possible de consulter le dataset grâce à sa fiche sans le token (erreur 404) - Le dataset n'est pas présent dans la liste des datasets pour un utilisateur quelconque, ou un utilisateur anonyme - Le dataset est présent dans la liste des datasets pour un l'auteur du dataset
T03	Fonctionnel	Dataset privés	Consulter un dataset privé sans le token correspondant	Un dataset privé quelconque. Attention, il ne faut pas être connecté avec le compte de l'auteur. Il possible de réaliser le test avec un utilisateur anonyme ou un utilisateur inscrit.	Il n'est possible de consulter le dataset sans le token (erreur 404)
T04	Fonctionnel	Dataset privés	Consulter un dataset privé avec le token correspondant	Un dataset privé quelconque avec son token. Attention, il ne faut pas être connecté avec le compte de l'auteur. Il possible de réaliser le test avec un utilisateur anonyme ou un utilisateur inscrit. Le token est passé en GET dans l'url : datasets/item/test-private.html?token=xxxxxxxxxxx	- Affichage de la page du dataset ainsi qu'un lien permettant de le télécharger - Les liens sur la page ne sont pas cliquable
T05	Fonctionnel	Dataset privés	Consulter un dataset privé en étant l'auteur	Un dataset quelqconque. Il faut être connecté avec le compte de l'auteur du dataset.	- Affichage de la page du dataset ainsi qu'un lien permettant de le télécharger - Les différents liens sur la page sont disponibles
T06	Fonctionnel	Dataset privés	Télécharger un dataset privé avec le token	Un dataset privé quelconque avec son token. Attention, il ne faut pas être connecté avec le compte de l'auteur. Il possible de réaliser le test avec un utilisateur anonyme ou un utilisateur inscrit. Le token est passé en GET dans l'url : datasets/download/test-private-valentin.zip?token=xxxxxxxxxxx	Téléchargement du dataset en question
T07	Fonctionnel	Dataset privés	Télécharger un dataset privé sans le token	Un dataset privé quelconque. Attention, il ne faut pas être connecté avec le compte de l'auteur. Il possible de réaliser le test avec un utilisateur anonyme ou un utilisateur inscrit.	Il n'est possible de télécharger le dataset sans le token (erreur 404)
T08	Fonctionnel	Dataset privés	Consulter la liste des datasets avec un utilisateur lambda	Être connecté avec un utilisateur ne possèdant pas de dataset privé Avoir des datasets privé dans l'application	L'utilisateur ne voit que les datasets public
T09	Fonctionnel	Dataset privés	Consulter la liste des datasets avec un utilisateur qui possède un dataset privé	Être connecté avec un utilisateur possèdant au moins un dataset privé	L'utilisateur voit les datasets public ainsi que les datasets privés lui appartenant. Il ne voit cependant pas les datasets privés des autres.
T10	Fonctionnel	Dataset privés	Consulter la liste des variantes avec un utilisateur lambda	- Être connecté avec un utilisateur ne possèdant pas de dataset privé - Avoir une variante ne contenant que des datasets privé dans l'application - Avoir une variante ne contenant que des datasets public dans l'application - Avoir une variante contenant à la fois des datasets privés et des datasets public	L'utilisateur peut voir : - Les variantes ne contenant que des datasets public - Les variantes contenant à la fois des public et des privés Par contre, il ne peut pas voir les variantes ne contenant que des datasets privés
T11	Fonctionnel	Dataset privés	Consulter la liste des variantes avec un utilisateur qui possède une variante dans laquel il n'y a que des dataset privé	- Être connecté avec un utilisateur possèdant une variante dans laquelle il n'y a QUE ses dataset privé - Avoir une variante ne contenant que des datasets public dans l'application - Avoir une variante contenant à la fois des datasets privés et des datasets public	L'utilisateur peut voir : - Les variantes ne contenant que des datasets public - Les variantes contenant à la fois des public et des privés - Les variantes qui ne contienent que des datasets privés lui appartenant Ne peut pas voir les variantes contenant uniquement des datasets privés ne lui appartenant pas
T12	Fonctionnel	Dataset privés	Consulter la liste des référence avec un utilisateur lambda	- Être connecté avec un utilisateur ne possèdant pas de dataset privé - Avoir une référence ne contenant que des datasets privé dans l'application - Avoir une référence ne contenant que des datasets public dans l'application - Avoir une référence contenant à la fois des datasets privés et des datasets public	L'utilisateur peut voir : - Les références ne contenant que des datasets public - Les références contenant à la fois des public et des privés Par contre, il ne peut pas voir les références ne contenant que des datasets privés
T13	Fonctionnel	Dataset privés	Consulter la liste des référence avec un utilisateur qui possède une référence dans laquel il n'y a que des dataset privé	- Être connecté avec un utilisateur possèdant une variante dans laquelle il n'y a QUE ses dataset privé - Avoir une variante ne contenant que des datasets public dans l'application - Avoir une variante contenant à la fois des datasets privés et des datasets public	L'utilisateur peut voir : - Les références ne contenant que des datasets public - Les références contenant à la fois des public et des privés - Les références qui ne contienent que des datasets privés lui appartenant Ne peut pas voir les références contenant uniquement des datasets privés ne lui appartenant pas
