

Développer une application web “Back-office” de la plateforme DataXchange

Remerciements

Je tiens à remercier dans un premier temps le Président de la société Zento et mon maître de stage, Yoann Amsellem pour la confiance qu'il m'a accordé dans la création de son back-office.

Ce stage m'a permis d'affirmer mes connaissances en conception et réalisation d'applications web. Il m'a aussi permis d'enrichir ma culture technique de part la technologie employée dans le développement de la plateforme DataXchange (JAVA).

Je tiens à remercier aussi toute l'équipe de la Wild Code School, pour son sérieux et son entrain dans la transmission de leur savoir.

Enfin je remercie mes camarades de promotion sans qui je n'aurai tant progresser et pris plaisir à apprendre à travailler en équipe.

I. Introduction	5
II. Présentation de la mission	5
A. Zento	5
B. Besoins clients (Zento)	6
1. Objectifs généraux	6
2. Technologies	6
3. Informations	6
III. La mission	7
A. Analyse de l'existant	7
1. Etude du fonctionnement du site	7
2. Les informations à traiter	7
3. Les entités	7
4. Challenges rencontrés	9
B. Approche technique et organisation projet	17
1. Méthode de travail Agile	17
2. Les différentes étapes du projet Gantt	17
3. Modélisation Merise	17
4. Gestion de versions de fichiers Git	18
5. Challenges rencontrés	20
C. Choix techniques	21
1. Back-end	21
a. Le langage de développement PHP 7	21
b. Le framework Symfony 4	21
i. Challenges rencontrés	22
c. L'ORM Doctrine	22
i. Lien à la base et création des entités	22
ii. Gestion des insertions en base de données	27
iii. Challenges rencontrés	29
d. Parser PHP	31
e. Finder Component	32
f. Déplacement du fichier après traitement	33
g. Exécution périodique du traitement des exceptions	33
h. Sécurité	36
2. Front-end	40
a. Le web mobile Bootstrap	40
b. Pré-processeur SCSS	40
3. Tests et validations	41
4. Déploiement	45
IV. Résultats	45
A. Fin du projet	45
B. Axes d'améliorations de l'existant	45

C. Développements futur	45
V. Conclusion	46
VI. Annexes	48
A. Table des illustrations	48
B. Sources des notes de bas de pages	49

I. Introduction

Une période d'immersion en entreprise est l'opportunité d'appréhender le travail au sein d'une équipe de développeurs et de comprendre les impératifs liés à ce métier. C'est de plus l'occasion d'approfondir les connaissances acquises pendant la formation et de monter en compétences sur les langages employés, voire d'en découvrir de nouveaux.

L'entreprise Zento qui m'a accueilli a compris mon besoin d'approfondir mes connaissances en PHP et Symfony. Mon maître de stage et Président de Zento, Yoann Amsellem, m'a alors proposé de réaliser le back-office de la plateforme d'achat/vente en ligne "DataXchange" en utilisant les compétences que j'ai eu à développer pendant ma formation à la Wild Code School.

La capacité d'une entreprise à capter, recenser, déterminer, stocker et corriger les retours d'erreurs est fondamentale pour maintenir une forte confiance de ses clients. Elle permet aussi d'augmenter sa clientèle, la fidéliser et d'assurer sa pérennité. C'est dans l'esprit de cette démarche de qualité que j'ai entrepris de répondre à son besoin.

La Wild Code School, centre de formation spécialisé dans les métiers du numérique, par sa formation de "Développeur Web et mobile", m'a permis de comprendre les étapes et les processus à suivre pour mener à bien la résolution d'un besoin d'application fonctionnelle. L'organisation du travail en méthode Agile (SCRUM) adaptée aux besoins de la formation, les étapes de conception à suivre et la présentation des fonctionnalités de l'application à chaque fin de sprint m'ont donné l'occasion de comprendre les attentes des entreprises et des clients.

Les retours d'erreurs par email étant déjà opérationnels, j'ai eu pour mission de réaliser le regroupement des informations délivrées en les répertoriant et en les classant par typologie. La compréhension des différentes entités a été indispensable pour organiser ces informations et mener à bien ce projet. L'entreprise prévoyant dans les 2 ans de travailler avec une équipe de Tierce Maintenance Applicative, il m'a fallu intégrer une possibilité d'évolutivité simple et rapide du back-office afin de prévoir une bonne maintenabilité de celui-ci.

Au travers ce document, je vais ainsi vous présenter les différentes étapes de la réalisation de ce projet. Tout d'abord, je présenterai l'entreprise, ainsi que l'environnement dans lequel elle évolue afin de dégager les besoins réels et les enjeux impliqués. Puis nous verrons en détails la mission qui m'a été proposée, avec une analyse des méthodes de retours d'erreurs. Et je présenterai les différents choix de solutions techniques sélectionnées pour la phase de développement. Enfin, une analyse de la réalisation permettra d'évaluer la pertinence de ses choix par rapport aux résultats obtenus.

II. Présentation de la mission

A. Zento

L'entreprise Zento a été créée en 2016 par Yoann Amsellem (Président) et Jérémy Amsellem (Directeur Général). Cette société par actions simplifiées a pour activité principale le conseil en systèmes et logiciels informatiques. Le nombre de personnes travaillant dans cette entreprise comprend le Président et le Directeur Général.

L'un des axes de développement de Zento est la Recherche & Développement, notamment au travers d'une plateforme d'achat/vente de biens dématérialisés. Ce site a pour but de permettre l'échange et la monétisation de fichiers en les rendants accessible à toutes plateformes grâce à l'analyse et la conversion des fichiers en plusieurs formats.

Au regard de la structure, mon maître de stage -M. Yoann Amsellem- aura également été le Scrum Master, Product Owner et Developpeur principal de référence durant ce stage. L'ensemble des réalisations back-office présentées dans ce rapport est le fruit de mon travail au travers du stage.

B. Besoins clients (Zento)

1. Objectifs généraux

L'objectif de réalisation du back-office est l'exhaustivité des logs remontés, la priorisation des actions aidées par l'affichage et les codes couleurs, ainsi que la lisibilité de l'intrication des informations et alertes.

La mission proposée par Zento concerne dans un premier temps, l'outillage pour la gestion de la plateforme (Zento). Dans un second temps, sa stratégie de développement à moyen terme (2 ans) : les utilisateurs seront une équipe dédiée de supervision sous-traitante pour Zento.

2. Technologies

Le site côté serveur de production délivrant les pages web et l'API se compose de différents modules encapsulés par une gestion des événements et anomalies communes et granulaires.

Le back-office de gestion des retours de logs pour les retours de logs d'erreurs est développé en PHP 7 avec le framework Symfony 4. La base de données est gérée en SQL¹ avec l'aide de l'ORM² Doctrine. Le front est géré avec Bootstrap et CSS³.

3. Informations

L'accès au back-office comportant des données sensibles, elle ne peut se faire qu'avec une connexion sécurisée accessible uniquement aux administrateurs Zento.

Le back-office n'a aucun besoin de référencement SEO⁴, car celui-ci n'étant pas exposé sur les IP publiques de la plateforme, il n'est donc pas référençable. Toutefois mon maître de stage m'a sensibilisé aux outils utilisés pour la plateforme DataXchange en production.

Le site doit être compatible avec au moins les navigateurs Chrome, Mozilla et Chrome mobile ; son hébergement sera à la charge de Zento.

¹ MySQL - PHPMyAdmin

² Object-Relational Mapping : Lié un objet à une base de données

³ Plus spécifiquement en SCSS via le préprocesseur SAAS

⁴Search Engine Optimization : recherche d'optimisation des procédures

III. La mission

Mon rôle dans cette entreprise pendant le stage a été de réaliser le site back-office afin que les retours d'erreurs, jusqu'alors générés au fil de l'eau en format ".txt" et envoyés par email, soient analysés classés et restitués au travers de tableaux de bord sur une plateforme web accessible sur le LAN (Local Area Network) de l'entreprise. Cette plateforme devant permettre de filtrer les types d'erreurs (utilisateurs, base de données, réseau...) selon différentes informations (type, date, criticité...).

A. Analyse de l'existant

Pour réaliser sereinement une application web, il est important de bien comprendre le fonctionnement de l'existant. C'est pourquoi une analyse approfondie de ce qui est déjà créé doit être effectuée.

1. Etude du fonctionnement du site

DataXchange est un site développé avec la technologie JAVA, il est déployé selon une architecture 3-TIER (avec un frontal web classique et une API⁵ pour les besoins B2B). Il propose aux clients d'acheter des fichiers mis en ligne par des tiers. Nous retrouvons la possibilité de s'inscrire sur la plateforme afin de procéder à des ajouts de fichiers sur cette dernière dans un but commercial.

Le site côté serveur de production délivrant les pages web et l'API⁶, se compose de différents modules encapsulés par une gestion des événements et anomalies communes et granulaires.

Les différents logs d'erreurs génèrent des fichiers de texte brute qui sont envoyés par email pour ceux de niveaux critiques.

2. Les informations à traiter

Comme mentionné dans l'étude du fonctionnement du site, plusieurs événements et anomalies peuvent avoir lieues. Les différents niveaux de criticités dépendant d'une même classe mère connaîtront le même processus d'analyse. Ainsi, la seule différence au terme de l'analyse, sera le traitement de l'information par les administrateurs.

3. Les entités

La structure des entités gérant les erreurs à une composition d'entités imbriquées et se construit selon ce schéma :

⁵ Application Programming Interface

⁶ Application Programming Interface

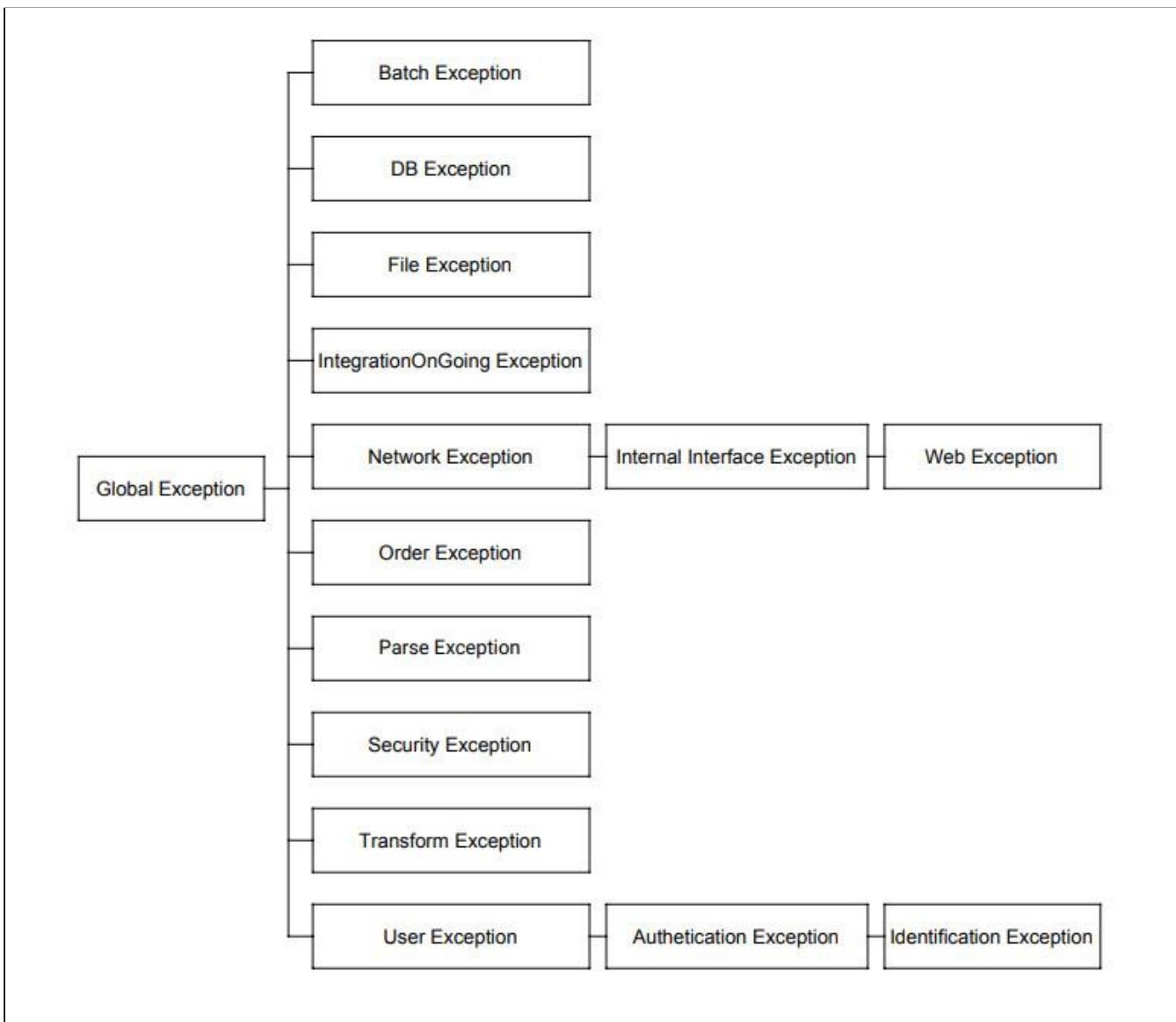


Fig. 1 : Schéma de hiérarchie des entités de gestion des exceptions.

