# Documentation

Aide à l'utilisation et l'amélioration

# Doletic - API

etic
INSA LYON
Junior-Entreprise

Amaris

EY

SOCIETE GENERALE

# Table des matières


**Introduction** ..... **4**
1. **Contributeurs de Doletic – Partie API** ..... **4**
2. **Intérêt de Doletic** ..... **4**
3. **Choix techniques et justification** ..... **4**
4. **Guide de style et conventions** ..... **5**
5. **Organisation générale** ..... **5**

**Description du modèle** ..... **6**
1. **KernelBundle** ..... **6**
   a. Contenu général ..... 6
   b. UML ..... 7
   c. Entités ..... 8
2. **RHBundle** ..... **13**
   a. Contenu général ..... 13
   b. UML ..... 14
   c. Entités ..... 15
3. **GRCBundle** ..... **17**
   a. **Contenu général** ..... 17
   b. UML ..... 18
   c. Entités ..... 19
4. **SupportBundle** ..... **21**
   a. Contenu général ..... 21
   b. UML ..... 22
   c. Entités ..... 23
5. **UABundle** ..... **24**
   a. Contenu général ..... 24
   b. UML ..... 25
   c. Entités ..... 26

**Gestion des droits et routes** ..... **32**
1. **Fonctionnement général des routes** ..... **32**
2. **Niveaux de droits** ..... **32**
3. **Comment ajouter/modifier les rôles** ..... **33**

**Lien avec l'API OVH** ..... **34**
1. **Présentation de l'API** ..... **34**
   a. Intérêt ..... 34
   b. Liens utiles ..... 34
2. **Services reliés à l'API** ..... **34**
   a. Constantes ..... 34
   b. Paramètres en entrée ..... 34
   c. Appels à l'API ..... 34

**Publipostage** ..... **36**
1. **Présentation et utilisation de PHPWord** ..... **36**
2. **Remplissage des dictionnaires** ..... **36**
3. **Gérer les documents types** ..... **36**

**Upload de fichiers** ..... **37**
1. **Service et Listener** ..... **37**
2. **Note sur l'upload de fichiers** ..... **37**

**Configuration** ..... **38**
**1. Configuration générale** ..... **38**
**2. Paramètres** ..... **38**
a. Que sont les paramètres ? ..... 38
b. Paramètres et confidentialité ..... 38
c. Quand utiliser des paramètres ? ..... 38
**3. Sécurité** ..... **39**
**4. Routes** ..... **39**
**Fonctionnalités en projet** ..... **40**
**1. Gestion simple des indicateurs** ..... **40**
**2. Base de données de consultants potentiels** ..... **40**

# Introduction

## 1. Contributeurs de Doletic – Partie API

Cette documentation a pour but de permettre de poursuivre le projet Doletic sans aide, mais il est souvent plus facile d'interroger directement le développeur sur sa création. Voici donc les personnes que tu peux contacter pour t'aider :

- **Paul DAUTRY** (Responsable DSI 2015-2016) : Initiateur du projet, *n'a pas contribué à l'implémentation Symfony* mais a été à l'origine des choix généraux
- **Nicolas SORIN** (Responsable DSI 2016-2017) : Créateur de l'API et auteur de cette documentation. Principal développeur de Doletic.

## 2. Intérêt de Doletic

Doletic est l'ERP d'ETIC. Il a été conçu pour être facile à maintenir et pour pouvoir s'adapter aux changements dans la structure. Il est vital pour le bon fonctionnement de la J.E. : il permet de suivre toutes les études, toutes les inscriptions, les contacts entreprise, de suivre les indicateurs, et surtout de publiposter des documents pré-remplis à partir des données stockées.

Le responsable DSI doit obligatoirement maîtriser Doletic, et être capable d'apprendre aux autres à s'en servir et à l'améliorer. **Cette documentation doit donc être tenue à jour au fur et à mesure des modifications apportées.**

## 3. Choix techniques et justification

La version actuelle de Doletic est séparée en deux parties pour faciliter sa maintenance. Cette documentation traite de la partie serveur. Il est à noter qu'il est parfaitement possible de modifier le client (ou d'ajouter de nouveaux clients) sans toucher à cette partie, à condition de savoir comment l'utiliser.

Le langage PHP a été choisi car c'est le langage le plus répandu en back-end, et il est relativement facile à apprendre. PHP seul est très permissif, et il est nécessaire d'utiliser un Framework pour une utilisation propre et sécurisée. Symfony a été choisi pour sa popularité et sa robustesse. C'est un Framework assez compliqué à maitriser, il faut donc nécessairement former les membres à son utilisation. Le choix du Framework a longtemps été délicat, car il ajoute une couche de complexité et force les membres à apprendre plus de choses ; cependant, Doletic étant un élément central d'ETIC, il nécessite d'être bien structuré et sécurisé.

DoleticREST s'appuie sur des Bundles Symfony populaires, et une dépendance non bundle :

- JMSSerializerBundle pour la sérialisation au format JSON des données de retour
- FOSUserBundle pour les utilisateurs
- FOSRESTBundle pour l'organisation de l'API
- FOSOAuthServerBundle pour sécuriser l'accès à l'API
- NelmioApiDocBundle pour générer la documentation des routes
- NelmioCorsBundle pour faire le lien avec le front-end
- DoctrineFixtureBundle pour les données de test
- ZoddoOVHBundle pour utiliser facilement l'API OVH
- PHPOffice/PHPWord pour le publipostage (ce n'est pas un bundle)

Ces Bundles ont leur propre documentation, disponible en ligne.

# 4. Guide de style et conventions

D'une manière générale, il convient de structurer le code selon les conventions et best-practice Symfony. L'indentation suit le style de l'indentation automatique de PhpStorm, et chaque fonction doit être documentée avec ses paramètres d'entrée, de sortie, ainsi que ce qu'elle fait.

Tous les noms de variables/fonctions/classes sont en anglais, les seules entorses à la règle étant les acronymes utilisés par ETIC : RH, GRC, UA. A noter que GRC (« Gestion Relation Client ») correspond à un pôle d'ETIC qui a été supprimé au moment où cette documentation est rédigée.

Chaque entité a son contrôleur, dont le nom est celui de l'entité + « Controller ». Si tu hésites entre deux contrôleurs pour une route, respecte ces règles simples :

- GET : Le contrôleur est défini par le format du retour. Une ou des entités « Exemple » en retour signifie que l'action doit être dans le contrôleur « ExempleController »
- POST/PUT/DELETE : Le contrôleur est défini par l'entité en entrée.

Par exemple, si tu veux rechercher les projets (« Project ») pour une société (« Firm »), le retour est un tableau de projets, donc la route va dans le contrôleur « ProjectController » et pas dans le « FirmController ».

Au niveau des données elles-mêmes, il faut particulièrement faire attention à deux points :

- La validation des données en entrée. Le principal problème des versions antérieures vient de ce point, car des données invalides et inexploitables sont présentes dans la BDD
- La cohérence des données. Attention, lorsque certaines données sont susceptibles de créer une redondance, il vaut mieux essayer de trouver un autre moyen. Si ce n'est pas possible, il faut s'assurer que tout reste cohérent (pas deux valeurs incompatibles).

# 5. Organisation générale

L'API est organisée en Bundles. Ces Bundles ne sont **pas** indépendants, ils correspondent à une séparation logique, qui apparaît dans les routes. Cette séparation se base sur l'activité liée au Bundle. Par exemple, le suivi d'étude a son Bundle, le suivi des membres (RH) également. Chaque Bundle contient des dossiers suivant l'organisation classique d'un projet Symfony : Entity, Controller, Service, etc.

Une partie de cette documentation sera dédiée à la configuration.

# Description du modèle

## 1. KernelBundle

### a. Contenu général

Ce Bundle regroupe les entités de base nécessaire au fonctionnement de Doletic. Ces entités comprennent :

- Le système d'authentification, tel qu'indiqué dans les tutoriels d'utilisation de FOSOAuthServerBundle
- L'entité utilisateur, héritant de l'entité de base de FOSUserBundle
- Les entités génériques utilisées ailleurs, comme les pays, civilités, etc.
- L'entité de base correspondant aux modèles de documents utilisés pour le publipostage.
- Les entités correspondant au système de listes de diffusion

**Liste des entités :**

- User
- Position
- Division
- Gender
- Country
- UserPosition
- Setting
- DocumentTemplate
- MailingList
    - CustomMailingList
    - DivisionMailingList
    - PositionMailingList
    - TeamMailingList