T14	Fonctionnel	Validation côté serveur	Inscription d'un utilisateur avec des informations correctes	- email : "ceciestuntest@gmail.com" - password : "test123" - First Name : "Jean-paul" - Family Name : "Pignon" - Country : "France" - Affiliation : "Polytech"	L'utilisateur est bien inscrit. Il recoit un email pour confirmer son compte
T15	Fonctionnel	Validation côté serveur	Inscription d'un utilisateur avec des informations incorrectes	Il faut désactiver la validation HTML5 ou utiliser safari afin de pouvoir voir la validation côté serveur Données : - email : "" - password : "" - First Name : "" - Family Name : "" - Country : "France" - Affiliation : ""	L'utilisateur n'est pas inscrit. Les erreurs s'affiche en dessous de chaque champ : - Email : "Please enter an email" - Password : "Please enter a password" - First Name : "Please enter a firstname" - Last Name : "Please enter a lastname" - Affiliation : "Please enter an affiliation"
T16	Fonctionnel	Chargement dynamique de l'extension des instances	Télécharger un dataset avec des instances au format VRP-REP (XML)	Un dataset au format VRP-REP (Augerat 1995 - Set A)	Téléchargement du dataset avec les instances au format XML
T17	Fonctionnel	Chargement dynamique de l'extension des instances	Télécharger un dataset avec des instances avec un format quelconque	Un dataset avec un format quelconque (ConVRP_small) : extension .vrp	Téléchargement du dataset avec les instances au format .vrp
T18	Fonctionnel	Migration sur un VPS	Vérification d'accès à l'application	Le test doit être effectué sur le VPS	L'application est disponible (page d'accueil au moins)
T19	Fonctionnel	Migration sur un VPS	Création d'un compte	Le test doit être effectué sur le VPS - email : "ceciestuntest@gmail.com" - password : "test123" - First Name : "Jean-paul" - Family Name : "Pignon" - Country : "France" - Affiliation : "Polytech"	L'utilisateur est bien inscrit. Il recoit un email pour confirmer son compte
T20	Fonctionnel	Migration sur un VPS	Téléchargement d'un dataset	Le test doit être effectué sur le VPS Un dataset quelconque par exemple (Augerat 1995 - Set A)	Téléchargement du dataset
T21	Fonctionnel	Migration sur un VPS	Déposer un dataset	Un dataset quelconque, par exemple celui la : (Augerat 1995 - Set A)	- L'upload se passe bien - Il est possible de consulter le dataset grâce à sa fiche - Le dataset est présent dans la liste des datasets
T22	Fonctionnel	Migration sur un VPS	Désposer une solution	Un dataset quelconque, par exemple celui la : (Augerat 1995 - Set A) Une solution quelconque pour ce dataset	- L'upload se passe bien - La solution est présente dans la table BKS
T23	Fonctionnel	Modification du texte de l'email envoyé pour la réinitialisation du mot de passe	Demande d'un nouveau mot de passe	Un compte qui est existe (peut importe le compte)	L'email envoyé contient le bon texte
T24	Fonctionnel	Empêcher la soumission de fichier autre que ZIP lors de l'upload de dataset	Upload un dataset avec les instances dans un fichier ZIP	Un fichier ZIP contenant des instances ainsi qu'un README	- L'upload se passe bien - Il est possible de consulter le dataset grâce à sa fiche - Le dataset est présent dans la liste des datasets
T25	Fonctionnel	Empêcher la soumission de fichier autre que ZIP lors de l'upload de dataset	Upload un dataset avec des fichiers autre que ZIP	Les fichiers d'instances sans ZIP	L'upload renvoie une erreur
T26	Fonctionnel	Création d'un système de newsletter	Création d'un utilisateur souscrivant à la newsletter	- email : "ceciestuntest@gmail.com" - password : "test123" - First Name : "Jean-paul" - Family Name : "Pignon" - Country : "France" - Affiliation : "Polytech" - newsletter : No	L'utilisateur est bien inscrit à la newsletter (visible ici : /profile/)
T27	Fonctionnel	Création d'un système de newsletter	Création d'un utilisateur ne souscrivant pas à la newsletter	- email : "ceciestuntest@gmail.com" - password : "test123" - First Name : "Jean-paul" - Family Name : "Pignon" - Country : "France" - Affiliation : "Polytech" - Newsletter : No	L'utilisateur n'est plus inscrit à la newsletter (visible ici : /profile/)
T28	Fonctionnel	Création d'un système de newsletter	Inscription à la newsletter à partir d'un utilisateur qui ne l'est pas	être connecté avec un utilisateur qui n'est pas inscrit à la newsletter	L'utilisateur est bien inscrit à la newsletter (visible ici : /profile/)
T29	Fonctionnel	Création d'un système de newsletter	Désinscription à la newsletter à partir d'un utilisateur qui est inscrit	être connecté avec un utilisateur qui est inscrit à la newsletter	L'utilisateur n'est plus inscrit à la newsletter (visible ici : /profile/)
T30	Fonctionnel	Envoyer un mail lors qu'il y a une nouvelle solution ou un dataset	Déposer une nouvelle solution	Un dataset quelconque, par exemple celui la : (Augerat 1995 - Set A)	- Un mail est envoyé aux personnes inscrit à la newsletter - Le mail contient les informations necessaire
T31	Fonctionnel	Envoyer un mail lors qu'il y a une nouvelle solution ou un dataset	Déposer un nouveau dataset	Un dataset quelconque, par exemple celui la : (Augerat 1995 - Set A) Une solution quelconque pour ce dataset	- Un mail est envoyé aux personnes inscrit à la newsletter - Le mail contient les informations necessaire
T32	Fonctionnel	Envoyer un mail aux administrateur lorsqu'un nouvel utilisateur s'es	Inscription d'un nouvel utilisateur	- email : "ceciestuntest@gmail.com" - password : "test123" - First Name : "Jean-paul" - Family Name : "Pignon" - Country : "France" - Affiliation : "Polytech"	Un email est envoyé sur l'adresse "vrp.repository@gmail.com"