Comme nous pouvons le constater à travers ce schéma, il existe 15 entités de gestion d'erreurs. Ce mécanisme de gestion des exceptions est intégré au langage JAVA⁷. De nombreuses méthodes des classes Java indiquent dans leur déclaration qu'elles peuvent lever une exception. Ceci garantit que certaines anomalies critiques seront prises explicitement en compte par la plateforme qui indiquera l'anomalie rencontrée aux utilisateurs avec un vocabulaire compréhensible tout en archivant les informations techniques en vues d'une analyse à postériori. Ces analyses permettent de rétablir l'état fonctionnel des comptes et achats effectués et de communiquer si besoin de façon ciblée vers les utilisateurs impactés. Dans le cas de la plateforme DataXchange, ce mécanisme standard est aussi exploité pour obtenir une traçabilité sur le fonctionnement sans anomalie de la plateforme (inscriptions utilisateurs, mises en vente de fichiers dématérialisés, achats et vente...)

Tout d'abord, arrêtons nous un instant sur la "Global Exception". Cette entité développée en JAVA, comporte une méthode dans laquelle toute une série de propriétés et de variables vont composer le corps des informations.

⁷ voir : <https://stackify.com/specify-handle-exceptions-java/>

C'est à partir de ces dernières que la qualification de la criticité va être stockée et se sont certains de ces champs qui vont devoir être affichés sur la page d'accueil du back-office.

Ces informations sont cruciales pour optimiser le temps d'identification, et donc de résolution, d'une anomalie.

Examinons maintenant les 10 entités filles de "Global Exception". Celles-ci vont hériter des éléments de leur mère et vont comporter des champs propres à la gestion de leur type d'erreur.

Enfin, nous constatons que les 4 dernières entités découlent du même procédé. D'une part la "Web Exception" fille de "Internal Interface Exception", elle-même fille de "Network Exception". Il en va de même pour "Identification Exception", héritant de "Authentication Exception", fille de "User Exception".

Une fois que la hiérarchie des entités et la prise de connaissance des propriétés et méthodes qui les composent sont établies, je démarre la tâche de modélisation de la base de données du back-office.

4. Challenges rencontrés

La principale difficulté rencontrée lors de cette étape préliminaire fut la décomposition des entités pour déterminer les éléments qui vont devoir être analysés et stockés. La lecture du langage JAVA, bien que différente du PHP, ne m'a pas posé de souci majeur, la logique étant très proche de ce que j'ai pu découvrir tout au long de ma formation en PHP / Symfony.

Pour illustrer cette partie, voici un extrait des classes JAVA pour la gestion des Exceptions et la schématisation des interactions de ma base de données :

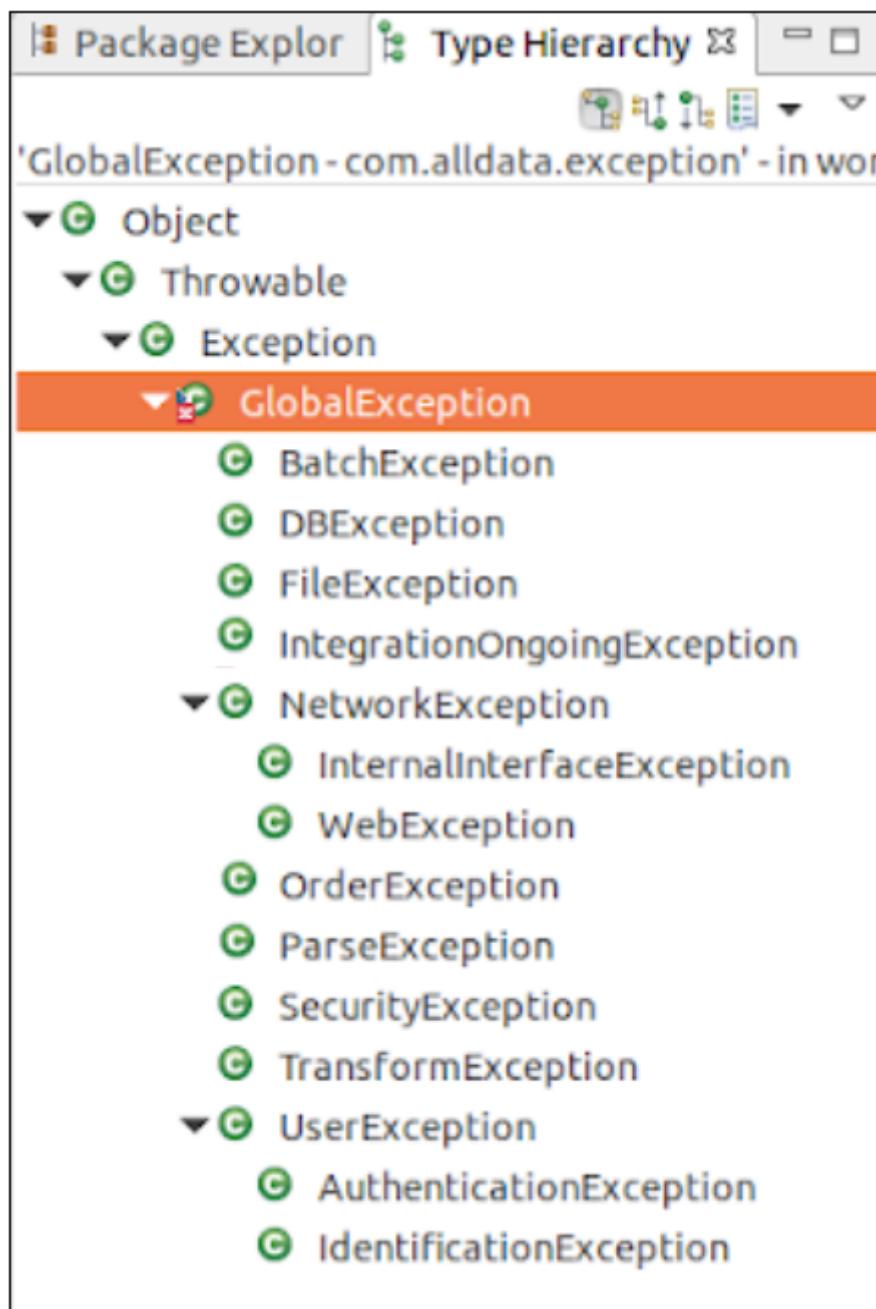


Fig. 2 : Schéma de hiérarchie des entités de gestion des exceptions JAVA

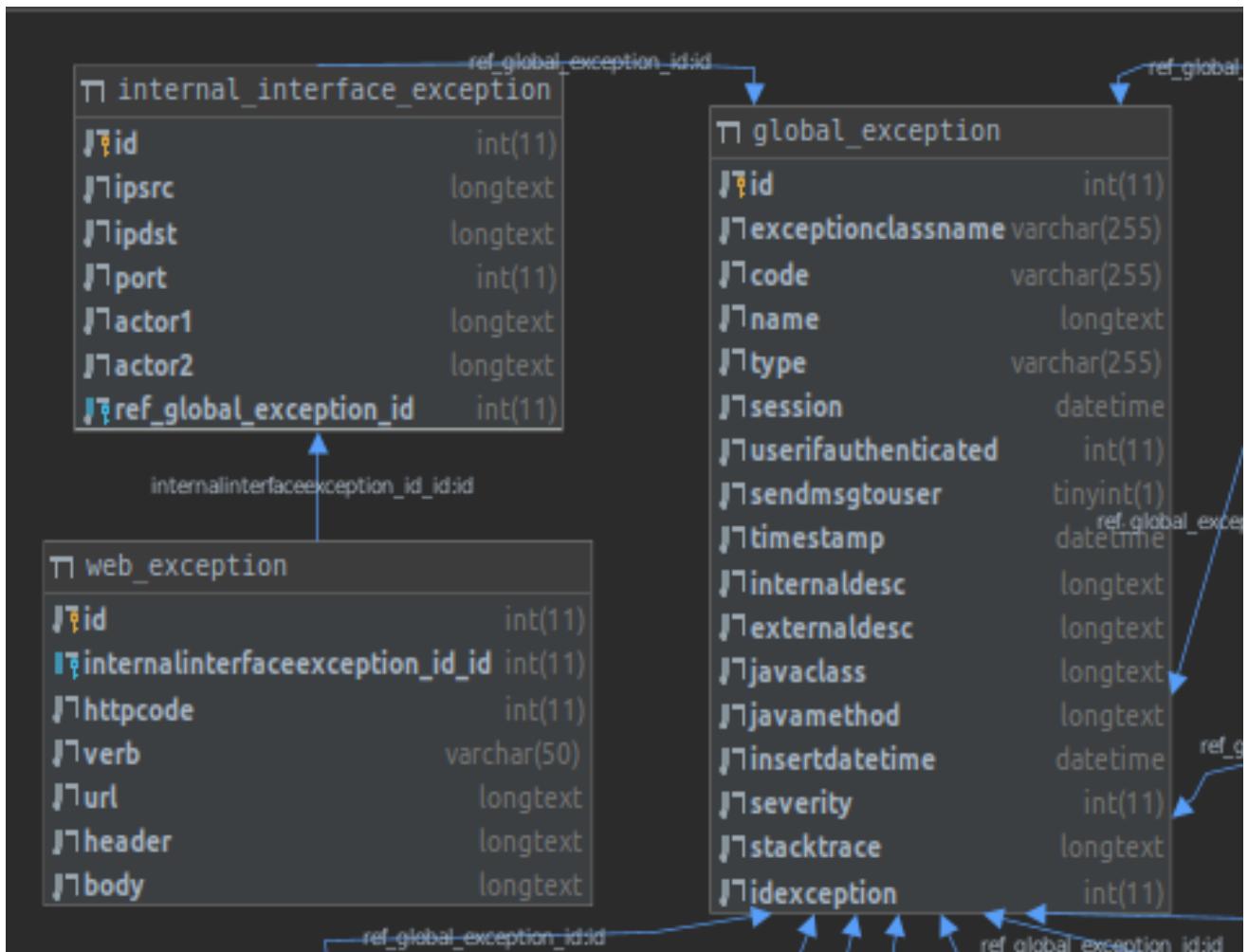


Fig. 3 : Schéma des entités de gestion des exceptions avec clés étrangères

Pour respecter cette hiérarchisation des classes JAVA, la solution a été d'établir des liens clés étrangères pointant sur une table mère contenant les informations à traiter. Pour se faire, j'ai dû lier (mapper) les objets Doctrine grâce aux annotations générées par la commande⁸ :

```
auzou@auzou-ThinkPad-L450:[~/Zento/Alldata] (master) $php bin/console make:entity
```

Cette commande me permet de déterminer un lien en suivant le protocole suivant :

Fig. 4 : Protocole pour créer les relations avec clés étrangères

⁸ Doc : <https://symfony.com/doc/current/Doctrine/associations.html>

```
Class name of the entity to create or update (e.g. BraveChef):
```

```
> Product
```

```
New property name (press <return>)
```

```
to stop adding fields):
```

```
> category
```

```
Field type (enter ? to see all types) [string]:
```

```
> relation
```

```
What class should this entity be related to?:
```

```
> Category
```

```
Relation type? [ManyToOne, OneToMany, ManyToMany, OneToOne]:
```

```
> ManyToOne
```

```
Is the Product.category property allowed to be null (nullable)? (yes/no) [yes]:
```

```
> no
```

```
Do you want to add a new property to Category so that you can access/update
```

```
Product objects from it - e.g. $category->
```

```
getProdu
```

```
> yes
```

```
New field name inside Category [products]:
```

```
> products
```

```
Do you want to automatically delete orphaned App\Entity\Product objects
```

```
(orphanRemoval)? (yes/no) [no]:
```

```
> no
```

```
New property name (press <return>)
```

```
to stop adding fields):
```

```
>
```

```
(press enter again to finish)
```