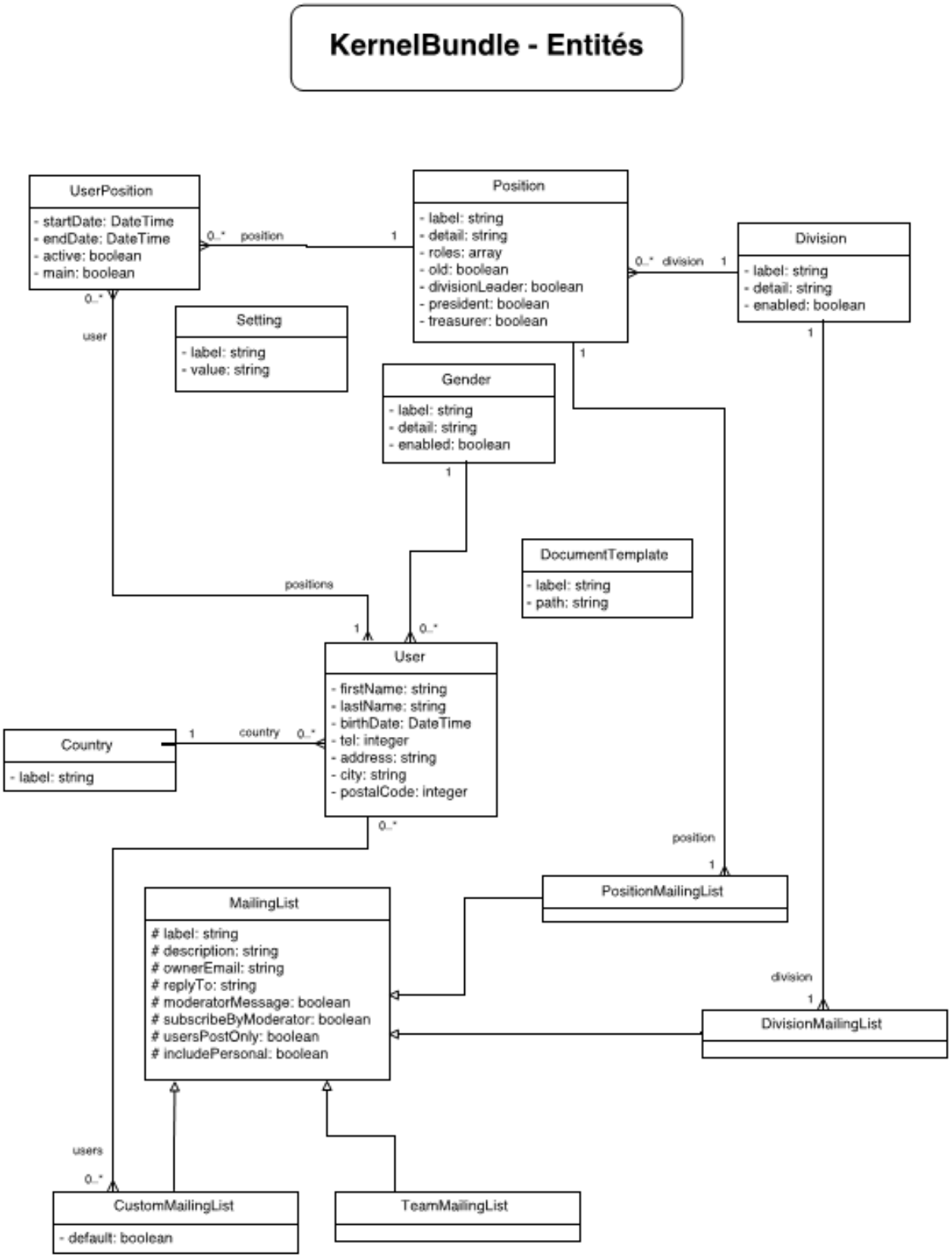
**Entités par défaut (Bundle externe)**

- Client
- AccessToken
- RefreshToken
- AuthCode

Pour voir comment utiliser ces entités :

https://github.com/FriendsOfSymfony/FOSOAuthServerBundle/blob/master/Resources/doc/index.md

## b. UML


KernelBundle - Entités
UserPosition
- startDate: DateTime
- endDate: DateTime
- active: boolean
- main: boolean
0..*
position
1
Position
- label: string
- detail: string
- roles: array
- old: boolean
- divisionLeader: boolean
- president: boolean
- treasurer: boolean
0..*
division
1
Division
- label: string
- detail: string
- enabled: boolean
0..*
user
Setting
- label: string
- value: string
Gender
- label: string
- detail: string
- enabled: boolean
1
1
1
DocumentTemplate
- label: string
- path: string
positions
1
0..*
User
- firstName: string
- lastName: string
- birthDate: DateTime
- tel: integer
- address: string
- city: string
- postalCode: integer
Country
- label: string
1
country
0..*
0..*
position
1
PositionMailingList
MailingList
# label: string
# description: string
# ownerEmail: string
# replyTo: string
# moderatorMessage: boolean
# subscribeByModerator: boolean
# usersPostOnly: boolean
# includePersonal: boolean
division
1
DivisionMailingList
users
0..*
CustomMailingList
- default: boolean
TeamMailingList

## c. Entités

| **User** | | |
|---|---|---|
| Correspond à un utilisateur de Doletic ainsi qu'à la personne physique correspondante. Cette entité hérite de l'entité BaseUser du FOSUserBundle, dont la documentation peut être trouvée sur le site de Symfony. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| firstName | String | Prénom de l'utilisateur |
| lastName | String | Nom de famille de l'utilisateur |
| birthdate | Date | Date de naissance de l'utilisateur |
| tel | Integer | Numéro de téléhpone personnel de l'utilisateur |
| address | String | Corps de l'adresse de l'utilisateur (type « 20 Avenue Albert Einstein ») |
| city | String | Ville de l'utilisateur |
| postalCode | Integer | Code postal de l'utilisateur |
| schoolYear | SchoolYear | Année d'étude parmis celles possibles. |
| department | Department | Département de l'école parmis ceux possibles |
| gender | Gender | Civilité de l'utilisateur |
| country | Country | Pays de l'utilisateur |
| recruitmentEvent | RecruitmentEvent | Session de recrutement correspondant à l'arrivée de l'utilisateur à la J.E. |
| consultantMembership | ConsultantMembership | Adhésion consultant de l'utilisateur (si c'est un consultant) |
| administratorMemberships | AdministratorMembership[] | Adhésions administrateur (membre actif) de l'utilisateur (si c'est un membre actif) |
| positions | Position[] | Poste de l'utilisateur à la J.E. |
| *Méthodes spécifiques* | | |
| getRoles() | String[] | Renvoie le tableau des rôles de l'utilisateur. Les rôles sont récupérés par fusion des rôles de toutes les positions actives de l'utilisateur. |
| *Listener* | | |
| prePersist() | A la création d'un utilisateur :<br>- Son nom d'utilisateur est généré automatiquement à partir de son prénom et de son nom, de manière itérative pour éviter les doublons.<br>- Son mot de passe est généré aléatoirement<br>- Un mail de confirmation est envoyé à l'adresse mail renseignée, qui contient le nom d'utilisateur et le mot de passe | |

| Position | | |
|---|---|---|
| Correspond au poste d'un utilisateur dans la J.E., et est directement relié aux droits de l'utilisateur dans Doletic. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| label | string | Nom abrégé du poste. |
| detail | string | Nom complet du poste. |
| roles | string[] | Roles dans l'application, associés au poste. |
| division | Division | Pôle de la J.E. associé au poste. |
| old | boolean | Indique si la position correspond à un ancien membre. |
| divisionLeader | boolean | Indique si le poste correspond au responsable de son pôle. |
| president | boolean | Indique si le poste est équivalent au poste de président. Utilisé uniquement pour le publipostage de documents nécessitant la signature du président. |
| treasurer | boolean | Indique si le poste est équivalent au poste de trésorier. Utilisé uniquement pour le publipostage de documents nécessitant la signature du trésorier. |
| *Listener* | | |
| postPersist() | Après la création d'un poste :<br>- Si le poste correspond à un ancien membre, il ne peut pas être président ou trésorier<br>- Si le poste correspond au président, il ne peut pas être trésorier<br>- Si le poste créé un doublon de président/trésorier, il n'est pas accepté en tant que président/trésorier | |
| postUpdate() | Les même vérifications sont effectuées à la mise à jour du poste. | |

| Division | | |
|---|---|---|
| Correspond à un pôle de la J.E. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| label | string | Nom abrégé du pôle. |
| detail | string | Nom complet du pôle. |
| enabled | boolean | Indique si le pôle est actuellement en fonctionnement dans la J.E. Si cet attribut vaut false, aucune autre entité ayant une association avec une Division ne peut être associée à l'entité. |

| Gender | | |
|---|---|---|
| Correspond à la civilité d'une personne physique. Peut correspondre à autre chose que Monsieur ou Madame (par exemple « maître » ou « docteur »). | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| label | string | Nom abrégé de la civilité, par exemple M. pour monsieur. |
| detail | string | Nom complet de la civilité. |
| enabled | boolean | Indique si la civilité est actuellement en utilisable dans la J.E. Si cet attribut vaut false, aucune autre entité ayant une association avec un Gender ne peut être associée à l'entité. |