11 Guide de migration de l'application

Durant cette partie recherche, une de mes tâches a été de créer un guide de migration afin de faciliter la migration de l'hébergement mutualisé vers le VPS.
Ce guide pourra servir par la suite si une autre migration doit être effectuée.

Durant ce guide j'aborderai plusieurs aspects :

- Préparation du VPS
- Installation de l'environnement (Apache2, PHP, Mysql)
- Installation des dépendances (Composer, NodeJs, Grunt, Gulp, Git)
- Déploiement et configuration de l'application
- Mise en place des backups

1 Préparation du VPS

Pour plus de sécurité, on va créer un compte qui s'occupera de gérer l'application. Ici je l'ai appelé vrp

```
1 sudo adduser vrp
```

Puis nous allons créer deux paires de clés SSH (une pour root et une pour vrp) afin de faciliter la connexion en SSH.

```
1 ssh-keygen
```

Côté client, on peut ajouter notre clé publique sur le VPS.

```
1 ssh-copy-id -i ~/.ssh/id_rsa.pub titi@toto.host.org
```

2 Installation de l'environnement

L'application tournant sur l'environnement LAMP [1], j'ai donc gardé le même environnement sur le VPS.
On va commencer par mettre à jour le système afin de profiter des dernières versions des paquets.

1. Linux, Apache, MySQL, PHP

```
sudo apt-get update && sudo apt-get upgrade
```

Puis nous allons installer Apache2 :

```
sudo apt-get install apache2 apache2-doc
```

On continue avec MySQL :

```
sudo apt-get install mysql-server php5-mysql
```

Il existe un script de sécurisation de MySQL qui permet d'avoir MySQL un peu plus sécurité en retirant par exemple le remote login sur root

```
sudo mysql_secure_installation
```

Pour finir, on installe PHP5 :

```
sudo apt-get install php5-common libapache2-mod-php5 php5-cli
```

À ce moment-là il y a une installation propre de LAMP. Cependant pour fonctionner Symfony2 demande un certain nombre de pré requis.

L'extension PHP "intl" :

```
sudo apt-get install php5-intl
```

Il faut aussi ajouter la timezone dans php.ini

3 Installation des dépendances

Maintenant que nous avons une installation de LAMP, il faut installer les dépendances logicielles que demande Symfony2 Il faut installer Composer mais pour installer Composer il faut installer Curl :

```
sudo apt-get install curl
```

Puis Composer ([WWW3]), on l'installe en global ce qui nous permettra du l'utiliser avec juste "composer"

```
curl -sS https://getcomposer.org/installer | sudo php -- --install-dir=/usr/local/bin ↩
    --filename=composer
```

L'application utilise Bower et Grunt qui s'installent grâce à NPM [2] :

```
sudo apt-get install nodejs npm
```

Puis on installe Bower et Grunt globalement :

```
npm install -g bower
npm install -g grunt
```

Pour finir, on installe Git :

```
sudo apt-get install git
```

2. Node Package Manager

4 Déploiement et configuration de l'application

Dans un premier temps, il faut récupérer l'application et les backups de l'ancien serveur. On pourra par exemple faire :

```
scp loginAncienServeur@AncienServeur:CheminDeLapplication ↩
        loginNouveauServeur@NouveauServeur:CheminOuOnSouhaiteLapplication
```

On va créer la base de données (dans mon exemple je l'appelle vrp) :

```
mysql -u root -p
CREATE DATABASE vrp;
exit
```

Par la suite, il faut changer les paramètres dans le fichier app/config/parameter.yml. Il faut changer les accès à la base de données et l'authentification mail.

Si on essaye d'accéder à l'application comme ça, on va avoir des problèmes d'écriture sur les dossiers app/cache et app/log. Il faut donc supprimer les fichiers et lui donner les droits (http ://symfony.com/doc/current/book/installation.html)

```
rm -Rf app/cache && rm -Rf app/logs
mkdir app/cache
mkdir app/logs
HTTPDUSER=`ps axo user,comm | grep -E '[a]pache|[h]ttpd|[_]www|[w]ww-data|[n]ginx' | grep ↩
        -v root | head -1 | cut -d\  -f1`
sudo setfacl -R -m u:"$HTTPDUSER":rwX -m u:`whoami`:rwX var/cache var/logs
sudo setfacl -dR -m u:"$HTTPDUSER":rwX -m u:`whoami`:rwX var/cache var/logs
```

Maintenant, on va vouloir accéder à l'application avec une URL, donc on va créer des virtual hosts.

```
sudo nano /etc/apache2/sites-enabled/000-default.conf
```

Voici un exemple de mes 2 virtual hosts (à adapter donc)

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    ServerName vrp.valentin-maerten.fr
    DocumentRoot /home/vrp/repos/live/web
    <Directory /home/vrp/repos/live/web>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride All
        Require all granted
    </Directory>

</VirtualHost>

<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    ServerName beta-vrp.valentin-maerten.fr
    DocumentRoot /home/vrp/repos/live/web
    <Directory /home/vrp/repos/live/web>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride All
        Require all granted
    </Directory>

</VirtualHost>
```

Maintenant il faut restaurer un backup pour avoir la base de données et les uploads.

Pour finir, il faut configurer git, il faut changer le remote origin (de git) par /CheminDeLapplication/vrp-rep.git. Il faut le faire pour live et pour staging.