La relation “0 to 1” n’existant pas sous Doctrine, il suffit de choisir une relation en “OneToOne” et de définir la clé étrangère à “nullable”. Pour exemple, ci-dessous les changements apportés à ma classe “InternalInterfaceException” que j’ai lié avec la table “GlobalException” via une clé étrangère nommée “RefGlobalException”.

```
/**  
 * @ORM\OneToOne(targetEntity="App\Entity\GlobalException", cascade={"persist", "remove"})  
 * @ORM\JoinColumn(nullable=false)  
 */  
private $RefGlobalException;
```

Fig. 5 : Détails des changements apportés dans la classe InternalInterface

J'ai exclu deux entités JAVA de ma base de données, les classes "NetworkException" et "UserException". Le choix que j'ai fait n'incluant pas ces entités vient de l'examen de ces dernières, car les champs "ipSrc", "ipDst" et "port" sont présents dans ces deux tables filles.

```
GlobalException.java  IdentificationException.java  UserException.java  NetworkException.java

1 package com.alldata.exception;
2
3 public class NetworkException extends GlobalException {
4     /**
5      * 
6      */
7     private static final long serialVersionUID = -1401518579169505243L;
8
9     protected String ipSrc;
10    protected String ipDst;
11    protected int port;
12
13    public NetworkException(int code, String name, int type, String internalDesc, String externalDesc,
14        boolean sendMsgToUser, long userIfAuthenticated, String javaClass, String javaMethod,
15        Exception[] includedExceptions, String ipSrc, String ipDst, int port)
16    {
17        // Pour ajouter les informations propres à la sous-classe de GlobalException
18        @Override
19        public String getSpecificToString()
20        {
21            return super.getSpecificToString() + "ipSrc = '" + ipSrc + "' ; ipDst = '" + ipDst + "' ; port = '" + port + "' ;
22        }
23    }
24}
```

Fig. 6 : Détails de la classe JAVA NetworkException

The screenshot shows a Java code editor with several tabs at the top: GlobalException.java, IdentificationException.java, UserException.java, NetworkException.java, InternalInterfaceException.java, and WebI. The InternalInterfaceException.java tab is active. The code is as follows:

```
1 package com.alldata.exception;
2
3 public class InternalInterfaceException extends NetworkException {
4
5     /**
6      *
7      */
8     private static final long serialVersionUID = 8099755184362663170L;
9     protected String actor1;
10    protected String actor2;
11
12    public InternalInterfaceException(int code, String name, int type, String internalDesc,
13        String externalDesc, boolean sendMsgToUser, long userIfAuthenticated, String javaClass, String javaMethod,
14        Exception[] includedExceptions, String ipSrc, String ipDst, int port, String actor1, String actor2)
15    {
16        // Pour ajouter les informations propres à la sous-classe de GlobalException
17        //Override
18        public String getSpecificToString()
19        {
20            return super.getSpecificToString() + "actor1 = '" + actor1 + "' ; actor2 = '" + actor2 + "' ; ";
21        }
22    }
23
24}
```

Fig. 7 : Détails de la classe JAVA InternalInterfaceException

```
1 package com.alldata.exception;
2
3 public class WebException extends NetworkException {
4
5     /**
6      * 
7      */
8     private static final long serialVersionUID = -5436779197463053745L;
9     protected int httpCode;
10    protected String verb;
11    protected String url;
12    protected String header;
13    protected String body;
14
15    public WebException(int code, String name, int type, String internalDesc, String externalDesc,
16                        boolean sendMsgToUser, long userIfAuthenticated, String javaClass, String javaMethod,
17                        Exception[] includedExceptions, String ipSrc, String ipDst, int port, int httpCode, String verb, S
18
19    // Pour ajouter les informations propres à la sous-classe de GlobalException
20    @Override
21    public String getSpecificToString()
22    {
23        return super.getSpecificToString() + "httpCode = '" + httpCode + "' ; verb = '" + verb + "' ; url = '" +
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
```

Fig. 8 : Détails de la classe JAVA WebException

La classe “Network Exception”, contient des champs qui sont hérités par ses deux classes filles, à ce stade seules les classes filles peuvent être instanciées donc la table associée à la classe

NetworkException n'est pas instanciée à ce stade. Il est toutefois envisageable d'intégrer cette table si le besoin du client venait à évoluer.

B. Approche technique et organisation projet

1. Méthode de travail Agile

Pour organiser au mieux mon travail, il m'a fallu établir un rétro-planning lié au différentes tâches que j'aurai à exécuter. La réalisation de ce planning a été faite dès le début du projet au travers du logiciel Gantt⁹. Bien que le planning soit établi en amont d'un projet, celui-ci n'est pas figé et peut s'adapter au cours du développement en fonction des besoins et aléas.

Pour une meilleure réactivité dans l'évolution du projet, l'intégration de méthodes Agiles dans l'organisation des étapes de travail a été un choix judicieux et cohérent. Nous avons donc établi avec le Scrum Master des sprints de 5 jours.

Une fois la méthode de travail établie, je me suis attelé à la création d'un backlog¹⁰ conjointement avec le Product Owner de Zento. Ce backlog est plus proche dans sa réalisation d'une méthode en cascade.

Ce backlog m'a servi à prioriser mes actions en accord avec Zento dans le développement de l'application web et adapter la planification en évitant d'omettre des informations. On y retrouve l'organisation des fonctionnalités et leur interactivité.

2. Les différentes étapes du projet Gantt

Pour développer le back-office demandé, une rigueur de travail est importante. En effet, la réalisation en cascade de fonctionnalités est indispensable pour présenter des livrables dans les sprints impartis. Certaines tâches ayant comme pré-requis la création de fonctionnalités, elles-mêmes imbriquées dans une autre.

J'ai donc dans un premier temps réalisé une modélisation Merise de la base de données que je vais devoir manipuler afin d'en ressortir les informations prévues dans le cahier des charges.

Ceci fait, j'ai pris en charge la création du schéma de transition ("workflow") et de la maquette du back-office. Le modèle du "workflow" a été réalisé à la main. Pour la maquette¹¹, j'ai utilisé la plateforme MockFlow dans sa version gratuite.

3. Modélisation Merise

La réalisation de la modélisation a été pour moi une tâche prenante et ardue. Notamment parce que cette modélisation doit permettre de supporter les développements à venir et également parce qu'elle doit être cohérente par rapport à la modélisation de classes Java préexistantes. La conception de la table "Global Exception" m'a demandé du temps et plusieurs essais pour optimiser au mieux sa conception et permettre des interactions avec les tables qui lui sont liées. On retrouve des clés étrangères pour assurer la gestion des relations inter-tables. La table concernant les "Included Exception" m'a aussi confronté à un problème que je vais vous détailler.

⁹ Voir Annexes

¹⁰ Voir Annexes

¹¹ Voir Annexes

Toutes les Exceptions générées peuvent comporter en leur sein d'autres exceptions, les "Included Exception". Cette imbrication peut se retrouver de multiples fois et doit donc être stockées dans la BDD d'une manière qui permettent de les relier entre elles. Pour ce faire, il m'a donc fallu inclure une clé récursive. Ce sont les champs "prev_id" et "next_id" qui vont remplir ce rôle.

Dans son fonctionnement, si une "Included Exception" se retrouve dans le traitement d'une exception, cette nouvelle exception comportera un "next_id" et de ce fait, le "prev_id" se dotera automatiquement de l'ID de l'exception parente traitée.

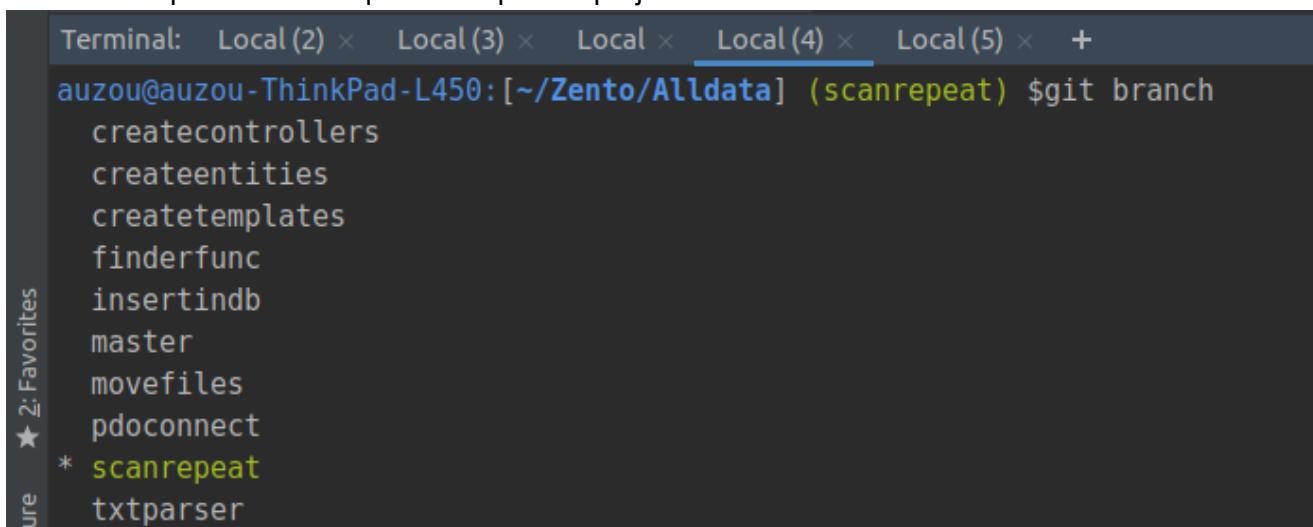
Included					
	id	INT	PK	AI	NN
0:1	previd_id	INT	FK		N
0:1	nextid_id	INT	FK		N
	finalrow	text			N
	RefGlobalException	INT	FK		NN
	isGlobalException	BOOL			NN

Fig. 9 : Schéma de la table "Included Exception"

4. Gestion de versions de fichiers Git

L'utilisation d'un système de gestion de versions de fichiers, s'il est indispensable lors de la réalisation d'un projet à plusieurs collaborateurs, n'en est pas moins important lorsqu'on travaille en autonomie. J'ai opté pour l'utilisation de Git et Github afin de pouvoir sauvegarder toutes les étapes d'avancement du projet et, si besoin, revenir à une version antérieure fonctionnelle.

La création de diverses branches nommées par rapport à la fonctionnalité développée m'a été très utile pour identifier quelles étapes du projet sont réalisées dans chacunes des branches.



```
Terminal: Local (2) × Local (3) × Local × Local (4) × Local (5) × +
aузou@auzou-ThinkPad-L450: [~/Zento/Alldata] (scanrepeat) $git branch
createcontrollers
createentities
createtemplates
finderfunc
insertindb
master
movefiles
pdoconnect
* scanrepeat
txtparser
```

The screenshot shows a terminal window with a dark background. At the top, there are five tabs labeled "Local (2)", "Local (3)", "Local", "Local (4)", and "Local (5)". The current tab is "Local". Below the tabs, the command "git branch" is run, and it lists several branches: "createcontrollers", "createentities", "createtemplates", "finderfunc", "insertindb", "master", "movefiles", "pdoconnect", and the current branch, "scanrepeat". The "master" branch is marked with an asterisk (*).