| UserPosition | | |
|---|---|---|
| Correspond à l'association entre un utilisateur et une position (poste dans Etic). Un utilisateur peut cumuler plusieurs postes. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| startDate | DateTime | Date de prise de poste de l'utilisateur. |
| endDate | DateTime | Date de fin de poste de l'utilisateur (si le poste n'est plus d'actualité). |
| active | boolean | Indique si l'utilisateur est actuellement à ce poste. |
| main | boolean | Indique si ce poste est le poste principal de l'utilisateur. Le poste principal a pour vocation d'être le poste affiché avec le nom de l'utilisateur. |
| position | Position | Poste de l'utilisateur. |
| user | User | Utilisateur concerné. |
| *Listener* | | |
| postPersist() | A la création d'une association utilisateur-poste :<br>- La date de prise de poste est renseignée automatiquement<br>- Si le poste est celui d'un ancien membre, tous les autres postes non anciens sont marqués comme inactifs et non poste principal<br>- Si l'utilisateur n'était pas déjà un ancien membre et que ce poste est celui d'un ancien membre, le nouveau poste est marqué comme principal<br>- Si le poste n'est pas celui d'un ancien membre, tous les postes d'ancien sont passés en inactifs<br>- Si le poste est marqué comme principal, tous les autres postes ne sont pas le poste principal. | |
| postUpdate() | A la modification de l'association :<br>- Si le poste devient le poste principal de l'utilisateur, tous les autres postes ne le sont pas. | |

| Country | | |
|---|---|---|
| Correspond à un pays existant. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| label | string | Nom du pays. |

| Setting | | |
|---|---|---|
| Correspond à un paramètre sauvegardé, et ayant pour vocation d'être modifié dans un formulaire. Les clés et valeurs sont des chaînes de caractères, pour les rendre adaptables. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| label | string | Nom du paramètre. |
| value | string | Valeur du paramètre. |

| DocumentTemplate | | |
|---|---|---|
| Correspond à un modèle de document pouvant être publiposté via un service dédié (voir partie publipostage). | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| label | string | Nom du document. |
| path | string | Chemin d'accès au fichier du template. |

| MailingList | | |
|---|---|---|
| Correspond à une liste de diffusion. Cette entité est abstraite car elle n'est pas faite pour être utilisée telle quelle. Certains attributs correspondent directement à des paramètres utilisés par l'API OVH. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| label | string | Nom de la liste. |
| description | string | Description détaillée de la liste. |
| ownerEmail | string | Adresse électronique du propriétaire. |
| replyTo | string | Adresse utilisée si quelqu'un répond à un mail envoyé à cette liste |
| moderatorMessage | boolean | Indique si les messages envoyés à cette liste doivent être modérés par le propriétaire de la liste. Si c'est le cas, le propriétaire reçoit un mail lui demandant de confirmer chaque envoi. |
| subscribeByModerator | boolean | Indique s'il faut qu'un modérateur inscrive le membre à la liste. |
| usersPostOnly | boolean | Indique si |
| includePersonal | boolean | Indique si la liste inclut les adresses personnelles des utilisateurs (ou simplement leurs adresses J.E.). |

| CustomMailingList | | |
|---|---|---|
| Correspond à une liste de diffusion dont les membres sont ajoutés un par un (pas de regroupement spécifique). Cette entité hérite de l'entité MailingList | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| users | User[] | Liste des utilisateurs concernés |
| default | boolean | Indique s'il s'agit d'une liste par défaut. Une liste par défaut est générée directement via un service dans le code est n'est donc pas gérée par les utilisateurs de l'API. Cela correspond entre autres à la liste « membres », qui regroupe tous les membres non anciens. |

| DivisionMailingList | | |
|---|---|---|
| Correspond à une liste de diffusion reliée à un pôle de la J.E. Cette entité hérite de l'entité MailingList. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| division | Division | Pôle concerné. |

| PositionMailingList | | |
|---|---|---|
| Correspond à une liste de diffusion reliée à un poste dans la J.E. Cette entité hérite de l'entité MailingList. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| position | Position | Poste concerné. |

| TeamMailingList | | |
|---|---|---|
| Correspond à une liste de diffusion reliée à une équipe de la J.E. Cette entité hérite de l'entité MailingList. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| team | Team | Equipe concernée. |

# 2. RHBundle

## a. Contenu général

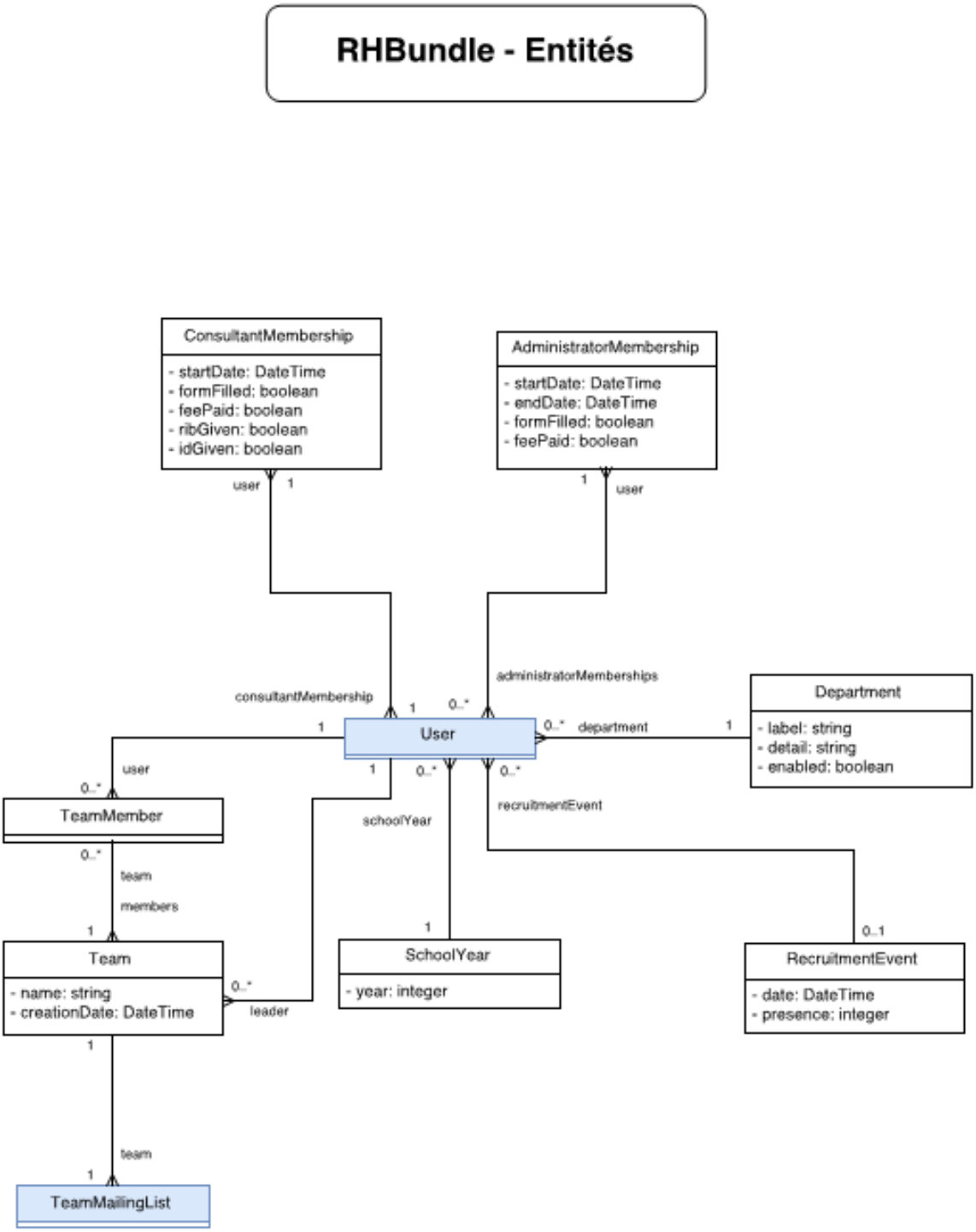
Ce Bundle regroupe les entités liées à la gestion RH et associative, c'est-à-dire :

- Les adhésions consultant/administrateur et les événements de recrutement
- Les informations relatives  l'établissement de la J.E. (spécialité, année d'étude)
- La gestion des équipes

**Liste des entités**

- AdministratorMembership
- ConsultantMembership
- RecruitmentEvent
- Department
- SchoolYear
- Team
- TeamMember

## b. UML


RHBundle - Entités
ConsultantMembership
- startDate: DateTime
- formFilled: boolean
- feePaid: boolean
- ribGiven: boolean
- idGiven: boolean
AdministratorMembership
- startDate: DateTime
- endDate: DateTime
- formFilled: boolean
- feePaid: boolean
user
1
1
user
consultantMembership
1
0..*
administratorMemberships
User
0..*
department
1
Department
- label: string
- detail: string
- enabled: boolean
user
0..*
1
0..*
0..*
recruitmentEvent
TeamMember
schoolYear
0..*
team
members
1
1
0..1
Team
- name: string
- creationDate: DateTime
0..*
leader
SchoolYear
- year: integer
RecruitmentEvent
- date: DateTime
- presence: integer
1
team
1
TeamMailingList