```
git remote remove origin
git remote add origin /CheminDeLapplication/vrp-rep.git
```

On peut ajouter une crontab qui va faire un backup par jour :

```
15 2 * * * /bin/bash /home/vrp/backup/src/backup.sh vrp-rep
```

12 Reproductibilité

1 Installer l'application sur un poste de développement

Dans cette section, je vais expliquer comment installer l'application sur un poste de développement. Pour cela, il faut cloner l'application depuis le serveur Git. À l'heure actuelle, l'adresse IP du serveur est : 51.254.132.44. Il faut donc exécuter cette commande :

```
git clone vrp@51.254.132.44:var/repos/vrp-rep.git
```

cela aura pour effet de cloner l'application dans le répertoire courant.

Pour la suite, il faut avoir **Composer** d'installé si ce n'est pas fait, vous pouvez vous référer au guide de migration 11. Il faut installer les dépendances grâce à **Composer**.
Pour cela :

```
cd vrp-rep
composer install
```

Une fois cette commande lancée, vous allez récupérer toutes les dépendances nécessaires au bon fonctionnement de l'application dans le dossier **vendor**. Il vous sera demandé un certain nombre d'information comme par exemple le nom de la base de données. Notez bien ce nom pour pouvoir créer la base de données par la suite.
Concernant les tokens Github, la valeur ne peut pas être null, mettez donc par exemple 'a'. Les paramètres sont stockés dans le fichier `app/config/parameters.yml`.

Maintenant, vous devez créer la base de données sur Mysql. Appelez la du nom même que précédemment. Par exemple :

```
mysql -u root -p
CREATE DATABASE test_vrprep;
exit
```

Maintenant, il faut restaurer la base de données depuis le serveur (ce qui nous permettra aussi d'avoir les fichiers associés). Pour cela, reportez-vous à la section sur les backups 2.

La base de données est opérationnelle, mais il y peut y avoir encore un petit problème de droit sur les dossiers de cache, se référer à la section 4 (Chapitre 11) .

Vous voudrez sûrement utiliser des virtualHosts pour l'application, la encore vous pouvez vous référer à la section 4 (Chapitre 11) .

Encore quelques petites configurations et l'application pourra être utilisée. Il faut modifier le fichier **app/config/config.yml** et remplacer la ligne

```
default_file_path:       %kernel.root_dir%/../web/uploads_%kernel.environment%
```

par celle-ci :

```
default_file_path:       %kernel.root_dir%/../web/uploads_prod
```

car ici on a uniquement le dossier **upload_prod**.
Il faut également modifier le fichier **app/config/config_dev.yml** et remplacer la ligne :

```
dbname:   "%database_name%_dev"
```

par celle-la :

```
dbname:   "%database_name%"
```

Pour finir, il faut lancer quelques commandes afin de vider le cache et installer les assets :

```
php app/console asset:install web
php app/console asset:dump
php app/console cache:clear --env=dev
```

Vous pouvez maintenant accéder à l'application grâce à l'url que vous avez définie dans le virtual host.

2 Faire un backup / une restauration

Pour commencer, il faut récupérer les scripts de backup / restauration :

```
scp -r vrp@51.254.132.44:backup/ .
```

Grâce á cela, nous allons récupérer un dossier avec l'architecture suivante :

- backup/ ..... Ce répertoire contient les scripts ainsi que les différents backup
  - apps/ ..... Contient les différentes applications (prod & beta)
    - vrp-rep ..... Contient les backups de la prod
      - data .....Contient les différents backup au format yyyy-mm-dd-hh-mm-ss
        - yyyy-mm-dd-hh-mm-ss
          - database.sql.gz ..... Fichier SQL permettant de restaurer la base de données
          - upload.tar.gz ..... Archive contenant tout les fichiers uploader sur le site
      - variables.sh .. fichier contenant la configuration du script (dossier où sont stockés les fichers, etc)
    - vrp-rep-staging ..... Contient les backups de la beta
      - data .....Contient les différents backup au format yyyy-mm-dd-hh-mm-ss
        - yyyy-mm-dd-hh-mm-ss ..... Fichier de route principal
          - database.sql.gz ..... Fichier SQL permettant de restaurer la base de données
          - upload.tar.gz ..... Archive contenant tout les fichiers uploader sur le site
      - variables.sh .. fichier contenant la configuration du script (dossier où sont stockés les fichers, etc)
  - README.md
  - src/ ..... Le dossier qui contient les sources du script
    - app/ ..... Contient les scripts de plus bas niveau qui sont utilisés par les autres scripts

```
    ├── create.sh
    └── get.sh
├── backup.sh ..... Script utilisé pour faire un backup
└── restore.sh ..... Script utilisé pour faire une restore
```

On a donc ici bien les différents scripts ainsi que les backups en question.
Il faut en premier paramétrer le fichier **variables.sh**. Je vous conseille de prendre les backups de la prod car ils sont faits tout les jours.

Mon fichier **variables.sh** ressemble à ça :

```
#!/bin/sh

# Locations
UPLOADS_FOLDER=/home/valentin/test/vrp-rep/web/uploads_prod
BACKUP_DATA=$APPS_DIR/vrp-rep/data

# Variables
BACKUP_DATABASE_NAME=database.sql.gz
BACKUP_UPLOADS_NAME=uploads.tar.gz

# Expiry in days
EXPIRY=1440

# Database information
DB_HOST=localhost
DB_USER=valentin
DB_PASSWORD=xxxxx
DB_NAME=test_vrprep
```

La restauration

Pour faire la restauration, il faut créer le dossier **uploads_prod** dans le dossier **web** oú seront stockés les fichiers uploadés. Puis il faut lancer le script grâce aux commandes :

```
cd vrp-rep
mkdir web/uploads_prod
cd ../backup/src
bash restore.sh vrp-rep
```

Le script prend en entrée le nom de l'application à partir de laquelle on souhaite faire la restauration.

Le backup

Concernant le backup, la commande marche de la même manière que le script de restauration :