Fig. 10 : Commande de la liste des différentes branches du projet

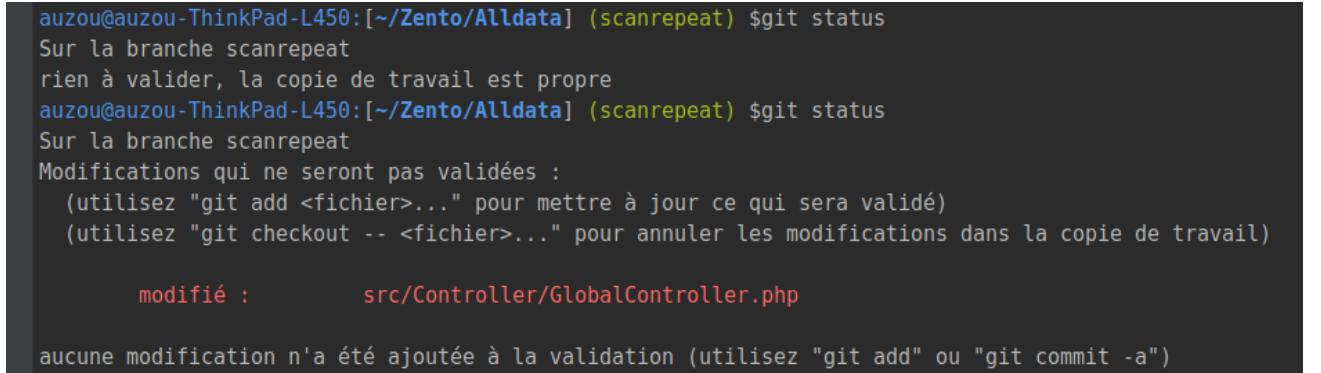
J'ai de plus configuré le dépôt distant sur Github afin de le rendre privé et assurer la non-divulgation d'informations confidentielles. J'y ai de plus ajouté comme unique collaborateur M. Amsellem.



```
auzou@auzou-ThinkPad-L450:[~/Zento/Alldata] (scanrepeat) $git remote -v
origin git@github.com:ArnaudAM/Zento.git (fetch)
origin git@github.com:ArnaudAM/Zento.git (push)
■ auzou@auzou-ThinkPad-L450:[~/Zento/Alldata] (scanrepeat) $
```

Fig. 11 : Commande vers le repo distant

Dernier point concernant l'utilisation de Git, il est crucial pour ne pas effectuer d'erreur, de vérifier régulièrement le suivi des fichiers.



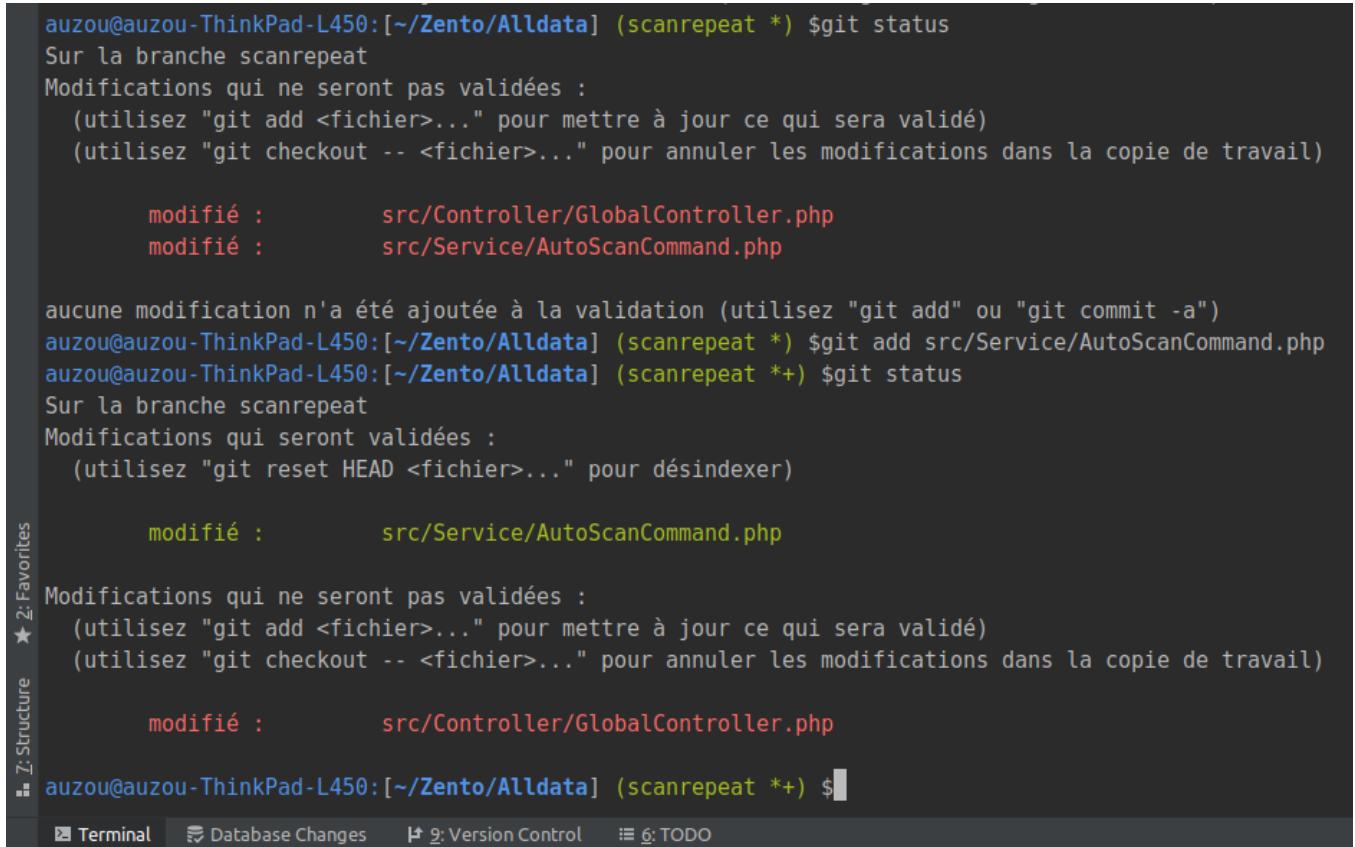
```
auzou@auzou-ThinkPad-L450:[~/Zento/Alldata] (scanrepeat) $git status
Sur la branche scanrepeat
rien à valider, la copie de travail est propre
auzou@auzou-ThinkPad-L450:[~/Zento/Alldata] (scanrepeat) $git status
Sur la branche scanrepeat
Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git checkout -- <fichier>..." pour annuler les modifications dans la copie de travail)

    modifié :      src/Controller/GlobalController.php

aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
```

Fig. 12 : Commande de vérification de suivi des fichiers

Dans le cas de modifications apportées dans une nouvelle branche, il sera plus simple d'ajouter uniquement les fonctionnalités liées à la branche en cours.



```
auzou@auzou-ThinkPad-L450:[~/Zento/Alldata] (scanrepeat *) $git status
Sur la branche scanrepeat
Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git checkout -- <fichier>..." pour annuler les modifications dans la copie de travail)

    modifié :      src/Controller/GlobalController.php
    modifié :      src/Service/AutoScanCommand.php

aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
auzou@auzou-ThinkPad-L450:[~/Zento/Alldata] (scanrepeat *) $git add src/Service/AutoScanCommand.php
auzou@auzou-ThinkPad-L450:[~/Zento/Alldata] (scanrepeat *) $git status
Sur la branche scanrepeat
Modifications qui seront validées :
  (utilisez "git reset HEAD <fichier>..." pour désindexer)

    modifié :      src/Service/AutoScanCommand.php

Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git checkout -- <fichier>..." pour annuler les modifications dans la copie de travail)

    modifié :      src/Controller/GlobalController.php

■ auzou@auzou-ThinkPad-L450:[~/Zento/Alldata] (scanrepeat *) $
```

Fig. 13 : Commande d'ajout de suivi des fichiers ciblé

5. Challenges rencontrés

Comme je l'ai mentionné plus haut, la principale difficulté que j'ai rencontré à ce stade d'avancement du projet est la modélisation de la BDD afin de pouvoir prendre en charge l'intégralité des ressources qui seront fournies lors de la génération des erreurs. Mon maître de stage m'a beaucoup aidé dans la mise en place de la récursivité de la table "Included_exception".

Included					
	id	INT	PK	AI	NN
0:1	prev_id	INT	FK		N
0:1	nextid_id	INT	FK		N
	finalrow	text			N
	RefGlobalException	INT	FK	NN	0:
	isGlobalException	BOOL		NN	

Fig. 14 : Commande d'ajout de suivi des fichiers ciblé

Pour expliquer plus en détails la manière dont agit cette clé recursive, je vais prendre un exemple simple :

- Supposons qu'une "GlobalException" générée contienne deux "IncludedException" de type "GlobalException".
- La première "IncludedException" détectée est liée par sa clé étrangère "isGlobalException" à la "GlobalException" racine.
- La deuxième "IncludedException", va suivre le même protocole, en base de données elle aura en plus son champ "prev_id" pointant sur l'id de la première "IncludedException".

Included N°1		Global	
0:1	0:n	0:1	
id	1	id	1
prev_id	NULL	ExceptionClassName	GlobalException
nextid_id	2	code	STR NN
finalrow	NULL	name	text NN
RefGlobalException	1	type	str NN
isGlobalException	TRUE	session	datetime NN
		userifauthenticated	int NN
		sendmsgstouser	BOOL NN
		timestamp	datetime NN
		internaldesc	text NN
		externaldesc	text NN
		javaclass	text NN
		javamethod	text NN
		insertdatetime	datetime NN

Included N°2	
0:1	
id	2
prev_id	1
nextid_id	NULL
finalrow	TRUE
RefGlobalException	1
isGlobalException	TRUE

Fig. 15 : Exemple d'interaction des IncludedException

Une autre difficulté fut la création de la maquette avec le choix d'afficher ou non certaines informations au chargement de la page. Pour reprendre l'exemple des “IncludedException”, j'ai fini par opter pour un affichage de cette section en “cacher” avec la possibilité d'agrandir l'affichage pour permettre l'examen de cette exception. Cela permet aussi d'épurer le nombre d'éléments sur la page et d'assurer une meilleure lisibilité.

C. Choix techniques

1. Back-end

La phase de conception du projet achevée, j'ai maintenant en charge le développement de l'application web. Pour se faire, j'ai porté mon choix sur le langage PHP 7. La formation suivie à la Wild Code School étant centrée sur l'utilisation de ce langage et de son framework Symfony. La conception de la base de données a été réalisée grâce à l'ORM Doctrine.

a. Le langage de développement PHP 7

Avant de développer plus en avant la réalisation de ce projet, je me permets d'effectuer un rappel sur PHP. C'est un langage typé de scripts généralistes et Open Source, il est spécialement conçu pour le développement d'applications web.

L'utilisation de ce langage intervient dans l'ensemble du projet. Il permet de mettre en place des fonctions natives assurant un gain de temps dans la création de méthodes. Ainsi, il m'a permis de créer le “parsing” du fichier texte généré lorsqu'une exception est levée, tout comme il me permet de créer des boucles et conditions et de manipuler les objets. Il en va de même pour l'analyse au fil de l'eau des erreurs rencontrées.

b. Le framework Symfony 4

Pour travailler dans l'environnement Symfony, il faut suivre un processus suivant les besoins du projet. Il faut pour cela créer un dossier de travail et suivre les différentes étapes liées à l'utilisation de Symfony 4 et de ses bibliothèques. Il faut ensuite lier le projet à un dépôt distant afin d'assurer la sauvegarde de versions antérieures en cas de régressions lors de l'ajout nouvelles fonctionnalités. J'ai fait le choix de Git et GitHub pour assurer la gestion des versions de fichiers.

Ci-dessous, les commandes à réaliser lors de la création d'un nouveau projet en PHP 7 / Symfony 4 :

```
→ auzou@auzou-ThinkPad-L450:[~] $mkdir Stage
→ auzou@auzou-ThinkPad-L450:[~] $cd Stage/
→ auzou@auzou-ThinkPad-L450:[~/Stage] $composer create-project symfony/website-skeleton AllData
→ auzou@auzou-ThinkPad-L450:[~/Stage] $cd AllData
→ auzou@auzou-ThinkPad-L450:[~/Stage/AllData] $
→ auzou@auzou-ThinkPad-L450:[~/Stage/AllData] $git init
→ auzou@auzou-ThinkPad-L450:[~/Stage/AllData](master) $touch README
→ auzou@auzou-ThinkPad-L450:[~/Stage/AllData](master) $git remote add origin
→ auzou@auzou-ThinkPad-L450:[~/Stage/AllData](master) $git add .
→ auzou@auzou-ThinkPad-L450:[~/Stage/AllData](master) $git commit -m "First commit"
→ auzou@auzou-ThinkPad-L450:[~/Stage/AllData](master) $git push origin master
→ auzou@auzou-ThinkPad-L450:[~/Stage/AllData](master) $yarn install
→ auzou@auzou-ThinkPad-L450:[~/Stage/AllData](master) $php bin/console server:run
→ auzou@auzou-ThinkPad-L450:[~/Stage/AllData](master) $yarn encore dev --watch
```

De plus l'utilisation du package “website-skeleton” incorporant un ensemble de bibliothèques

va permettre par exemple l'utilisation du moteur de template TWIG, de récupérer facilement et lisiblement les informations stockées en base de données ou bien encore la gestion des "Callers/Listeners". Les appels des méthodes liées à la lecture de la base de données étant générés automatiquement, cela permet un gain de temps important par rapport à une utilisation native de PHP et de MySQL. Ce gain est corrélé au besoin du projet, par exemple : si des requêtes spécifiques sont majoritaires dans un projet, l'utilisation de Doctrine n'est pas conseillé.