## c. Entités

| AdministratorMembership | | |
|---|---|---|
| Correspond à une adhésion administrateur (en tant que membre actif). Cette entité contient un attribut pour chaque élément nécessaire à l'inscription, ce qui permet d'évaluer la validité de l'adhésion. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| user | User | Utilisateur concerné par l'adhésion. |
| startDate | Date | Date de prise d'effet de l'adhésion. |
| endDate | Date | Date de fin de validité de l'adhésion. |
| feePaid | boolean | Indique si la cotisation a été payée. |
| formFilled | boolean | Indique si le formulaire d'inscription a été rempli. |

| ConsultantMembership | | |
|---|---|---|
| Correspond à une adhésion consultant (en tant que réalisateur d'études). Cette entité contient un attribut pour chaque élément nécessaire à l'inscription, ce qui permet d'évaluer la validité de l'adhésion. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| user | User | Utilisateur concerné par l'adhésion. |
| startDate | Date | Date de prise d'effet de l'adhésion. |
| socialNumber | string | Numéro de sécurité sociale de l'utilisateur associé. |
| feePaid | boolean | Indique si la cotisation a été payée. |
| formFilled | boolean | Indique si le formulaire d'inscription a été rempli. |
| ribGiven | boolean | Indique si un relevé d'identité bancaire a été donné. |
| idGiven | boolean | Indique si une pièce d'identité a été donnée. |

| RecruitmentEvent | | |
|---|---|---|
| Correspond à une session de recrutement. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| date | Date | Date de l'énénement associé. |
| presence | integer | Nombre de personnes présentes à la présentation. |

| Department | | |
|---|---|---|
| Correspond à un département ou une spécialité de l'école. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| label | string | Nom abrégé du département. |
| detail | string | Nom complet du département. |
| enabled | boolean | Indique si le département est actuellement en utilisable dans la J.E. Si cet attribut vaut false, aucune autre entité ayant une association avec un Department ne peut être associée à l'entité. |

| SchoolYear | | |
|---|---|---|
| Correspond à une année d'étude dans l'école. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| year | integer | Numéro de l'année. |

| Team | | |
|---|---|---|
| Correspond à une équipe de membres de la J.E., par exemple une équipe de projet. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| name | string | Nom de l'équipe. |
| creationDate | DateTime | Date de création de l'équipe. |
| leader | User | Chef de l'équipe. |
| division | Division | Pôle de la J.E. associé à l'équipe. |
| members | TeamMember[] | Membres de l'équipe (incluant le chef d'équipe). |
| *Listener* | | |
| prePersist() | A la création d'une équipe, la date de création est automatiquement associée. | |
| postPersist() | Après la création d'une équipe, on ajoute le chef d'équipe en tant que membre. | |
| postUpdate() | Après la mise à jour d'une équipe, si le chef d'équipe n'est pas déjà membre, il est ajouté en tant que membre. | |

| TeamMember | | |
|---|---|---|
| Correspond à une association entre un utilisateur membre et une équipe. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| user | User | Utilisateur membre. |
| team | Team | Equipe concernée. |

# 3. GRCBundle

## a. Contenu général

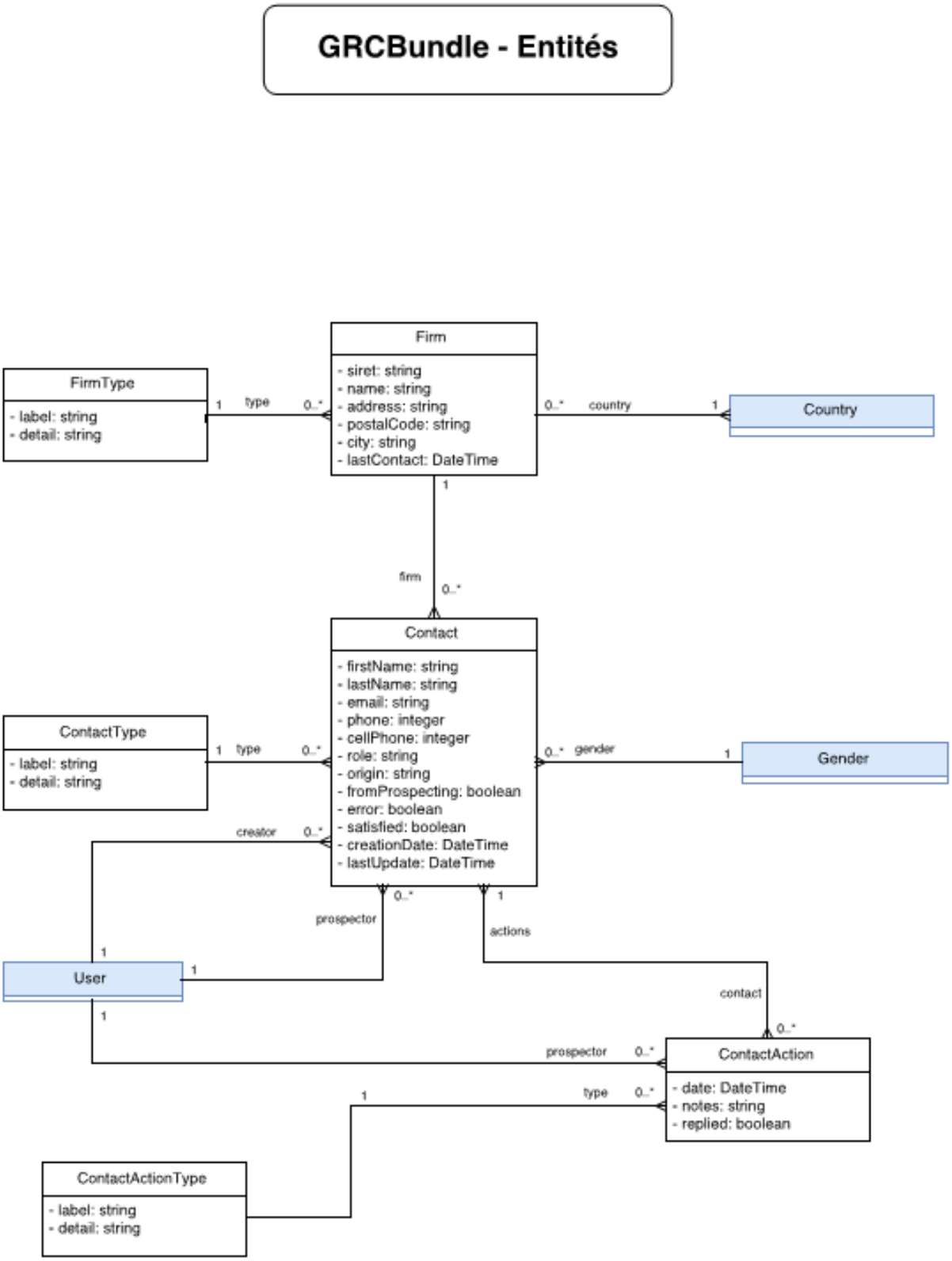
Ce bundle regroupe les entités liées à la gestion des sociétés et contacts entreprise de la J.E., soit les entités correspondant aux éléments suivants :

- Contacts et leurs catégories
- Sociétés et leurs catégories (ceci s'étend aux particuliers ou administrations par exemple)
- Les actions de prospections et leurs catégories

**Liste des entités**

- FirmType
- Firm
- ContactType
- Contact
- ContactActionType
- ContactAction

## b. UML


GRCBundle - Entités
Firm
- siret: string
- name: string
- address: string
- postalCode: string
- city: string
- lastContact: DateTime
FirmType
- label: string
- detail: string
type
country
Country
firm
Contact
- firstName: string
- lastName: string
- email: string
- phone: integer
- cellPhone: integer
- role: string
- origin: string
- fromProspecting: boolean
- error: boolean
- satisfied: boolean
- creationDate: DateTime
- lastUpdate: DateTime
ContactType
- label: string
- detail: string
gender
Gender
creator
prospector
actions
User
contact
ContactAction
- date: DateTime
- notes: string
- replied: boolean
ContactActionType
- label: string
- detail: string

## c. Entités

| FirmType | | |
|---|---|---|
| Correspond à une catégorie de société, incluant également les clients n'ayant pas ce statut. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| label | string | Nom abrégé du type de société. |
| detail | string | Nom complet du type de société |

| Firm | | |
|---|---|---|
| Correspond à une société. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| siret | string | Numéro SIRET de la société s'il existe. |
| name | string | Nom de la société. |
| address | string | Corps de l'adresse de la société. |
| postalCode | string | Code postal de la société. |
| city | string | Ville de la société. |
| lastContact | Date | Date du dernier contact avec la société. |
| type | FirmType | Type de société. |
| country | Country | Pays de la société. |

| ContactType | | |
|---|---|---|
| Correspond à un type de contact. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| label | string | Nom abrégé du type de contact. |
| detail | string | Nom complet du type de contact. |

| Contact | | |
|---|---|---|
| Correspond à un contact (personne réelle) dans une société. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| firstName | string | Prénom du contact. |
| lastName | string | Nom de famille du contact. |
| email | string | Adresse électronique du contact. |
| phone | integer | Numéro de téléphone fixe du contact. |
| cellPhone | integer | Numéro de téléphone mobile du contact. |
| role | string | Rôle du contact au sein de sa structure. |
| origin | string | Origine des coordonnées du contact. |
| fromProspecting | boolean | Indique si le contact a été obtenu via la prospection. |
| error | boolean | Indique s'il y a une erreur dans les coordonnées renseignées sur Doletic. |
| satisfied | boolean | Indique s'il serait judicieux de recontacter le contact. |
| notes | string | Notes diverses sur le contact. |
| creationDate | DateTime | Date de création du contact. |
| lastUpdate | DateTime | Date de dernière modification. |
| type | ContactType | Type du contact. |
| gender | Gender | Civilité du contact. |
| firm | Firm | Société associée au contact. |
| prospector | User | Prospecteur associé au contact. |
| creator | User | Créateur du contact dans Doletic. |
| actions | ContactAction[] | Actions de prospection entreprises avec ce contact. |
| *Listener* | | |
| prePersist() | A la création d'un contact :<br>- La date de création est automatiquement associée.<br>- Le créateur est automatiquement associé<br>- La date de dernière modification est identique à celle de création | |
| preUpdate() | A la mise à jour d'un contact, la date de dernière modification est associée. | |

| ContactActionType | | |
|---|---|---|
| Correspond à un type d'action entreprise pour un contact. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| label | string | Nom abrégé du type d'action. |
| detail | string | Nom complet du type d'action. |

| ContactAction | | |
|---|---|---|
| Correspond une action de prospection (prise de contact). | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| date | Date | Date de la prise de contact. |
| notes | string | Notes diverses sur la prise de contact. |
| replied | boolean | Indique si le contact a répondu. |
| type | ContactActionType | Type de prise de contact. |
| contact | Contact | Contact associé. |
| prospector | User | Utilisateur ayant effectué la prise de contact. |