```
bash backup.sh vrp-rep
```

Cela aura pour effet de sauvegarder l'application **vrp-rep**.

3 Comment interagir avec le serveur

Dans cette partie, je vais expliquer comment le serveur fonctionne avec Git. Vous pouvez déjà visualiser l'organisation de mon Git ici : 1 (Chapitre 8) .
Lorsqu'un push est fait sur le serveur, il y a un hook qui se déclenche. Un hook permet de réaliser des actions (souvent un script) lors de certains événements (ici le push).
Le hook fait les actions suivantes :

— Récupération de la dernière version des sources pour la branche en question

— Installation des nouvelles dépendances si besoin (avec Composer install)

— Suppression du cache (php app/console cache :clear –env=prod –no-debug)

— Migration Doctrine (php app/console –no-interaction doctrine :migrations :migrate –env=prod)

— Installation des assets

Lorsqu'un push est fait sur la beta, la beta est automatiquement mise à jour.

La version production fonctionne avec les tags Git, il faut donc push les tags avant afin que lorsque le hook se déclenche il existe un nouveau tag à récupérer.

Pour ce faire, vous pouvez suivre cette série de commande :

```
1 git tag versionDuTag shaDuDernierCommit
2 git push --tags
```

4 Conseils pendant le développement

Pour développer sur l'application, je conseille une séparation en branche pour chaque fonctionnalité (comme je l'ai décrit 1 (Chapitre 8)).
Pour cela, il peut être intéressant d'utiliser GitFlow.

Git Flow permet facilement de créer des branches suivant certaines conventions, de faire les merges facilement, de faire les pulls. Il offre une couche d'abstraction.

Je conseille de développer sous un système UNIX (Linux ou OSX), l'utilisation des commandes ainsi que la gestion des droits y sont plus aisées.

J'ai utilisé l'IDE PHPStorm qui est développé par JetBrains. L'éditeur est payant, mais il est possible d'avoir une version gratuite en étant étudiant.
Cet IDE couplé au plugin **Symfony2** permet de naviguer facilement dans les fichiers. Par exemple : depuis une vue il est très facile de remonter au contrôleur correspondant et ce même si l'on ne le connaît pas. De la même manière, il est possible de faire l'opération inverse, c'est-à-dire de passer d'une action d'un contrôleur à la vue associée.

Je conseille aussi d'utiliser Maildev qui permet de ne pas envoyer les mails, mais des les récupérer dans une interface web, ce qui est très pratique pour le développement.

Conclusion

En conclusion, la partie recherche m'a permis de découvrir des nouvelles technologies comme Symfony et Doctrine mais j'y ai aussi découvert le problème VRP (problème de tournées de véhicules).
De plus, cette partie m'a permis d'améliorer mes compétences en gestion de projet, grâce au découpage en tâche et à la création d'un planning avec un diagramme de Gantt.

Les objectifs

Pour ce projet, les objectifs ont été atteints. J'ai implémenté toute les fonctionnalités décrite dans la section 2.
Certaines fonctionnalités sont déjà utilisées en production tandis que d'autres sont encore sur la beta, afin que le comité puisse tester et valider le développement. Une fois le développement validé, je vais les déployer sur la production.

Bilan du calendrier

Concernant le planning, je vous rappelle le planning prévisionnel (1 (Chapitre 7)).

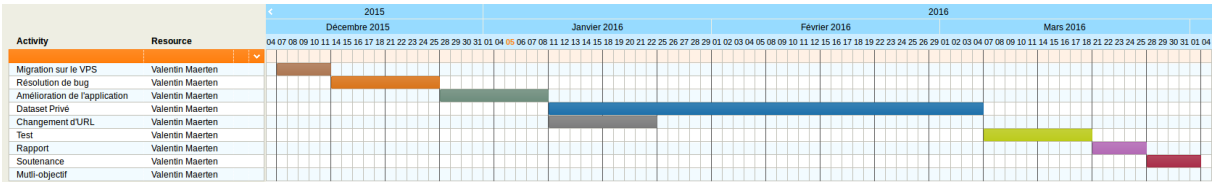


Figure 1 – *Planning prévisionnel*

Le planning réel est le suivant :

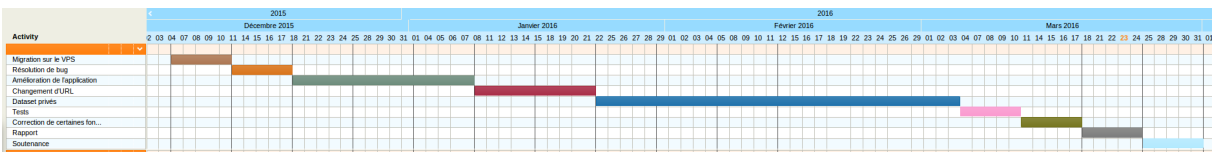


Figure 2 – *Planning Réel*

Ces plannings sont des macros-plannings, pour avoir le détail du travail effectué chaque semaine vous pouvez vous référer aux comptes rendu hebdomadaires.

On peut noter 3 différences entre ces plannings :

— La résolution de bug a été plus rapide que prévu, mais en contre parti j'ai passé plus de temps sur l'amélioration de l'application

— Le changement d'URL a été fait avant les datasets privés et non en même temps

— Les tests généraux ont durées moins longtemps mais une semaine a été dédié aux corrections que mon encadrant m'a demandé d'effectuées.

Ressenti personnel

Sur le plan personnel, je suis très content d'avoir pu travailler sur ce projet. Cela a été enrichissant car je ne connaissais ni Symfony2 ni son écosystème (Twig, Doctrine, etc).
De plus, travailler sur une application déjà en production est très valorisant pour mon projet professionnel car l'application étant utilisé par de nombreux utilisateurs, il faut faire très attention à ne pas déployer de code erroné.

Comptes rendus hebdomadaires

Compte rendu n°1 du 17/09/2015

Contexte et objectifs du projet

Le site VRP-REP permet de centraliser les instances des problèmes avec un unique format. Ainsi que de permettre au contributeur de proposer leurs solutions afin de comparer les différents algorithmes et d'éventuellement connaître le score optimal sur une instance.