De même, il est possible grâce à ce framework de créer une gestion des utilisateurs rapide et aisément configurable et de créer une fonction de filtres et de tris des éléments d'un tableau.

Pour assurer une maintenabilité fiable, il est préconisé de fractionner son code de manière à rendre l'identification des éléments aisée. J'ai donc placé les éléments liés au traitement des fichiers textes dans une partie Service qui va s'occuper des différentes étapes.

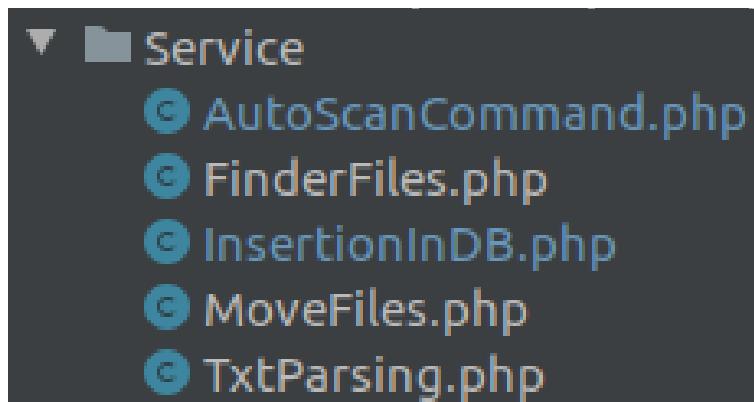


Fig. 16 : Les Services

i. Challenges rencontrés

Il est évident que l'utilisation du framework Symfony 4 permet un gain de temps considérable ainsi qu'une meilleure sécurité interne générale. Cependant, j'ai eu quelques difficultés à comprendre ce que l'utilisation de certains packages sous-entendaient. Comme je l'ai dit précédemment, j'ai choisi d'utiliser le package "website-skeleton" comme "boîte à d'outils" pour la réalisation du projet back-office. L'utilisation que j'en avais fait dans mes précédents projets était quasi-identique. Aussi, je n'avais pas saisi le nombre de bibliothèques utilisées.

Jusqu'à maintenant, j'ai surtout utilisé les annotations pour la définition des routes et url, ainsi que pour la gestion de l'ORM Doctrine. La prise de conscience des possibilités apportées par ce package m'est apparu lors de la création de la fonctionnalité d'insertion en base avec les "callback"¹². J'ai ainsi réalisé que tout n'était pas prévu dans ce package lors de la phase d'automatisation du scan¹³, de la présence des fichiers dans le fichier dédié aux retours de fichiers textes d'erreur JAVA de la plateforme DataXchange.

Je détaillerai ces deux points plus explicitement dans les prochains chapitres.

c. L'ORM Doctrine

i. Lien à la base et création des entités

¹² Voir chapitre III.C.1.c.a1

¹³ Voir chapitre III.C.1.g

Comme expliqué précédemment, j'ai utilisé l'ORM Doctrine pour la création de la BDD du back-office. Il permet de créer une BDD relationnelle sous la forme de base de données "objet". Cela permet de manipuler plus facilement les données persistantes du site.

Pour pouvoir utiliser Doctrine, il faut configurer le chemin d'accès à la base de données via le fichier ".env.local" copie du fichier ".env".

```
DATABASE_URL=mysql://root:@127.0.0.1:3306/zentoalldata?serverVersion=5.7
```

Fig. 17 : configuration du chemin et des autorisations pour l'accès à la BDD

Pour utiliser facilement Doctrine, il est impératif d'être conscient que toute erreur faite lors de la création de l'entité amènera à effectuer des modifications dans les entités associées créées. Par exemple, la création de l'entité "Global Exception" va générer 3 entités différentes, qui sont respectivement :

- GlobalException
- GlobalExceptionController
- GlobalExceptionRepository

```
<?php

namespace App\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity(repositoryClass="App\Repository\GlobalExceptionRepository")
 */
class GlobalException
{
    /**
     * @ORM\Id()
     * @ORM\GeneratedValue()
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $exceptionclassname;

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $code;

    /**
     * @ORM\Column(type="text")
     */
    private $name;

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $type;

    /**
     * @ORM\Column(type="datetime")
     */
}
```

Fig. 18 : Présentation d'un extrait de l'entité “GlobalException”

```
<?php

namespace App\Controller;

use App\Repository\GlobalExceptionRepository;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

/**
 * Class GlobalController
 * @package App\Controller
 * @Route("/global", name="home_")
 */
class GlobalController extends AbstractController
{
    /**
     * @Route("/", name="global")
     */
    public function index(GlobalExceptionRepository $globalExceptionRepository)
    {
        return $this->render( view: 'global.html.twig', [
            'globalexception' => $globalExceptionRepository->findAll(),
        ]);
    }
}
```

Fig. 19 : Présentation de l'entité “GlobalExceptionController”

```
<?php

namespace App\Repository;

use ...

/**
 * @method GlobalException|null find($id, $lockMode = null, $lockVersion = null)
 * @method GlobalException|null findOneBy(array $criteria, array $orderBy = null)
 * @method GlobalException[] findAll()
 * @method GlobalException[] findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
 */
class GlobalExceptionRepository extends ServiceEntityRepository
{
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, entityClass: GlobalException::class);
    }
}
```

Fig. 20 : Présentation de l'entité “GlobalExceptionRepository”

Pour arriver à ce résultat, voici un exemple du processus à suivre depuis le terminal de l'IDE PhpStorm :

- auzou@auzou-ThinkPad-L450:[~/Stage/AllData](master) \$php bin/console Doctrine:database:create
- auzou@auzou-ThinkPad-L450:[~/Stage/AllData](master) \$php bin/console make:entity

Ci-dessous, un exemple du procédé de création d'une entité¹⁴ :

```
$ php bin/console make:entity

Class name of the entity to create or update:
> Product

New property name (press <return>                               to stop adding fields):
> name

Field type (enter ? to see all types) [string]:
> string

Field length [255]:
> 255

Can this field be null in the database (nullable) (yes/no) [no]:
> no

New property name (press <return>                               to stop adding fields):
> price

Field type (enter ? to see all types) [string]:
> integer

Can this field be null in the database (nullable) (yes/no) [no]:
> no

New property name (press <return>                               to stop adding fields):
>

(press enter again to finish)
```

Fig. 21 : Les étapes de création d'une entité

A la fin de ces étapes préliminaires, les entités sont créées dans le projet mais les tables n'existent pas encore dans MySQL. Pour que la classe créée existe en tant que table dans la base de données, il faut inscrire d'autres commandes.

La première commande va ajouter au fichier des migrations les requêtes SQL permettant de créer les tables dans la base. La deuxième va elle, exécuter les requêtes.

```
$ php bin/console make:migration
```

```
$ php bin/console doctrine:migrations:migrate
```

Fig. 22 : Commande pour créer la table en base et l'insérer en base

¹⁴ Doc Entités Doctrine : <https://symfony.com/doc/current/Doctrine.html>

Dans PHP, l'utilisation de Doctrine lors de l'insertion en base intervient après l'instanciation de l'objet relié à la BDD.

ii. Gestion des insertions en base de données

Pour réaliser l'insertion en base de données, j'ai créé une classe "InsertionInDB", qui va me permettre de récupérer les informations et de les insérer en prenant en compte l'"ExceptionClassName". J'ai donc besoin dans un premier temps de créer un constructeur me donnant accès à l'EntityManager. Ce dernier me permet de gérer les informations parsées comme un objet et me permet d'utiliser Doctrine. Dans un second temps, je dois gérer les informations générales à toutes les tables en passant par ma table "global_exception". Une fois mon objet instancié, il est obligatoire de renseigner les champs qui seront remplis dans la base de données grâce à des "setters".

```
/*
 * @var EntityManagerInterface
 */
private $entityManager;

public function __construct(EntityManagerInterface $entityManager)
{
    // parent::__construct($entityManager);
    $this->entityManager = $entityManager;
}

public function dbInsertion($result)
{
    $e = new GlobalException();
    $e->setExceptionclassname($result['ExceptionClassName']);
    $e->setSession(\DateTime::createFromFormat('Y/m/d H:i:s', $result['session']));
    $e->setExceptionid($result['id']);
    $e->setSeverity($result['severity']);
    $e->setName($result['name']);
    $e->setInternaldesc($result['internalDesc']);
    $e->setCode($result['code']);
    $e->setType($result['type']);
    $e->setExternaldesc($result['externalDesc']);
    $e->setSendmsgtouser($result['sendMsgToUser']);
    $e->setTimestamp($result['timestamp']);
    $e->setUserifauthenticated($result['userIfAuthenticated']);
    $e->setJavaclass($result['javaClass']);
    $e->setJavamethod($result['javaMethod']);
    $e->setStacktrace($result['StackTrace']);
}
```

Fig. 23 : Présentation de la classe DBInsertion

Il faut maintenant s'occuper des différents cas d'exceptions. Pour cela j'ai utilisé la fonction "switch case" en utilisant comme paramètre le nom de l'exception résultant du parsing.

```
switch ($result['ExceptionClassName'])
{
```

Fig. 24 : Fonction switch

Je définis ensuite les actions à réaliser pour gérer ces exceptions. Afin d'étayer ce point, prenons le cas de la table “web_exception”.

```
case "InternalInterfaceException":  
    $iI = new InternalInterfaceException();  
    $iI->setRefGlobalException($e);  
    $iI->setIpSrc($result['ipSrc']);  
    $iI->setIpDst($result['ipDst']);  
    $iI->setPort($result['port']);  
    $iI->setActor1($result['actor1']);  
    $iI->setActor2($result['actor2']);  
  
    $this->entityManager->persist($iI);  
    $this->entityManager->flush();  
    break;  
  
case "WebException":  
    $iI = new InternalInterfaceException();  
    $iI->setRefGlobalException($e);  
    $iI->setIpSrc($result['ipSrc']);  
    $iI->setIpDst($result['ipDst']);  
    $iI->setPort($result['port']);  
    $iI->setActor1($result['actor1']);  
    $iI->setActor2($result['actor2']);  
  
    $web = new WebException();  
    $web->setInternalinterfaceexceptionId($iI);  
    $web->setHttpcode($result['httpCode']);  
    $web->setVerb($result['verb']);  
    $web->setUrl($result['url']);  
    $web->setHeader($result['header']);  
    $web->setBody($result['body']);  
  
    $this->entityManager->persist($web);  
    $this->entityManager->flush();  
    break;
```

Fig. 25 : Le cas WebException

Comme je l'ai déjà mentionné, la table “web_exception” dépend de la table “internalinterface_exception” ; je dois donc gérer plusieurs cas de figure : soit l'erreur générée est liée au web, soit elle est liée à l'interface interne. Par conséquent, il faut instancier mon objet

“InternalInterfaceException” lors du cas d'une “WebException”. Je commence par instancier la classe “InternalInterfaceException” puisque l'id de cette dernière va me permettre de la relier à mon entité “WebException”.

L'entityManager me permet de sauvegarder les informations du premier objet “GlobalException” et de les appeler lors de mes autres instances de classes. Dans cet exemple, les informations sont stockées dans la variable “\$e”, puis placées en paramètre de mon entité “InternalInterfaceException” et sauvegardées dans ma variable “\$il”. Je réitère ce procédé pour mon instantiation de “WebException” avec la variable “\$il” qui sera stockée dans “\$web”. J'ajoute ensuite cette variable en base.

\$this->entityManager->persist();

La fonction “persist()” lie l'objet à la base de données et c'est à partir de là que les informations remplies seront utilisées.