# 4. SupportBundle

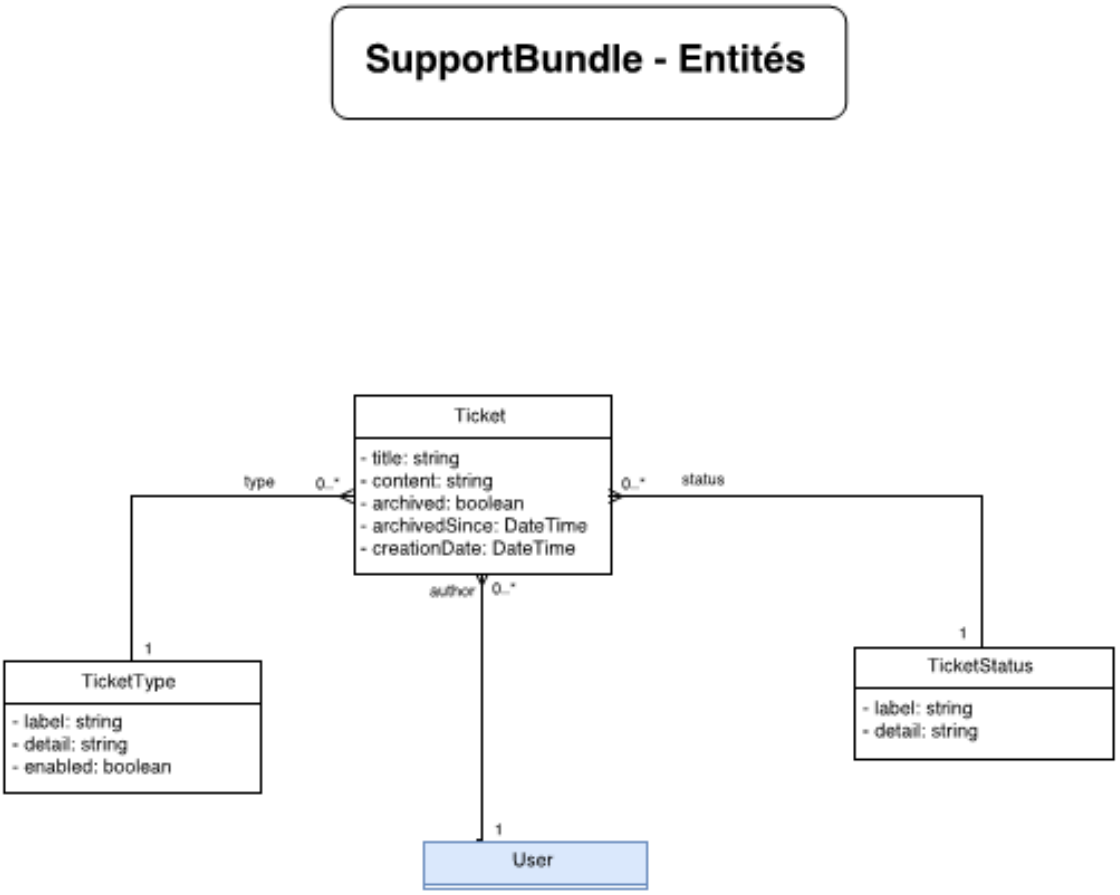
## a. Contenu général

Ce Bundle regroupe les éléments composant le système de support en ligne de Doletic et la gestion des tickets.

**Liste des entités**

- TicketType
- TicketStatus
- Ticket

## b. UML


SupportBundle - Entités
Ticket
- title: string
- content: string
- archived: boolean
- archivedSince: DateTime
- creationDate: DateTime
type
0..*
0..*
status
author
0..*
1
1
1
TicketType
- label: string
- detail: string
- enabled: boolean
TicketStatus
- label: string
- detail: string
User

## c. Entités

| TicketType | | |
|---|---|---|
| Correspond à un type de ticket, soit un sujet général. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| label | string | Nom abrégé du type de ticket. |
| detail | string | Nom complet du type de ticket. |
| enabled | boolean | Indique si le type est actuellement en utilisable dans la J.E. Si cet attribut vaut false, aucune autre entité ayant une association avec un TicketType ne peut être associée à l'entité. |

| TicketStatus | | |
|---|---|---|
| Correspond à un statut de ticket, soit un état d'avancement de son traitement. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| label | string | Nom abrégé du statut de ticket. |
| detail | string | Nom complet du statut de ticket. |

| Ticket | | |
|---|---|---|
| Correspond à un ticket posté par un utilisateur. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| title | string | Titre du ticket. |
| content | string | Contenu du ticket. |
| type | TicketType | Type de ticket. |
| status | TicketStatus | Statut actuel du ticket. |
| author | User | Auteur du ticket. |
| archived | boolean | Indique si le ticket est archivé. |
| archivedSince | DateTime | Date d'archivage du ticket s'il est archivé. |
| creationDate | DateTime | Date de création du ticket. |
| *Listener* | | |
| prePersist() | A la création du ticket, son auteur et sa date de création sont automatiquement associés. | |

# 5. UABundle

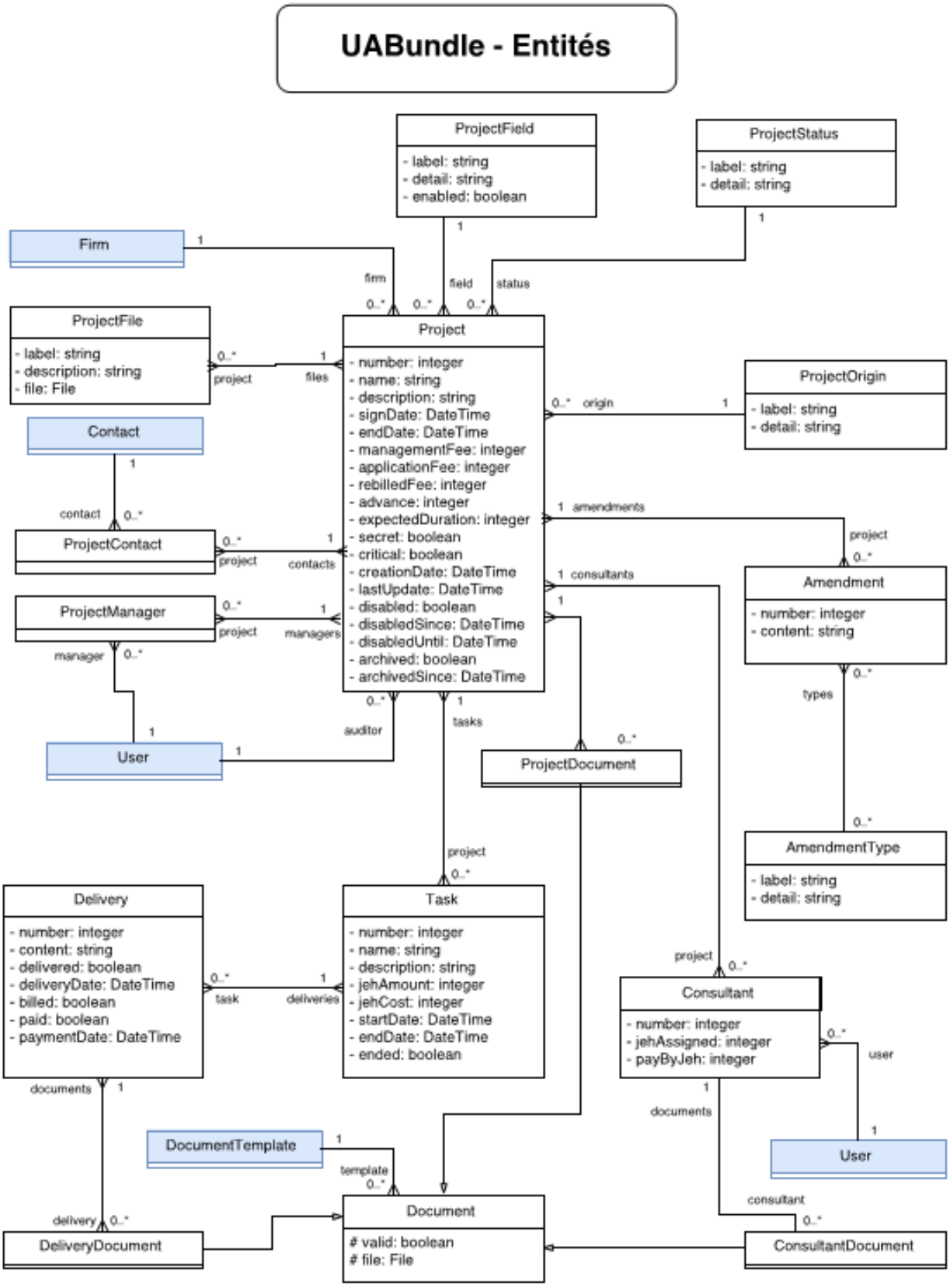
## a. Contenu général

Ce Bundle regroupe tout ce qui est en lien avec la gestion des études, incluant la gestion des documents d'étude.