Actuellement, dans le site, il n'y possible que d'avoir uniquement une seule fonction objectif alors que certains problèmes nécessitent plusieurs fonctions objectif. Par exemple, minimiser le nombre de camions ainsi que la durée totale de la tournée.

Le site étant fait avec Symfony2, je dois donc dans un premier temps apprendre Symfony2. Puis par la suite, étudier le code de l'actuel afin de le comprendre. Enfin, proposer une solution pour pouvoir ajouter plusieurs fonctions objectifs par instance et l'implémenter.

Ce que j'ai fait

Le site étant fait avec Symfony2, j'ai commencé par installer tout le stack nécessaire (LAMP, ACL, etc) et tout configuré. Puis par la suite, j'ai commencé mon apprentissage sur Symfony avec le site officiel (`https://symfony.com`).

Ce que je compte faire

Je compte continuer mon apprentissage sur Symfony2 et une fois que j'aurais accès au code l'étudier afin de mieux comprendre l'existant. Nous aurons une réunion avec la personne qui a créé le site à l'origine le jeudi suivant (24 septembre).

Compte rendu n°2 du 24/09/2015

Ce que j'ai fait

Mon encadrant m'a mis en contact avec celui qui a créé le site à l'origine (Hubert) afin qu'il puisse m'autoriser à récupérer les sources du projet. J'ai ensuite cloné le projet, configuré les paramètres. Puis par la suite j'ai étudié les sources afin de commencer à comprendre le code.

Le jeudi, Hubert est venu sur Tours afin de faire le point avec nous. En première partie de la journée, nous avons fait un point sur l'actuel du projet et défini mes tâches pour les semaines à venir. Puis dans la deuxième partie, Hubert m'a aidé à tout configurer et à comprendre le workflow du projet.

Ce que je compte faire

Je vais étudier les différentes offres de VPS (Virtual Private Server) chez les différents hébergeurs afin de les présenter à mon encadrant. Dans le but de migrer de l'hébergement mutualisé vers un VPS.

Compte rendu n°3 du 01/10/2015

Ce que j'ai fait

Durant cette semaine, j'ai fait un comparatif de tous les VPS du marché correspondant au budget. J'ai comparé les processeurs, la mémoire vive, la mémoire disque, le type de disque (SSD ou HDD) et le prix.
J'ai aussi mis les avantages et inconvénients pour chacun d'entre eux et fait un tableau récapitulatif.
Par la suite, j'ai regardé les commentaires des utilisateurs sur différents forums afin d'avoir un retour (surtout pour les hébergeurs très peu chers, arnaque ?)
Pour finir, j'ai commencé à installer LAMP et l'application sur mon propre VPS.

Ce que je compte faire

La semaine prochaine, je compte me documenter sur les bundles utilisateur (FOSUserBundle & Sonata User Bundle) pour la résolution de bug

Compte rendu n°4 du 08/10/2015

Ce que j'ai fait

J'ai découvert qu'il n'y avait pas de validation côté serveur au niveau du formulaire d'inscription ainsi que du formulaire pour éditer son profil.
Je me suis alors documenté sur les différents bundles (FOSUserBundle et SonataUserBundle) afin d'ajouter de la validation.
Cela me permettra de mieux comprendre l'application et le code afin d'être efficace durant la deuxième partie de ce PR&D.

J'ai au une réunion avec mon encadrant afin de lui montrer ce qui a été fait et de définir les prochaines étapes.

Ce que je compte faire

Afin de voir si la potentielle migration sur un VPS résoudrait les problèmes, je vais installer l'application sur mon serveur personnel. Une fois cette installation faite, il faudra la tester.

Je vais aussi commencer à faire des recherches pour la procédure de migration (installation de LAMP, de Symfony2, de Git etc)

Compte rendu n°5 du 15/10/2015

Ce que j'ai fait

J'ai installé l'application sur mon serveur. Et après de nombreuse, réinstallation je commence à avoir une procédure de migration qu'il faudra bien sûr adapter au serveur choisi.

J'ai commencé à étudier les petites résolutions de bug par exemple à chaque fois qu'un dataset est téléchargé il est téléchargé forcement avec l'extension .xml quelque soit son extension de base afin d'en estimer le temps de résolution.

Les résolutions de bug me permettent surtout de comprendre le code.

Ce que je compte faire

Continuer à étudier le code et à résoudre les petits bugs

Compte rendu n°6 du 22/10/2015

Durant cette séance, j'ai étudié l'upload des datasets ainsi que la vérification des instances afin de comprendre son fonctionnement.
J'ai remarqué que lors que la vérification des instances prend du temps (i.e : gros fichier ou beaucoup de fichiers), il n'y a aucune information permet à l'utilisateur de savoir si l'application n'est pas plantée. J'ai donc soumis l'idée à M. Mendoza de mettre un loader durant la vérification des instances et afficher le message d'erreur si une erreur s'est produite.

Compte rendu n°7 du 04/11/2015

Durant cette séance, j'ai étudié le livre que m'a prêté M. Mendoza afin de m'informer sur le problème du Vehicle Routing Problem

Compte rendu n°8 du 12/11/2015

J'ai mis en forme le guide de migration.

Compte rendu n°9 du 18/11/2015