\$this->entityManager->flush();

Cette fonction quand à elle, va permettre à Doctrine de vérifier les informations données et d'appliquer les instructions.

Ces appels aux méthodes de l'entityManager doivent être fait à la fin de la réalisation de chaque cas.

À la fin des différents cas gérés, j'ai ajouté l'action à effectuer par défaut si aucun des cas enregistrés n'est sollicité.

```
default :  
    echo "abnormal";  
    break;  
}
```

Fig. 26 : Cas par défaut

La particularité rencontrée dans ce projet est la création d'un champ servant à enregistrer la date courante correspondant à l'insertion en base de données de l'exception générée. Ce besoin fait suite à une volonté de pouvoir comparer la date des erreurs remontées par la gestion des exceptions JAVA avec celle de son traitement sur la plateforme back-office. Cette partie m'a confronté à une nouvelle façon de gérer l'insertion en base que je détaille dans la chapitre suivant.

iii. Challenges rencontrés

La principale difficulté que j'ai rencontré dans la réalisation du “back-end” est liée au champ “insertdatetime” de mes tables. Ce champ correspond à la génération de la date et de l'heure à laquelle l'exception est insérée dans la base de données.

L'erreur remontée m'indiquant que le champ ne pouvant pas être nul, doit être renseigné pour que l'insertion se fasse. Ce champ est défini en annotation de la façon suivante :

```
/**  
 * @ORM\Column(name="insertdatetime", type="datetime", options={"default"="CURRENT_TIMESTAMP"})  
 */  
private $insertdatetime;
```

Fig. 27 : Propriété du champ “insertdatetime”

Pour remédier à cela, j'ai découvert une annotation spécifique permettant de définir au moment de l'instanciation de ce champ la valeur à attribuer. J'ai donc ajouté l'appel d'une classe¹⁵ qui me permet de modifier le “setter” de la propriété du champ “insertdatetime” comme suit :

```
/**  
 * @ORM\Entity(repositoryClass="GlobalExceptionRepository")  
 * @ORM\HasLifecycleCallbacks()  
 */  
class GlobalException
```

Fig. 28 : Paramétrage de la classe “@ORM\HasLifecycleCallbacks”

```
/**  
 * @ORM\PrePersist  
 */  
public function setInsertdatetime(): self  
{  
    $this->insertdatetime = new \DateTime(); //  
  
    return $this;  
}
```

Fig. 29 : Fonction spécifique pour le setter le champ “insertdatetime”

La particularité de la bibliothèque `@ORM\HasLifecycleCallbacks()`, me permet d'utiliser la pré-persistances dans mon entité. Le problème venant initialement au moment de la commande :

```
$this->getDoctrine->getManager()->persist($r);
```

Cela permet de donner l'instruction que la commande de la figure 29 doit s'effectuer avant le “persist”.

L'anomalie se résout automatiquement, puisqu'au moment de l'appel de “persist” le champ “insertdatetime” non annoncé dans les setters est initialisé automatiquement par la base de données.

¹⁵ Doc : <https://symfony.com/doc/current/Doctrine/events.html>

d. Parser PHP

Pour insérer les informations dans la base de données telle qu'elle a été conçue, j'ai établi les critères des différents champs en fonction des informations reçues dans le fichiers de logs d'erreurs généré par JAVA et envoyé en format texte vers le back-office.

J'ai ainsi dû créer une fonction utilisant différentes fonctions natives de PHP :

```
class TxtParsingController extends AbstractController
{
    public function parseContent(string $txt): array
    {
        $txtexplode = explode( delimiter: ";", string: "$txt");
        $i = 0;
        $t[] = '';

        for ($i; $i < sizeof($txtexplode); $i++) {
            $distinctElem = explode( delimiter: "=>", string: "$txtexplode[$i]");

            if (sizeof($distinctElem) > 1)
            {
                $t[trim($distinctElem[0])] = trim($distinctElem[1], charlist: " \n");
            }
            else
            {
                echo "\n skipping element un parseContent, no Key/value pair !! \n";
            }
        }
        return $t;
    }
}
```

Fig. 30: Fonction “parseContent”

J'ai d'abord créé une variable contenant le délimiteur “;” séparant les éléments à exploser¹⁶. J'ai établi ce délimiteur suite à l'examen des fichiers d'erreurs existants qui m'ont été fournis par le Product Owner. Le tableau généré m'a permis d'accéder aux valeurs que j'ai dû séparer pour le rendre exploitable.

Comme je souhaitais utiliser une boucle “for” pour parcourir le tableau “\$txtexplode” et que j'avais besoin de récupérer les informations du prochain “explode”, j'ai créé aussi un tableau “\$t”.

J'ai ensuite réitéré l'explode sur “\$txtexplode” avec le nouveau délimiteur “=>”. Pour les besoins du client qui utilisait déjà le signe “=” dans le contenu de certaines informations, j'ai remis au développeur de référence DataXchange une synthèse des éléments à modifier dans les classes JAVA gérant la génération des messages d'erreurs.

L'intégration de la condition à l'explosion du dernier élément est là pour s'assurer que dans le cas où l'explosion ne conviendrait pas à l'insertion en base en raison d'un manque de lien “clé->valeur”, un message d'erreur sera affiché.

¹⁶ Voir Annexes : Log d'erreur JAVA généré

Dans le cas où l'explosion de “\$txtexplode” correspond au schéma attendu, je définis le premier élément du tableau “\$t” comme “clé” et le second comme “valeur”. Le texte présente des guillemets et des espaces qui entrent en conflit avec SQL car sensibles à la casse. Aussi j'utilise la fonction native “trim” pour supprimer les espaces et caractères inutiles. Après avoir réalisé ces différentes étapes, la clé correspond au nom du champ dans ma table et le second à la valeur à insérer dans la base¹⁷.

e. Finder Component

L'objectif du back-office étant l'intégration, le traitement et la lisibilité des exceptions JAVA générées. La mise en place d'un scan de répertoire pour y lire la présence des fichiers contenant les erreurs à traiter est indispensable.

Pour répondre à ce besoin, il existe le composant Symfony “Finder”¹⁸ que l'on peut utiliser ainsi :

use Symfony\Component\Finder\Finder;

Ci-dessous la manière de l'utiliser :

```
/* START FINDER */
echo "<br/><br/>GO FILES FINDER<br/><br/>";
$finder = new Finder();

// find all files in Files directory
$finder->files()->in( dirs: '/home/auzou/Zento/Alldata/src/Files');

// check if there are any search results in Files directory
if ($finder->hasResults())
{
    // Read contents of return files
    foreach ($finder as $file)
    {
        $txt = $file->getContents();
        var_dump($txt);
        echo "<br/><br/>END FILES FINDER<br/><br/>";
    }
    /* END FINDER */
```

Fig. 31 : Utilisation de la classe Finder pour le scan de fichier

Le composant utilisé permet de faire référence à une classe Finder.php et donc de pouvoir instancier cet objet par la première ligne. Les méthodes “files()”, “in()”, “hasResults()” et “getContents()” dépendent de cette classe Finder.

La méthode “files()” permet de préciser que le traitement concerne des fichiers et la méthode “in()” doit contenir le chemin absolu vers le dossier visé.

La méthode “hasResults()” va tester la présence de fichiers dans le répertoire mentionné dans “in()”. Si des éléments correspondant à “files()” sont trouvés, la boucle sur le tableau de contenant n objet \$finder va récupérer le contenu de chaque fichier via “getContents()”.

¹⁷ Voir Annexes : var_dump du Parser

¹⁸ Doc : <https://symfony.com/doc/current/components/finder.html>

f. Déplacement du fichier après traitement

Pour prévenir de la possibilité de rentrer plusieurs fois les mêmes informations en base, le Product Owner et moi avons décidé que le fichier une fois traité devait être déplacé dans un autre répertoire.

Afin de pouvoir insérer cette fonctionnalité que je n'avais jamais rencontré, je suis allé rechercher ce que Symfony proposait et après quelques documentations lues, j'ai décidé d'utiliser la fonction native "rename". Cette fonction, utilisant le chemin absolu rend aisément le déplacement de l'élément voulu dans le dossier approprié. La variable "\$file" correspond à la variable définie dans la boucle "for".

```
// find all files in Files directory
$finder->files()->in( dirs: '/home/auzou/Zento/Alldata/src/Files');
```

Fig. 32 : Chemin contenant le fichier à traiter

```
$uploadsDir = '/home/auzou/Zento/Alldata/src/Files/Treated/';
// $toMove = $finder->files()->in('/home/auzou/Zento/Alldata/src/Files');
rename( oldname: $file->
    getPath() . '/' . $file->
    getFilename(), newname: $uploadsDir . $file->
    getFilename()
);
```

Fig. 33 : Fonction rename

g. Exécution périodique du traitement des exceptions

Pour effectuer une exécution cyclique du Finder Component, j'ai décidé de rechercher ce que les bibliothèques Symfony pouvaient proposer. Les fonctions "Ev"¹⁹, "EvTimer" sont celles qui semblent le plus correspondre à ce besoin. Néanmoins, l'utilisation de la bibliothèque "Ev" a été assez laborieuse à effectuer.

J'ai donc choisi de m'orienter vers la planification de tâches via CRON²⁰. Les tests ont été concluants à ce stade malgré des difficultés lors de l'exécution du scan. En créant l'action via la planification, une erreur concernant le manque de la bibliothèque Doctrine apparaissait alors qu'elle était bien présente dans les appels liés aux classes utilisées.

¹⁹ Doc : <https://www.php.net/manual/fr/book.ev.php>

²⁰ Doc : <https://doc.ubuntu-fr.org/cron>

```
GNU nano 2.9.3                                         /tmp/
```

```
# Edit this file to introduce tasks to be run by cron.  
#  
# Each task to run has to be defined through a single line  
# indicating with different fields when the task will be run  
# and what command to run for the task  
#  
# To define the time you can provide concrete values for  
# minute (m), hour (h), day of month (dom), month (mon),  
# and day of week (dow) or use '*' in these fields (for 'any').#  
# Notice that tasks will be started based on the cron's system  
# daemon's notion of time and timezones.  
#  
# Output of the crontab jobs (including errors) is sent through  
# email to the user the crontab file belongs to (unless redirected).  
#  
# For example, you can run a backup of all your user accounts  
# at 5 a.m every week with:  
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/  
#  
# For more information see the manual pages of crontab(5) and cron(8)  
#  
# m h dom mon dow   command  
*/1 * * * * /usr/bin/php7.2 /home/auzou/Zento/Alldata/bin/console app:AutoScan
```

Fig. 34 : Configuration de la planification CRON

Pour éclaircir ce point, voici les détails des critères à remplir et les informations que j'ai choisi d'utiliser :

m : minute -> */1
h : heure -> *
dom : jours dans le mois -> *
mon : mois -> *
dow : jour de la semaine -> *
command : La commande à lancer

La partie "Command" étant plus conséquente, je vous détaille ci-dessous les chemins absous inscrits.

Pour la première partie :

/usr/bin/php7.2 : Pour spécifier que l'élément est du PHP

Pour la deuxième partie :

/home/auzou/Zento/Alldata/bin/console : Va exécuter une commande PHP

Pour la troisième partie :

app:Autoscan : Lance la méthode liée grâce à la bibliothèque Command

Début d'édition :

Une anomalie est apparue dans l'exécution de cette Command, la méthode exploitée pour obtenir l'EntityManager et les fonctionnalités Doctrine ad-hoc dans le cas d'une classe Controller ne fonctionnent pas pour une classe Command.

Dans le cas d'une classe Controller, héritant de AbstractController, il est possible d'utiliser la méthode de la classe parente pour obtenir l'objet Doctrine et son EntityManager :

\$this->getDoctrine->getManager()->persist(\$r);

Mais dans notre cas nous héritons de Command pour pouvoir être exécuté au travers du planificateur de tâches Cron. Après de nombreux tests, investigation et essais, la solution trouvée est de déclarer l'EntityManager comme attribut du constructeur afin qu'à l'exécution le framework puisse le mettre à disposition de notre classe héritant de Command.