**Liste des entités**

- Amendment
- AmendmentType
- ProjectField
- ProjectStatus
- ProjectOrigin
- Task
- Project
- Consultant
- Document
- Delivery
- ConsultantDocument
- DeliveryDocument
- ProjectDocument
- ProjectFile

## b. UML


UABundle - Entités
ProjectField
- label: string
- detail: string
- enabled: boolean
ProjectStatus
- label: string
- detail: string
Firm
ProjectFile
- label: string
- description: string
- file: File
Project
- number: integer
- name: string
- description: string
- signDate: DateTime
- endDate: DateTime
- managementFee: integer
- applicationFee: integer
- rebilledFee: integer
- advance: integer
- expectedDuration: integer
- secret: boolean
- critical: boolean
- creationDate: DateTime
- lastUpdate: DateTime
- disabled: boolean
- disabledSince: DateTime
- disabledUntil: DateTime
- archived: boolean
- archivedSince: DateTime
ProjectOrigin
- label: string
- detail: string
Contact
ProjectContact
ProjectManager
User
Amendment
- number: integer
- content: string
AmendmentType
- label: string
- detail: string
ProjectDocument
Delivery
- number: integer
- content: string
- delivered: boolean
- deliveryDate: DateTime
- billed: boolean
- paid: boolean
- paymentDate: DateTime
Task
- number: integer
- name: string
- description: string
- jehAmount: integer
- jehCost: integer
- startDate: DateTime
- endDate: DateTime
- ended: boolean
Consultant
- number: integer
- jehAssigned: integer
- payByJeh: integer
DocumentTemplate
Document
# valid: boolean
# file: File
DeliveryDocument
ConsultantDocument
firm
field
status
project
files
origin
contact
contacts
managers
manager
auditor
tasks
amendments
consultants
types
task
deliveries
documents
template
delivery
user
consultant

## c. Entités

| Amendment | | |
|---|---|---|
| Correspond à un avenant sur une étude. Cette entité existe à titre indicatif mais n'est pas utilisé pour le publipostage des avenants. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| project | Project | Projet sur lequel l'avenant a été édité |
| content | string | Contenu de l'avenant |
| attributable | boolean | Indique si l'avenant est imputable à la J.E. |
| date | DateTime | Date de l'avenant. |
| types | AmendmentType[] | Types de l'avenant. Un avenant peut porter sur différents points de la convention en même temps. |

| AmendmentType | | |
|---|---|---|
| Correspond à un type d'avenant (points de la convention sur lesquels porte l'avenant). | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| label | string | Nom abrégé du type. |
| detail | string | Nom complet du type. |

| ProjectField | | |
|---|---|---|
| Correspond à un domaine de compétence de la J.E., duquel dépend un projet. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| label | string | Nom abrégé du domaine de compétence. |
| detail | string | Nom complet du domaine de compétence. |
| enabled | boolean | Indique si le domaine de compétence est actuellement en fonctionnement dans la J.E. Si cet attribut vaut false, aucune autre entité ayant une association avec une Division ne peut être associée à l'entité. |

| ProjectStatus | | |
|---|---|---|
| Correspond au statut (état d'avancement) d'un projet. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| label | string | Nom abrégé du statut. |
| detail | string | Nom complet du statut. |

| ProjectOrigin | | |
|---|---|---|
| Correspond à l'origine d'une sollicitation, donc au moyen de contact. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| label | string | Nom abrégé de l'origine. |
| detail | string | Nom complet de l'origine. |

| Task | | |
|---|---|---|
| Correspond à une phase d'un projet. Les phases sont dans un certain ordre et peuvent être réoranisées. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| number | integer | Numéro de la phase. |
| project | Project | Etude dont dépend la phase. |
| name | string | Nom de la phase. |
| description | string | Description plus complète de la phase. |
| jehAmount | integer | Nombre de JEH associés à la phase. |
| jehCost | integer | Prix unitaire du JEH pour cette phase. |
| startDate | Date | Date théorique de début de la phase. |
| endDate | Date | Date théorique de fin de la phase. |
| ended | boolean | Indique si la phase a été réalisée. |
| deliveries | Delivery[] | Livrables associés à la phase |
| *Listener* | | |
| prePersist | A la création de la phase, un numéro lui est automatiquement attribué en fonction des phases existantes. S'il n'y a pas d'autres phases pour ce projet, le numéro est 1. Sinon, la phase succède à celle au numéro le plus élevé. | |
| preRemove | A la suppression de la phase, les numéros des autres phases sont réajustés pour ne pas créer de vide. Par exemple, si on supprime la phase 2, la phase 3 devient 2, 4 devient 3, et ainsi de suite. | |

| Project | | |
|---|---|---|
| Correspond à une étude (ou projet) de la J.E., avec toutes ses informations. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| number | integer | Numéro de l'étude. |
| name | string | Nom de l'étude. |
| description | string | Description plus complète de l'étude. |
| signDate | Date | Date de signature si l'étude est signée. |
| endDate | Date | Date de clôture si l'étude est clôturée. |
| managementFee | integer | Frais de gestion de l'étude. |
| applicationFee | integer | Frais de dossier de l'étude. |
| rebilledFee | integer | Frais refacturés au client de l'étude. |
| advance | integer | Acompte de l'étude. |
| expectedDuration | integer | Durée prévue (en semaine) de l'étude. |
| secret | boolean | Indique si l'étude est confidentielle. |
| critical | boolean | Indique si l'étude est dans un état critique, c'est-à-dire si elle demande une attention particulière. |
| creationDate | DateTime | Date de création de l'étude dans Doletic. |
| lastUpdate | DateTime | Date de dernière modification dans Doletic. |
| disabled | boolean | Indique si l'étude est désactivée, c'est-à-dire en attente. |
| disabledSince | DateTime | Date de désactivation de l'étude. |
| disabledUntil | DateTime | Date de réactivation prévue de l'étude. L'étude est réactivée automatiquement à cette date. |
| archived | boolean | Indique si l'étude est archivée. |
| archivedSince | DateTime | Date d'archivage de l'étude si elle est archivée. |
| firm | Firm | Socété cliente de l'étude. |
| auditor | User | Correspondant qualité de l'étude. |
| managers | ProjectManager[] | Chargés d'affaires de l'étude. |
| contacts | ProjectContact[] | Contacts client de l'étude. |
| consultants | Consultant[] | Consultants de l'étude. |
| field | ProjectField | Domaine de compétence principal de l'étude. |
| origin | ProjectOrigin | Origine de la sollicitation initiale. |
| status | ProjectStatus | Statut actuel de l'étude. |
| tasks | Task[] | Phases de l'étude. |
| amendments | Amendment[] | Avenants de l'étude. |

| | | |
|---|---|---|
| documents | ProjectDocument[] | Documents liés à l'étude (publipostés). |
| files | ProjectFile[] | Fichiers divers en rapport avec l'étude. |
| *Listener* | | |
| prePersist | A la création de l'étude, différentes valeurs sont attribuées :<br>- La date de création<br>- La date de dernière mise à jour (identique à la date de création)<br>- Le numéro d'étude, qui est récupéré à partir du dernier numéro d'étude attribué. | |
| preUpdate | A la mise à jour de l'étude, la date de dernière mise à jour est attribuée. | |

| **Consultant** | | |
|---|---|---|
| Correspond au lien entre un membre consultant et une étude. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| project | Project | Etude sur laquelle travaille le consultant. |
| user | User | Utilisateur consultant sur l'étude. |
| number | integer | Numéro du consultant. Ce numéro apparaît sur les RM. |
| jehAssigned | integer | Nombre de JEH assignés au consultant. |
| payByJeh | integer | Rémunération du consultant par JEH. |
| documents | ConsultantDocument[] | Documents d'étude reliés au consultant. |
| *Listener* | | |
| prePersist | A la création du consultant, son numéro est attribué. | |
| preRemove | A la suppression du consultant, les numéros des autres consultants sont réajouster pour ne pas qu'il y ait de vide. | |