Durant cette séance, j'ai commencé à rédiger le cahier des charges.

Compte rendu n°10 du 25/11/2015

Durant cette séance, j'ai fini le cahier des charges et j'ai continué le rapport.

Compte rendu n°11 du 03/12/2015

Durant cette séance, j'ai bien avancé sur le rapport.

Compte rendu n°12 du 10/12/2015

J'ai effectué la migration de l'application sur le VPS. M. Mendoza ainsi que le comité vont tester afin de valider la migration et je vais par la suite changer les DNS pour les faire pointer sur le nouveau serveur.

Compte rendu n°13 du 17/12/2015

J'ai commencé la résolution de bug. Cette semaine, je me suis confronté à l'ajout de la validation côté serveur par le biais de Sonata User Bundle. L'affiliation est maintenant obligatoire.

Compte rendu n°14 du 07/01/2016

J'ai fini de résoudre les comportements non souhaités de l'application. Avant les instances étaient forcement téléchargées avec l'extension .xml, maintenant l'extension est dynamique. J'ai ajouté une barre de chargement lors de la validation des fichiers lors de l'upload de datasets ainsi qu'une gestion des erreurs.

Compte rendu n°15 du 14/01/2016

J'ai mis en place une newsletter pour les utilisateurs. Les fonctionnalités sont :

- Possibilité de choisir de s'y inscrire ou non lors de la création de compte
- Possibilité de changer d'avis dans la page du profile
- Création d'une commande pour enlever tout les utilisateurs de la liste de souscription (sera utilisé sur la beta)

Compte rendu n°16 du 21/01/2016

J'ai développé un système qui envoie un mail aux utilisateurs inscrit à la newsletter lorsqu'un dataset ou une solution est déposé.

Compte rendu n°17 du 28/01/2016

Cette semaine j'ai développé une fonctionnalité qui envoie un mail aux administrateur à chaque fois qu'une personne s'inscrit. J'ai également homogénéiser les noms d'utilisateurs (migration de la base de données + changement automatique lors de l'inscription).

Compte rendu n°18 du 04/02/2016

J'ai modifié le comportement de l'application lors de l'upload des datasets, il n'est désormais plus possible d'uploader des datasets autres qu'au format ZIP. J'ai commencé a développer le changement d'URL pour les datasets (ajout d'un ID unique pour les datasets qui est différents de l'id en base).

Compte rendu n°19 du 11/02/2016

J'ai fini le changement d'URL et commencé les datasets privés (base de données ainsi que le filtrage sur la liste des datasets).

Compte rendu n°20 du 25/02/2016

Durant les vacances, ainsi que cette semaine j'ai pas mal avancé, j'ai développé la page pour voir les datasets privés grâce au token en faisant bien attention à la sécurité (pas visible pas les autres mais uniquement par l'auteur ou une personne en possession du token). J'ai également modifié le formulaire d'upload des datasets afin de respecter les règles données par mon encadrant suivant les références et variantes.

Compte rendu n°21 du 03/03/2016

Durant cette semaine, j'ai géré la sécurité au niveau des pages de consultation des variantes / références afin qu'elle ne soit pas visible par les utilisateurs n'y ayant pas le droit.

Compte rendu n°22 du 10/03/2016

J'ai améliorer la gestion de la page des datasets (privés ou non) demandé par mon encadrant. J'ai aussi ajouté un bouton Copy To Clipboard pour mettre dans le presse papier le lien de partage du dataset privé.

Compte rendu n°23 du 17/03/2016

J'ai continué mon rapport.

Compte rendu n°24 du 24/03/2016

Le mecredi j'ai fini mon rapport. Le jeudi, j'ai codé la fonctionnalité qui va envoyer un mail aux auteurs possédant des dataset privés qui le sont depuis plus de 6 mois afin de leurs proposés de les passer en public.

Webographie

[WWW1] 1&1. *1&1 - Packs des Serveurs Virtuels*. URL : http://www.1and1.fr/serveurs-virtuels-packs?__lf=Order-Product#stage-end.

[WWW2] Behost. *VPS Classic : VPS pas cher de haute performance - Behost*. URL : https://www.behost.fr/vps-classic/.

[WWW3] *Composer*. URL : https://getcomposer.org/.

[WWW4] Doctrine. *Doctrine*. URL : http://www.doctrine-project.org.

[WWW5] Doctrine. *Doctrine DBAL*. URL : http://docs.doctrine-project.org/projects/doctrine-dbal/en/latest/reference/introduction.html.

[WWW6] Jorge E. Mendoza. *VRP-REP : the vehicle routing problem repository*. URL : https://github.com/VRP-REP.

[WWW7] OVH. *VPS SSD : Le VPS pas cher & haute performance*. URL : https://www.ovh.com/fr/vps/vps-ssd.xml.

[WWW8] Pulseheberg. *Pulseheberg - Location et hebergement de serveurs VPS*. URL : https://www.pulseheberg.com/vps/simple.

[WWW9] SensioLabs. URL : http://symfony.com/doc/current/bundles/FOSUserBundle/index.html.

[WWW10] SensioLabs. *Composants Symfony*. Sous la dir. de SensioLabs. URL : https://symfony.com/fr/components.

[WWW11] SensioLabs. *Create your First Page in Symfony*. Sous la dir. de SensioLabs. URL : https://symfony.com/doc/current/book/page_creation.html.