Le problème lié à l'utilisation de Doctrine est réglé grâce à l'utilisation de l'EntityManager qui utilise déjà l'ORM Doctrine et qui me permet de créer la persistance en base de données.

```
use App\Repository\GlobalExceptionRepository;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Component\Console\Command\Command;
use Symfony\Component\Console\Input\InputInterface;
use Symfony\Component\Console\Output\OutputInterface;
use Symfony\Component\Finder\Finder;

class AutoScanCommand extends Command//extends ContainerAwareCommand//
{
    protected static $defaultName = 'app:autoscan';
    /**
     * @var EntityManagerInterface
     */
    private $entityManager;

    public function __construct(EntityManagerInterface $entityManager)
    {
        parent::__construct();
        $this->entityManager = $entityManager;
    }

    protected function execute(InputInterface $input, OutputInterface $output)
    {
        $output->writeln([
            'Try use (new Autoscancmmand($this))->scanNow() with php bin/console',
            '=====',// Another line
            ''// Empty line
        ]);

        $this->scanNow();

        $output->writeln([
            '=====',// Another line
            'My Final Symfony command',// A line
            ''// Empty line
        ]);
    }
}
```

Fig. 35 : Présentation de l'entité AutoScanCommand

```
function scanNow(): void
{
    $g = new FinderFiles($this->entityManager);
    $g->findFiles();

}
```

Fig.36 : Présentation de la méthode ScanNow

L'entité créée pour la réalisation du scan automatique des fichiers présents utilise la bibliothèque “Command”, celle-ci me permet de définir par app:AutoScan dans le CRON la classe à exécuter. Cette exécution fait appel à la méthode du composant “Command” s’appelant “execute”. Je lui fourni la fonction “scanNow” qui va lancer tout le processus de traitement des données.

Fin d'édition

h. Sécurité

Dans cette partie, j’aborde la sécurité avec un besoin d’authentification de l’utilisateur via un ensemble de composants Symfony “User”. Pour réaliser cette tâche, je dois d’abord utiliser “Login Form”²¹ afin de créer une table d’utilisateur dans laquelle des rôles vont être attribués.

```
$ php bin/console make:auth

What style of authentication do you want? [Empty authenticator]:
[0] Empty authenticator
[1] Login form authenticator
> 1

The class name of the authenticator to create (e.g. AppCustomAuthenticator):
> LoginFormAuthenticator

Choose a name for the controller class (e.g. SecurityController) [SecurityController]:
> SecurityController

Do you want to generate a '/logout' URL? (yes/no) [yes]:
> yes

created: src/Security/LoginFormAuthenticator.php
updated: config/packages/security.yaml
created: src/Controller/SecurityController.php
created: templates/security/login.html.twig
```

Fig. : Crédit de l’authentification

²¹ Doc : https://symfony.com/doc/current/security/form_login_setup.html

Comme précisé dans l'image ci-dessus, cette action va engendrer des créations de fichiers et la modification du fichiers de sécurité "security.yaml". Le fichier de sécurité permet de configurer les autorisations d'accès aux pages du site.

```
# Easy way to control access for large sections of your site
# Note: Only the *first* access control that matches will be used
access_control:
    - { path: ^/admin, roles: ROLE_ADMIN }
    # - { path: ^/profile, roles: ROLE_USER }
    - { path: ^/login$, roles: IS_AUTHENTICATED_ANONYMOUSLY }
```

```
firewalls:
    dev:
        pattern: ^/(_profiler|wdt)|css|images|js/
        security: false
    main:
        anonymous: true
        guard:
            authenticators:
                - App\Security\LoginFormAuthenticator
        logout:
            path: app_logout
            # where to redirect after logout
```

Fig. : Paramétrage du fichier "security.yaml"

Ci-dessus, je définis mes routes administrateurs avec une route (liées aux annotations de l'ORM) avec le chemin "^/admin", le lien vers le fichier permettant la vérification des informations pour la connexion dans la partie "authenticators" et le chemin lors de la déconnexion.

Le fichier "login.html.twig" créé représente l'espace de connexion où l'utilisateur va entrer les champs lui permettant de s'identifier.

Le fichier présent dans le dossier "src/Security", va quand à lui procéder aux vérifications des informations entrées par l'utilisateur ainsi qu'aux actions à exécuter en fonction de la réussite où de l'échec de l'opération. Il permet aussi de rediriger l'utilisateur lors de la déconnexion.

```
public function onAuthenticationSuccess(Request $request, TokenInterface $token, $providerKey)
{
    if ($targetPath = $this->getTargetPath($request->getSession(), $providerKey)) {
        return new RedirectResponse($targetPath);
    }

    // For example : return new RedirectResponse($this->urlGenerator->generate('some_route'));
    return new RedirectResponse($this->urlGenerator->generate( name: 'home_index'));
}

protected function getLoginUrl()
{
    return $this->urlGenerator->generate( name: 'app_login');
}
```

Fig. : Présentation des méthodes de vérification de connexion et déconnexion

La suite du processus pour définir les rôles va s'effectuer après la création d'un table contenant les utilisateurs avec la commande spécifique²² :

php bin/console make:user

Il y a ensuite un protocole bien particulier à suivre.

```
$ php bin/console make:user

The name of the security user class (e.g. User) [User]:
> User

Do you want to store user data in the database (via Doctrine)? (yes/no) [yes]:
> yes

Enter a property name that will be the unique "display" name for the user (e.g.
email, username, uuid [email]
> email

Does this app need to hash/check user passwords? (yes/no) [yes]:
> yes

created: src/Entity/User.php
created: src/Repository/UserRepository.php
updated: src/Entity/User.php
updated: config/packages/security.yaml
```

Fig. : Création de la table “User”

On peut constater que la création de cette classe va affecter le fichier “security.yaml” et comme pour les autres créations d’entités, cela va créer l’entité “User” et le “repository”, “UserRepository”. La modification de l’entité intervient à cause de la demande de “hashage” du mot de passe utilisateur.

²² Doc : <https://symfony.com/doc/current/security.html>

```
security:
    encoders:
        App\Entity\User:
            algorithm: auto

    # https://symfony.com/doc/current/security.html#where-do-users-come-from-user-providers
    providers:
        # used to reload user from session & other features (e.g. switch_user)
        app_user_provider:
            entity:
                class: App\Entity\User
                property: email
        # used to reload user from session & other features (e.g. switch_user)
```

Fig. : Paramètres supplémentaires ajoutés dans “security.yaml”

L'utilisation de la bibliothèque "Fixtures" est indispensable afin de faire des tests en créant l'utilisateur à vérifier.

Fig. : Exemples de création d'un administrateur

Pour limiter l'accès aux différentes pages du back-office pour un utilisateur non-authentifié, il faut restreindre ces derniers grâce aux annotations.

```
/** 
 * Class FileController
 * @package App\Controller
 * @Route("/admin/file", name="home_")
 */

class FileController extends AbstractController
```

Fig. : Exemple pour le contrôleur “FileController”

Toutefois, il ne faut pas oublier d'insérer le formulaire de connexion dans la vue de la/les page/s concernée/s. Ici, l'espace de connexion est prévue dans le “base.html.twig” contenant la base des éléments repris sur toutes les pages.



```
        , button...  
            <div class="dropdown-menu" aria-labelledby="dropdownMenuButton">  
                <a class="dropdown-item" href="{{ path('home_authentication') }}"><button type="button" class="dropdown-item">Home authentication</button>  
                <a class="dropdown-item" href="{{ path('home_identification') }}"><button type="button" class="dropdown-item">Home identification</button>  
            </div>  
        </div>  
    </ul>  
</div>  
<p>  
    {% if app.user %}  
        Bonjour {{ app.user.email }} !  
        <a href="{{ path('app_logout') }}> Se déconnecter</a>  
    {% else %}  
        <a href="{{ path('app_login') }}> Se connecter</a>  
    {% endif %}  
</p>
```

Fig. : Insertion du “Login / Logout” dans la vue

Il me reste cependant à déplacer les éléments présent ma page “index” dans une page dédiée afin d'utiliser mon index pour permettre uniquement la connexion au back-office sans laisser fuiter certaines informations contenus dans la page actuellement en place.

2. Front-end

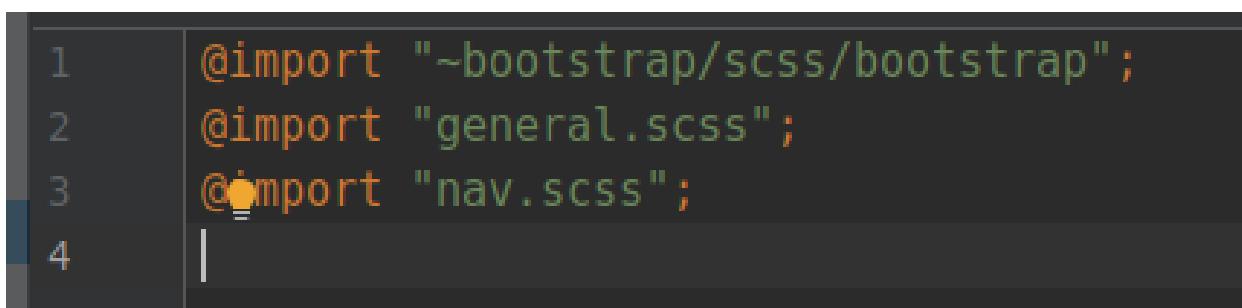
a. Le web mobile Bootstrap

Le cahier des charges du projet mentionnant le besoin d'adaptabilité rapide du site, Bootstrap permettait de répondre aisément à cette problématique. Cela m'a permis de créer une barre de navigation qui apparaît sous forme de "menu burger" et de disposer les informations de retour d'erreurs plus facilement.

Pour pouvoir utiliser Bootstrap dans le projet, j'ai dû installer la bibliothèque “popper.js” pour que l'affichage du site fonctionne.

b. Pré-processeur SCSS

J'ai choisi d'utiliser du SCSS pour permettre une identification rapide de la structure de la mise en forme et aider à la maintenabilité. La possibilité de définir des variables pour les éléments d'affichages récurrents garantit un gain de temps non négligeable pour les futurs développements du back-office.



```
1 @import "~bootstrap/scss/bootstrap";  
2 @import "general.scss";  
3 @import "nav.scss";  
4 |
```

Fig. 37 : Les variables SCSS

3. Tests et validations

Pour effectuer les tests de validation, notamment pour vérifier toutes les étapes du scan de fichier pour le parser et l'insérer en BDD, j'ai choisi l'utilisation de PHPUnit. En effet, pour assurer un développement sans forte régression, il est nécessaire d'effectuer des tests réguliers.

Dans mon cas, cela m'a permis de vérifier plusieurs de mes fonctions liées au scan du fichier entrant, son découpage et son insertion en base de données.

Ces différents tests m'ont permis de mettre en exergue les modifications à apporter aux fichiers de gestion des exceptions afin d'avoir un parser optimisé répondant au besoin du client²³.