| **Document** | | |
|---|---|---|
| Correspond à un document d'étude. Cette entité est abstraite, car les documents sont différenciés selon qu'ils sont reliés à l'étude elle-même, à un consultant ou à un livrable de l'étude. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| template | DocumentTemplate | Modèle associé au document. |
| auditor | User | Utilisateur ayant validé le document. |
| valid | boolean | Indique si le document a été validé. |
| file | string | Fichier uploadé pour ce document. |

| Delivery | | |
|---|---|---|
| Correspond à un livrable d'étude (intermédiaire ou final). | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| task | Task | Phase de l'étude à l'issue de laquelle le livrable est terminé. |
| number | integer | Numéro du livrable (utilisé dans sa référence). |
| content | string | Résumé du contenu du livrable. |
| delivered | boolean | Indique si le livrable a été remis au client. |
| deliveryDate | Date | Date de remise du livrable au client le cas échéant. |
| billed | boolean | Indique si le livrable est facturé au client. |
| paid | boolean | Indique si le livrable a été payé (s'il a été facturé). |
| paymentDate | Date | Date de paiement du livrable par le client. |
| documents | DeliveryDocument[] | Documents d'étude reliés au livrable. |

| ProjectDocument | | |
|---|---|---|
| Hérite de la classe Document. Correspond à un document relié directement à l'étude. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| project | Project | Etude liée au document. |

| DeliveryDocument | | |
|---|---|---|
| Hérite de la classe Document. Correspond à un document relié à un livrable d'une étude. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| delivery | Delivery | Livrable lié au document. |

| ConsultantDocument | | |
|---|---|---|
| Hérite de la classe Document. Correspond à un document relié à un consultant. | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| consultant | Consultant | Consultant lié au document. |

| ProjectFile | | |
|---|---|---|
| Correspond à un fichier associé à une étude. Il n'y a pas de restriction précise sur le fichier (il ne s'agit pas d'un document d'étude). | | |
| *Attributs* | | |
| **Nom** | **Type/Retour** | **Description** |
| project | Project | Etude liée au fichier. |
| label | string | Nom du fichier |
| description | string | Description détaillée du fichier |
| file | string | Chemin vers le fichier associé |

***Durant toute cette partie, les lignes colorées en bleu correspondent aux attributs uniques. Lorsque plusieurs attributs sont uniques ensemble, ils sont colorés en bleus et soulignés par une bordure bleue plus foncé.***

# Gestion des droits et routes

## 1. Fonctionnement général des routes

Les routes sont réparties par Bundle. La structure générale d'une route est toujours la même :

/api/[bundle]/[action]

Par exemple, pour afficher la liste des utilisateurs, la route GET est :

/api/kernel/users

Le tableau suivant récapitule la convention de nommage des routes les plus classiques, pour un bundle « Bundle » et une entité « Object » :

| **Utilité** | **Méthode** | **Format** |
|---|---|---|
| Trouver un objet par id | GET | /api/bundle/object/{id} *(contrainte sur id)* |
| Trouver tous les objets | GET | /api/bundle/objects |
| Trouver un objet par un attribut unique | GET | /api/bundle/object/{attribut} *(pas de contrainte)* |
| Trouver des objets par un attribut non unique | GET | /api/bundle/objects/attribut/{attribut} |
| Trouver les objets de l'utilisateur courant | GET | /api/bundle/objects/current |
| Créer un nouvel objet | POST | /api/bundle/object |
| Modifier un objet existant | POST* | /api/bundle/object/{id} |
| Supprimer un objet existant | DELETE | /api/bundle/object/{id} |

**La méthode appropriée serait plutôt PUT ou PATCH, mais à la rédaction de cette documentation, des problèmes techniques liés à Symfony empêchent de respecter la convention.*

## 2. Niveaux de droits

Le système de droit est globalement toujours le même, et réparti par Bundle. Une route spécifique permet d'obtenir le niveau de droit de l'utilisateur courant dans le bundle concerné. Les niveaux de droits sont les suivants :

- Super-admin : tous les droits sur le Bundle
- Admin : La plupart des droits, sauf les actions dangereuses (suppressions avec dépendances) et rares (ajout d'un type de société par exemple)
- User : La plupart des droits de lecture, sauf données sensibles
- Guest : Droits de lecture limités aux données très génériques
- Aucun droit

Ces droits sont associés à un indice chiffré de 0 (aucun droit) à 4 (Super-admin) qui est récupéré par appel à la route spécifique du Bundle. Cette route suit le format suivant :

/api/[bundle]/rights

Elle est présente dans un contrôleur spécifique. Chaque bundle possède ce contrôleur, qui est nommé, pour un bundle « Exemple », « ExempleController ».

# 3. Comment ajouter/modifier les rôles

Comme dans tout projet Symfony, les rôles sont présents dans le fichier security.yml. D'une manière générale, il est recommandé de conserver le fonctionnement déjà adopté : quatre rôles distincts par bundle. Si un bundle demande une séparation plus fine des rôles, il faut évidemment la mettre en place.

A chaque poste correspond une liste de rôles, un par Bundle (il est inutile d'en avoir plus d'un pour le même bundle, puisque un rôle englobe également tous les rôles inférieurs).

# Lien avec l'API OVH

## 1. Présentation de l'API

### a. Intérêt

L'API OVH permet d'automatiser les interactions avec les différents services fournis par OVH. Dans la pratique, cette API est principalement utilisée pour gérer les comptes mails et les listes de diffusion. Son fonctionnement est similaire à celui de Doletic, puisque basé sur les mêmes principes généraux.

### b. Liens utiles

La documentation officielle de l'API est disponible à cette adresse : https://api.ovh.com/console/

## 2. Services reliés à l'API

Dans Doletic, il existe un service du bundle KernelBundle, « OVHMailManager », qui permet d'effectuer des interactions avec l'API OVH. L'organisation de ce service est un peu particulière.

### a. Constantes

Il contient de nombreuses constantes.

- Celles dont le nom commence par FUNC_ correspondent aux noms des routes.
- Celles dont le nom commence par ARG_ correspondent aux paramètres GET
- Celles dont le nom commence par PARAM_ correspondent aux paramètres POST
- Celles dont le nom commence par OPTION_ correspondent aux options facultatives disponibles sur certaines routes
- Celles dont le nom commence par P_ correspondent à d'autres paramètres.

Les tableaux constants suivants contiennent les routes correspondant à chaque fonction. Il y a un tableau par méthode http.

### b. Paramètres en entrée

Le service prend en paramètres plusieurs valeurs. Ces valeurs permettent d'identifier le compte utilisateur OVH que possède ETIC et sont présentes dans le fichier parameters.yml (cf partie Configuration). Elles sont disponibles dans le Manager OVH.

### c. Appels à l'API

Il existe une méthode générique d'appel pour chaque méthode http : les méthodes get, post, delete et put. Ces méthodes contrôle la validité de la requête et l'envoie si elle est correcte. Les paramètres GET sont

automatiquement trouvés dans l'URL et remplacés par les valeurs fournies. Si une valeur manque pour un paramètre GET présent dans l'URL, la requête n'est pas envoyée.

Les appels utilisent l'objet Api, qui vient d'un projet visant à faciliter les appels à l'API. C'est une dépendance de Doletic, le bundle ZoddoOVHBundle.

# Publipostage

## 1. Présentation et utilisation de PHPWord

PHPOffice/PHPWord est une bibliothèque PHP permettant d'éditer des documents Word en PHP, ainsi que d'utiliser des templates Word pour remplacer des variables par des valeurs calculées ou stockées. Son utilisation dans Doletic se résume pour le moment à cette dernière fonctionnalité.

Dans Doletic, PHPWord est utilisé dans le service « DocumentPublisher » du bundle KernelBundle. Ce service contient une méthode « publishFromTemplate » dont le fonctionnement est simple :

- A partir d'une entité DocumentTemplate fournie en entrée, un objet « TemplateProcessor » (objet de PHPWord) est instancié.
- A partir d'un dictionnaire fourni en entrée (association clé/valeur), les variables sont remplacées.
- Le fichier obtenu est sauvegardé. La destination du fichier est calculée comme suit :
    - Le dossier de destination est un paramètre de parameters.yml (cf partie Configuration).
    - Un préfixe, correspondant dans le cas d'un document d'étude au numéro d'étude, est ajouté.
    - La fin du nom correspond au label du DocumentTemplate fourni en entrée.
- Le fichier créé est renvoyé à l'utilisateur sous forme de réponse http.

L'utilisation de PHPWord est très simple, et cette partie ne devrait pas changer. **Les variables dans les documents sont toujours suivant le format ${VARIABLE}**. Cette contrainte est imposée par PHPWord.

## 2. Remplissage des dictionnaires

Comme vu dans la description du modèle, il existe différents documents. Actuellement, seul le bundle UABundle utilise des documents liés à des templates. Chaque type de document a sa propre entité qui le relie à une autre entité dont il fait l'objet. Les documents actuellement existant dépendent d'une étude (« Project »), d'un consultant (« Consultant ») ou d'un livrable (« Delivery »).

Un service « DocumentService » du bundle UABundle permet de remplir les dictionnaires. Chaque type de document a sa méthode associée, qui remplit les valeurs nécessaires. A noter que toutes ces valeurs ne seront pas forcément utilisées dans la pratique, mais cela ne perturbe pas le fonctionnement de PHPWord.