[WWW12] SensioLabs. *Qu'est ce que Symfony*. Sous la dir. de SensioLabs. URL : https://symfony.com/fr/what-is-symfony.

[WWW13] SensioLabs. *Routing (The Symfony Book)*. URL : http%20://symfony.com/doc/current/book/routing.html.

[WWW14] SensioLabs. *Symfony and HTTP Fundamentals*. URL : https://symfony.com/doc/current/book/http_fundamentals.html.

[WWW15] SensioLabs. *Symfony : Controller*. URL : http://symfony.com/doc/current/book/controller.html.

[WWW16] SensioLabs. *Symfyony : Creating and Using Templates*. URL : http://symfony.com/doc/current/book/templating.html.

[WWW17] SensioLabs. *The Bundle System*. URL : https://symfony.com/doc/current/book/bundles.html.

[WWW18] *Sonata-project user's documentation*. URL : https://sonata-project.org/bundles/user/2-2/doc/index.html.

[WWW19] Friends Of Symfony. *FOSUserBundle*. URL : https://github.com/FriendsOfSymfony/FOSUserBundle.

Bibliographie

[1] David L. Applegate, Robert E. Bixby, Vasek Chvatal et William J. Cook. *The traveling Salesman Problem : A computational Study*. Sous la dir. de Princeton University Press. 2006.

[2] Nicola Secomandi et Francois Margot. « Reoptimization Approaches for the Vehicle Routing Problem with Stochastic Demands ». In : (Submitted : December 2005 ; Revised : April 2007, July 2007).

[3] Paolo Toth et Daniele Vigo. *Vehicle Routing Problems, Methods, and Applications seconde edition*. 2014.

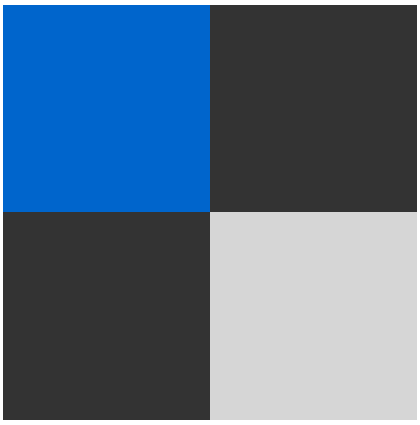
VRP-REP

Valentin Maerten

Encadrement : Jorge Mendoza

Existant

Projet sur le site VRP-REP (www.vrp-rep.org) qui contient des jeux de données ainsi que des solutions proposées par les utilisateurs pour le problème du VRP (problème de tournées de véhicules).



Logo du site VRP-REP

Objectif

- Migrer le site depuis l'hébergement mutualisé vers un VPS
- Possibilité d'ajouter des jeux de données privés (actuellement tout est public)
- Développement de petites fonctionnalités annexe
- Réaliser un guide de migration qui sera fourni pour une future migration

Logo de Debian

Technologies utilisées

- Symfony2
- Doctrine
- MySQL
- Environnement Linux (Debian)

Logo de Symfony

Ecole Polytechnique de l'Université François Rabelais de Tours
Département Informatique
64 avenue Jean Portalis, 37200 Tours, France
www.polytech.univ-tours.fr

POLYTECH TOURS
Informatique

VRP-REP

Résumé

Ce projet recherche et développement est réalisé en DI5. Ce projet porte sur le site VRP-REP. VRP-REP contient des jeux de données ainsi que des solutions pour le problème de tournée de véhicules. Ce site a été développer grâce à Symfony2 et Doctrine avec une base de données MySQL. Ce site est actuellement hébergé sur un hébergement mutualisé.

Les objectifs de ce projet sont multiples. L'objectif principal est de pouvoir avoir des jeux de données privés, car actuellement lors qu'un utilisateur dépose un jeu de données il est visible par tout le monde. De plus, l'hébergement mutualisé bride le site (à cause de la limite d'upload ainsi que la quantité de mémoire allouée à PHP), je vais donc migrer le site sur un VPS que nous aurons choisi au préalable. Pour finir, je vais faire des corrections de bug ainsi que du développement de petites fonctionnalités annexe.

Dans ce rapport, j'ai décrit en détail l'ensemble de mes tâches ainsi que leurs planifications avec un diagramme de Gantt. Durant ce projet, j'ai eu un contact régulier avec mon encadrant, de plus, les comptes-rendus hebdomadaires permettent de suivre le bon avancement du projet.

Mots-clés

Problème de tournée de véhicules, VRP-REP, PHP, Symfony2, Doctrine, MySQL, Linux, VPS, PRD, Projet recherche et développement, Cahier des charges, Diagramme de Gantt, Migration

Abstract

This research and development projet has been completed in DI5. This projet is related to the website VRP-REP. VRP-REP contains data sets and solutions for the vehicle routing problem. This website has been developped thanks to Symfony2 and Doctrine with a MySQL database. This website is currently hosted by shared hosting.

Objectives for this project are multiples. The main objective is to allow users to publish private data sets, because currently when a user publish a data set on VRP-REP, the data set is visible for everyone. Moreover, the shared hosting restrain the VRP-REP website (mainly because of the upload limite and the amout of memory grant to PHP), so i will migrate this website on a VPS that we have pre-selected. To finish, I will correct some little bug and develop some little additionalfeature.

In this report, I described in detail all my taks and their planification with a Gantt diagram. During this project, I had a regular contact with my supervisor, furthermore, the weeklyreports monitors the progress of the project.

Keywords

VRP, Vehicle Routing Problem, VRP-REP, PHP, Symfony2, Doctrine, MySQL, Linux, VPS, PRD, Research and development project, Specifications, Gantt diagram, Migration

Tuteurs académiques
Jorge Mendoza

Étudiants
Valentin Maerten (DI5)