²³ Voir chapitre 3.C.1.d.

```

mer 15:12
AllData [~/Stage/AllData] - .../tests/TxtparsingControllerTest.php
File Edit View Navigate Code Refactor Run Tools VCS Window Help
File tests TxtparsingControllerTest.php base.html.twig GlobalException.php IncludedException.php HomeController.php GlobalExceptionRepository.php HomeController.php PDOConnect.php PDOExceptionController.php PDOAuthentica Database
98
99     $this->assertEqual($expected, "WebException", $result['ExceptionClassName']);
100    $this->assertEqual($expected, "2018/05/03 04:22:38", $result['session']);
101    $this->assertEqual($expected, "4", $result['id']);
102    $this->assertEqual($expected, "4", $result['severity']);
103    $this->assertEqual($expected, "JSON Interface ERROR", $result['name']);
104    $this->assertEqual($expected, "Frontend web service endpoint got Exception : com.stripe.exception.PermissionException: The provided key 'sk_test_*****' does not have a");
105    $this->assertEqual($expected, "Impact unitaire", $result['code']);
106    $this->assertEqual($expected, "ANOMALIE CRITIQUE", $result['type']);
107    $this->assertEqual($expected, "An error occurred while retrieving data from Stripe call", $result['externalDesc']);
108    $this->assertEqual($expected, "False", $result['sendEmailToUser']);
109    $this->assertEqual($expected, "Fri May 04 15:02:52 CEST 2018", $result['timestamp']);
110    $this->assertEqual($expected, "-1", $result['userRef']);
111    $this->assertEqual($expected, "CrunchifyRESTService.StripeHooker", $result['javaClass']);
112    $this->assertEqual($expected, "sendhook", $result['javaMethod']);
113    $this->assertEqual($expected, "");
114    at CrunchifyRESTService.Stripehooker.sendhook(StripeHooker.java:347)
115    at sun.reflect.GeneratedMethodAccessorImpl.invoke(Unknown Source)
116    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
117    at java.lang.reflect.Method.invoke(Method.java:498)
118    at org.glassfish.jersey.server.model.internal.ResourceMethodInvocationHandlerFactory$1.invoke(ResourceMethodInvocationHandlerFactory.java:81)
119    at org.glassfish.jersey.server.model.internal.AbstractJavaResourceMethodDispatcher$1.run(AbstractJavaResourceMethodDispatcher.java:144)
120    at org.glassfish.jersey.server.model.internal.AbstractJavaResourceMethodDispatcher.invoke(AbstractJavaResourceMethodDispatcher.java:161)
121    at org.glassfish.jersey.server.model.internal.ResourceMethodDispatcherProvider$ResponseOutInvoker.invoke(ResourceMethodDispatcherProvider.java:99)
122    at org.glassfish.jersey.server.model.internal.ResourceMethodDispatcher.dispatch(AbstractJavaResourceMethodDispatcher.java:99)
123    at org.glassfish.jersey.server.model.internal.AbstractJavaResourceMethodDispatcher$1.run(AbstractJavaResourceMethodDispatcher.java:271)
124    at org.glassfish.jersey.server.model.ResourceMethodInvoker.invoke(ResourceMethodInvoker.java:389)
125    at org.glassfish.jersey.server.model.ResourceMethodInvoker.apply(ResourceMethodInvoker.java:347)
126    at org.glassfish.jersey.server.model.ResourceMethodInvoker.invoke(ResourceMethodInvoker.java:192)
127    at org.glassfish.jersey.server.ServerRuntime$2.run(ServerRuntime.java:326)
128    at org.glassfish.jersey.jersey.internal.Errors$1.call(Errors.java:321)
129    at org.glassfish.jersey.jersey.internal.Errors$1.call(Errors.java:267)
130    at org.glassfish.jersey.jersey.internal.Errors.process(Errors.java:315)
131    at org.glassfish.jersey.jersey.internal.Errors.process(Errors.java:297)
132    at org.glassfish.jersey.jersey.internal.Errors.process(Errors.java:267)
133
TxtparsingControllerTest
Terminal: Local (2) + Local (2) Event Log
autouza@autouza-ThinkPad-L450: [~/Stage/AllData] (debugCG *8) $ 20/11/2019
autouza@autouza-ThinkPad-L450: [~/Stage/AllData] (debugCG *8) $ 15:12 Module resolution rules from webpack.config.js are now used
autouza@autouza-ThinkPad-L450: [~/Stage/AllData] (debugCG *8) $ 15:12 Symfony Plugin: Enable the Symfony Plugin with auto.config
autouza@autouza-ThinkPad-L450: [~/Stage/AllData] (debugCG *8) $ 15:12 Event Log
autouza@autouza-ThinkPad-L450: [~/Stage/AllData] (debugCG *8) $ 16:10 LF UTF-8 4 spaces Git: debugCG 2 Event Log

```

Fig. 38 : Test du parseur (partie 1 sur 2)

Fig. 39 : Test du parseur (partie 2 sur 2)

Fig. 40 : Résultat du test

Point sur les modifications à effectuer

Après réflexion, je pense que le plus simple à mettre en place comme changement pour le parsing est de :

- Remplacer toute les occurrences "`=`" de l'explode par "`==>`"
- Remplacer les différents ";" internes par des ","
- Remplacer les différents "\$" internes par des "\\$"
- Dans DBException changer "`batabase`" en "`database`"

En partant sur ce principe, je pense que les anomalies seront réglées pour les exceptions.
En tout cas, cela se vérifie bien après avoir effectué 3 tests différents d'exceptions.

Fig. 41 : Extrait du fichier de synthèse

4. Déploiement

Cette partie est en cours pour le moment, le rendu demandé étant pour le 26 décembre. Ce chapitre sera plus détaillé dans la version remise le jour de l'oral (lundi 09 décembre 2019). Je vous remercie par avance de votre compréhension.

IV. Résultats

A. Fin du projet

À ce jour, le projet de site back-office répond aux besoins essentiels du client. Dès qu'une erreur est détectée par les exceptions JAVA, un fichier texte est généré. Le scan automatique qui s'exécute toutes les minutes va repérer cet ajout et commencer son travail.

Le fichier ".txt" est ensuite fractionné comme prévu, puis inséré dans les tables correspondantes. On retrouve dans le back-office, la page d'accueil qui regroupe les informations essentielles de toutes les exceptions, ainsi que les différentes sections qui sont identiques aux différentes tables. Ce visuel permet de retrouver, puis d'identifier aisément les informations contenues dans les erreurs. De plus, la possibilité de rechercher par un input textuel ou par des fonctions de tris permet de cibler une erreur, un type ou une date en particulier.

B. Axes d'améliorations de l'existant

La majorité des améliorations à apporter au projet concerne principalement le rendu "front" des informations sur le site. On peut par exemple imaginer la possibilité de cacher certains éléments non prioritaires dans une fenêtre à volet fermé comportant la possibilité de l agrandir.

C. Développements futur

Les besoins d'un client étant évolutifs dans le temps, ses demandes vont elles aussi aller croissantes.

Les exceptions JAVA permettent aussi de gérer les messages d'informations et se basent sur les mêmes informations que celles générées par les retours d'erreurs. En effet, les retours de type erreur ont un niveau de criticité différent des informations. Cela permettra dans le futur, de

développer le back-office en offrant la possibilité d'ajouter sur le même modèle de code, les informations liées aux autres sources de retours.

Si la plateforme doit gérer les classes “NetworkException” et “UserException”, il sera possible de reporter les champs commun à leurs classes filles dans leur propre table et d'ajouter facilement ces informations dans le site existant.

V. Conclusion

Le stage de fin d'étude en PHP / Symfony effectué chez Zento depuis fin octobre a été une expérience très enrichissante.

J'ai eu la possibilité de choisir sur quel langage développer le site back-office de la plateforme DataXchange. Cela m'a permis d'affirmer et d'approfondir mes connaissances sur ce langage et sur son framework Symfony.

Dans un premier temps, le fait d'avoir à gérer la conception du besoin en autonomie et d'apporter mes productions lors des réunions hebdomadaires, m'a apporté une meilleure vision des missions réalisées par le chef de projet, le Product Owner, le SCRUM master, le lead développeur. Ces différents métiers font parti intégrante des projets et m'ont ouvert le champs des possibles quant à l'évolution de ma carrière professionnelle et de mon métier de développeur.

J'ai également appris à travailler en autonomie et à gérer mon temps au regard du planning et en fonction de la charge de travaille hebdomadaire.

Dans un second temps, la partie développement m'a apporté une affirmation des connaissances ainsi qu'un approfondissement des fonctions PHP. J'ai aussi une meilleure compréhension de l'aide amenée par le framework Symfony mais aussi des limites atteintes avec le package “website / skeleton”, package pour moi de référence lors de la construction d'un projet web.

La méthode Agile utilisée tout au long du projet, m'a permis d'adapter les tâches à effectuer pour répondre aux impératifs de production.

Mon maître de stage, professeur à l'EPSI, s'est montré disponible et présent lorsque le besoin s'est fait sentir et il a pris le temps de m'expliquer les choix des modifications effectuées sur toutes les étapes du projet. Il s'est montré également à l'écoute de mes suggestions dans le développement du projet.

Ce stage a été très enrichissant grâce au rôle qui m'a été donné de développer en autonomie un véritable projet et d'approfondir mes connaissances techniques.

VI. Annexes

A. Table des illustrations

CHAPITRE 3

Chapitre A

- A.3. -> Fig. 1 : Schéma de hiérarchie des entités de gestion des exceptions.
- A.4. -> Fig. 2 : Schéma de hiérarchie des entités de gestion des exceptions JAVA
- A.4. -> Fig. 3 : Schéma des entités de gestion des exceptions avec clés étrangères
- A.4. -> Fig. 4 : Protocole pour créer les relations avec clés étrangères
- A.4. -> Fig. 5 : Détails des changements apportés dans la classe InternalInterface
- A.4. -> Fig. 6 : Détails de la classe JAVA NetworkException
- A.4. -> Fig. 7 : Détails de la classe JAVA InternalInterfaceException
- A.4. -> Fig. 8 : Détails de la classe JAVA WebException

Chapitre B

- B.3. -> Fig. 9 : Schéma de la table “Included Exception”
- B.4. -> Fig. 10 : Commande de la liste des différentes branches du projet
- B.4. -> Fig. 11 : Commande vers le repo distant
- B.4. -> Fig. 12 : Commande de vérification de suivi des fichiers
- B.4. -> Fig. 13 : Commande d'ajout de suivi des fichiers ciblé
- B.5. -> Fig. 14 : Commande d'ajout de suivi des fichiers ciblé
- B.5. -> Fig. 15 : Exemple d'interaction des IncludedException

Chapitre C

- C.1.b. -> Fig. 16 : Les Services
- C.1.c.i. -> Fig. 17 : configuration du chemin et des autorisations pour l'accès à la BDD
- C.1.c.i. -> Fig. 18 : Présentation d'un extrait de l'entité “GlobalException”
- C.1.c.i. -> Fig. 19 : Présentation de l'entité “GlobalExceptionController”
- C.1.c.i. -> Fig. 20 : Présentation de l'entité “GlobalExceptionRepository”
- C.1.c.i. -> Fig. 21 : Les étapes de création d'une entité
- C.1.c.i. -> Fig. 22 : Commande pour créer la table en base et l'insérer en base
- C.1.c.ii. -> Fig. 23 : Présentation de la classe DBInsertion
- C.1.c.ii. -> Fig. 24 : Fonction switch
- C.1.c.ii. -> Fig. 25 : Le cas WebException
- C.1.c.ii. -> Fig. 26 : Cas par défaut
- C.1.c.iii. -> Fig. 27 : Propriété du champ “insertdatetime”
- C.1.c.iii. -> Fig. 28 : Appel de la classe “@ORM\HasLifecycleCallbacks”
- C.1.c.iii. -> Fig. 29 : Fonction spécifique pour le setter le champ “insertdatetime”
- C.1.d. -> Fig. 30 : Fonction “parseContent”
- C.1.e. -> Fig. 31 : Utilisation de la classe Finder pour le scan de fichier
- C.1.f. -> Fig. 32 : Chemin contenant le fichier à traiter
- C.1.f. -> Fig. 33 : Fonction rename
- C.1.g. -> Fig. 34 : Configuration de la planification CRON
- C.1.g. -> Fig. 35 : Présentation de l'entité AutoScanCommand
- C.1.g. -> Fig. 36 : Présentation de la méthode ScanNow
- C.1.2.b. -> Fig. 37 : Les variables SCSS

- C.3. -> Fig. 38 : Test du parseur (partie 1 sur 2)
- C.3. -> Fig. 39 : Test du parseur (partie 2 sur 2)
- C.3. -> Fig. 40 : Résultat du test
- C.3. -> Fig. 41 : Extrait du fichier de synthèse

B. Sources des notes de bas de pages

1. MySQL - PHPMyAdmin : <https://dev.mysql.com/doc/mysql-shell/8.0/en/>
2. Object-Relational Mapping : Lié un objet à une base de données :
<https://www.Doctrine-project.org/projects/Doctrine-orm/en/2.6/tutorials/getting-started.html>
3. Plus spécifiquement en SCSS via le préprocesseur SAAS : <https://sass-lang.com/guide>
4. Search Engine Optimization : recherche d'optimisation des procédures :
<https://searchengineland.com/guide/what-is-seo>
5. Application Programming Interface : https://fr.wikipedia.org/wiki/Interface_de_programmation
6. Doc Exception JAVA : <https://stackify.com/specify-handle-exceptions-java/>
7. Doc Associations Doctrine : <https://symfony.com/doc/current/Doctrine/associations.html>
13. Doc Entités Doctrine : <https://symfony.com/doc/current/Doctrine.html>
14. Doc Évenements Doctrine : <https://symfony.com/doc/current/Doctrine/events.html>
17. Doc Finder Component : <https://symfony.com/doc/current/components/finder.html>
18. Doc EV : <https://www.php.net/manual/fr/book.ev.php>
19. Doc CRON : <https://doc.ubuntu-fr.org/cron>