Le dictionnaire est un simple tableau associatif. Les clés sont les noms des variables **sans les ${ et }** et les valeurs sont ce qui doit remplacer les variables.

A noter que certaines fois, des calculs ou traitement sont nécessaires. Par exemple, il peut être nécessaire de mettre en majuscules, supprimer les accents, effectuer des opérations mathématiques…

## 3. Gérer les documents types

La gestion des documents types n'est pas directement du ressort de la DSI, mais doit être faite en collaboration avec les responsables Performance et UA. Lorsque de nouveaux documents types sont créés ou que certains sont modifiés, il faut les mettre en ligne et les tester.

# Upload de fichiers

## 1. Service et Listener

L'upload de fichier est géré par Symfony et est relativement simple d'utilisation. Il utilise deux éléments.

Le service « FileUploader » du KernelBundle permet de hasher le nom du fichier puis de le placer dans son dossier de destination. Il est appelé chaque fois qu'un upload de fichier a lieu, et renvoie le chemin d'accès au fichier sauvegardé.

Le Listener « DocumentUploadListener » du bundle UABundle assure la sauvegarde des fichiers des entités Document. En effet, chaque fois qu'un fichier est stocké, on ne garde que sa destination en base de données.

Lorsque le fichier est ajouté ou modifié, on vérifie qu'il s'agit bien d'un fichier, puis le service FileUploader est appelé pour le sauvegarder et récupérer son chemin d'accès. Ce chemin est ensuite conservé en base de données dans l'attribut correspondant au fichier.

## 2. Note sur l'upload de fichiers

L'upload de fichier est une source de danger pour l'application. Il faut donc contrôler le type de fichier uploadé par l'utilisateur, afin de n'accepter que les types pertinents (et surtout de refuser les scripts ou autre fichier potentiellement dangereux !).

Ce contrôle se fait dans l'entité contenant le fichier, grâce à l'annotation suivante :

```
@Assert\File(mimeTypes={ "application/pdf" }, maxSize=4096)
```

Dans cet exemple, seuls les fichiers de type PDF sont acceptés. La liste des types possibles est disponible à cette adresse : http://www.iana.org/assignments/media-types/media-types.xhtml

Le paramètre maxSize permet de limiter la taille des fichiers. Les formats de taille et les autres paramètres disponibles sont présents dans la documentation Symfony.

# Configuration

## 1. Configuration générale

La configuration générale se fait, comme dans tout projet Symfony, dans le fichier /app/config/config.yml du dossier. Ce fichier contient la configuration de base de Symfony, ainsi que celle des bundles installés. Normalement, la configuration existante ne devrait pas être modifiée. Si de nouveaux bundles sont ajoutés, il faudra bien sûr rajouter leur configuration à ce fichier.

## 2. Paramètres

### a. Que sont les paramètres ?

Le fichier /app/config/parameters.yml contient les paramètres, susceptibles d'être modifiés en cas d'évolution du SI. Il faut bien faire la différence entre :

- Les **paramètres d'application**, rarement modifiés, uniquement accessibles à l'administrateur de l'API (Responsable DSI). Par exemple : les identifiants permettant d'utiliser l'API OVH.
- Les **paramètres de fonctionnement** de la J.E., qui peuvent être sujets à des modifications plus fréquentes, et pas forcément par l'administrateur de l'API, mais par un utilisateur ayant des droits suffisants. Par exemple : le taux de TVA en vigueur.

Les premiers ont leur place dans le fichier parameters.yml, les seconds doivent plutôt être conservés sous forme d'instances de l'entité Setting, afin d'être accessible une interface reliée à l'API. Cette partie traitera par la suite uniquement des paramètres d'application.

### b. Paramètres et confidentialité

Au sujet de la confidentialité, **il est impératif de protéger certains paramètres**. Les identifiants utilisés pour se connecter à un service sont par exemple secrets.

Symfony et composer utilisent le fichier parameters.yml.dist pour générer le fichier parameters.yml à l'installation de l'application. Tous les paramètres existants doivent y figurer pour savoir dont l'application a besoin. Ce fichier permet d'entrer des valeurs par défaut et est présent sur le dépôt git du projet. Il ne faut donc pas mettre de données confidentielles dans ce fichier, mais uniquement des valeurs génériques, voire rien du tout si aucune valeur par défaut sécurisée ne peut être trouvée.

### c. Quand utiliser des paramètres ?

Le fichier parameters.yml permet de centraliser toutes les valeurs susceptibles de varier entre les différentes installations de l'API. Chaque fois qu'une valeur de configuration est entrée quelque part, il faut se poser la question : est-ce qu'une autre personne utilisant l'API est susceptible de la changer sans que cela n'affecte le bon fonctionnement de l'API ? Si c'est le cas, la valeur ne devrait pas être entrée en dur, mais être dans un paramètre. Pour rappel, dans les configurations YAML de Symfony, on fait référence à un paramètre en l'entourant de symboles %.

Un autre cas d'utilisation des paramètres est celui de valeurs fixes, qu'il faudra récupérer depuis le code PHP. Attention dans ce cas à bien se demander s'il ne vaudrait pas mieux créer une instance de Setting.

# 3. Sécurité

Le fichier /app/config/security.yml permet de gérer les rôles et firewalls. Les rôles ont déjà été évoqués plus haut. Les firewalls permettent de restreindre l'accès à toute une catégorie de routes. Dans l'API Doletic, chaque bundle a pour convention un firewall qui empêche d'accéder à ses routes à moins de posséder au moins le rôle « Guest » correspondant.

Le niveau de droit des routes individuelles est directement dans le code. Certaines routes ont des vérifications simples, qui utilisent la fonction « denyAccessUnlessGranted », dont le nom est assez clair : si l'utilisateur n'a pas le rôle indiqué, l'accès est refusé. D'autres routes utilisent des services plus complexes pour déterminer l'autorisation : c'est le cas par exemple de certaines routes concernant l'entité Project, qui vérifient si l'utilisateur est chargé d'affaires sur le projet *ou* s'il possède suffisamment de droits pour gérer tous les projets.

Le fichier contient également le nom de l'algorithme de chiffrement des mots de passe utilisés par FOSUserBundle. L'algorithme actuellement utilisé est bcrypt avec sel (le sel est une chaine de caractères aléatoire et propre à chaque utilisateur qui est ajoutée au mot de passe avant sont chiffrage). Au moment de la rédaction de cette documentation, cet algorithme est considéré comme sûr. Il faut cependant penser à se renseigner de temps en temps, pour voir si c'est toujours le cas. En effet, si la complexité des algorithmes de chiffrage est trop élevée pour retrouver le mot de passe original à partir du hash, certains sites se basent sur de grandes bases de données pour retrouver la chaîne originale. C'est le cas pour les algorithmes sha1 et md5 par exemple.

# 4. Routes

Les préfixes des routes (par Bundle) se trouvent dans le fichier /app/config/routing.yml. A noter qu'elles doivent toujours avoir le même format de préfixe : /api/[bundle], et le même type : « rest ».

Les routes individuelles se trouvent directement dans les contrôleurs, sous forme d'annotation. N'oublie pas qu'une action d'un contrôleur doit toujours être de la forme « descriptionAction ». On utilise ensuite les annotations du bundle FOSRestBundle : @Get, @Post, @Delete… pour définir la route.

# Fonctionnalités en projet

## 1. Gestion simple des indicateurs

Les indicateurs sont cruciaux pour suivre l'avancement des objectifs de la J.E., et doivent être visibles sur Doletic. Obtenir la valeur d'un indicateur revient en fait à effectuer une requête sur la base de données, requête impliquant souvent des opérations complexes telles que des jointures, unions, opérations récursives… Une telle requête est assez longue à écrire et tester même pour quelqu'un d'assez expérimenté en SQL. Il ne faut donc pas attendre de tous les membres d'ETIC qu'ils soient capables d'écrire la requête correspondant à leur indicateur.

Pour remédier à ce problème, il faudrait créer un système permettant, via des formulaires, de créer et modifier des indicateurs. Il s'agit en fait d'un générateur de requête, qui permettrait via une interface simple de générer une requête qui pourra être stockée et exécutée sur demande.

D'autres éléments pourraient être utiles :

- Classer les indicateurs par catégorie, par exemple par Bundle, pour qu'on puisse côté front-end récupérer uniquement les indicateurs correspondant à une activité et tous les afficher.
- Activer/désactiver des indicateurs. Plutôt que de supprimer un indicateur qui ne sera plus suivi au nouveau mandat, il peut être judicieux de simplement le « mettre de côté » pour une éventuelle réutilisation future.

## 2. Base de données de consultants potentiels

Afin de trouver plus facilement de potentiels consultant lorsqu'une sollicitation arrive à la J.E., il pourrait être intéressant de tenir une base de données contenant des informations très basiques sur des étudiants intéressés. Cette base pourrait par exemple contenir simplement : Nom, Prénom, Département de l'école, adresse électronique de l'école.

Ces consultants pourraient ensuite être « transformés » facilement en utilisateurs Doletic à part entière s'ils deviennent consultants pour la J.E, en ajoutant les informations manquantes. Ceci faciliterait le travail de la gestion RH.