**R**apport de **S**tage



**R**APPORT **F**INALDE **S**TAGE

RÉALISÉ PAR **R**omain **MILLAN**

STAGE ENCADRÉ PAR

**R**omain **LEBRETON**, **L**aurent **DARGENT** et **M**arek **NECESANY**

POUR L’OBTENTION DU **BUT I**nformatique

**A**NNÉE **U**NIVERSITAIRE 20**22**-20**23**

# Remerciements

Je tiens à remercier l'entreprise Spiriit pour m'avoir accueilli tout au long de ce stage. Mes remerciements vont également à Laurent DARGENT et Thomas DUHAMEL pour m'avoir guidé dans la gestion de projet. J'aimerais par ailleurs remercier précisément Marek NECESANY mais aussi toute l’équipe de chez Spiriit pour leur aide technique précieuse tout au long de mon séjour dans l’entreprise.

De plus, j'exprime ma gratitude envers Romain LEBRETON pour son accompagnement tout au long de mon stage. Enfin, je souhaite adresser mes remerciements à tous mes professeurs qui m'ont transmis les connaissances nécessaires. Leurs enseignements ont été essentiels pour ma formation.

# 

# Résumé

### Résumé en français

Ce document présente en détail les activités que j'ai effectuées tout au long de mon stage chez Spiriit. Pendant cette période, j'ai eu l'opportunité de me plonger dans le domaine du développement Symfony, où j'ai pu travailler sur une variété de projets de différentes envergures et types. Parmi ces projets, j'ai particulièrement apprécié l'occasion de contribuer à un projet open-source, de mettre en place un projet de test pour évaluer les fonctionnalités et de réaliser des projets sur mesure pour des clients.

Ce stage m'a permis de découvrir et d'explorer de nouvelles technologies passionnantes telles que Stimulus et TypeScript, ce qui a élargi mes compétences techniques. En outre, j'ai acquis une expérience précieuse en matière de développement back-end et aux traitements des données.

J'ai également développé des compétences essentielles en travaillant en équipe, en collaborant avec des collègues et en participant activement à la gestion de projets.

Les mots-clés qui pourraient définir mon stage sont : Symfony, développement back-end, apprentissage et travail en équipe.

### Abstract in english

This document provides a detailed overview of the activities I carried out throughout my internship at Spiriit. During this period, I had the opportunity to immerse myself in the field of Symfony development, where I worked on a variety of projects of different sizes and types. Among these projects, I particularly appreciated the chance to contribute to an open-source project, set up a testing project to evaluate functionalities, and deliver customized projects for clients.

This internship allowed me to discover and explore exciting new technologies such as Stimulus and TypeScript, which expanded my technical skills. Additionally, I gained valuable experience in backend development and data processing.

Furthermore, I developed essential skills by working in teams, collaborating with colleagues, and actively participating in project management.

The keywords that can be defined in my internship are: Symfony, backend, and project management.

# 

# Sommaire

[**Remerciements 1**](#_a1pyozq5x2an)

[**Résumé 2**](#_lkcfxyjizehb)

[Résumé en français 2](#_6o0l50q52sbx)

[Abstract in english 2](#_kyho47yzhddu)

[**Sommaire 4**](#_9mkctj8g1qp)

[**Table des figures 5**](#_dt5rkn57bm9)

[**Glossaire 6**](#_pbvwiiq2f8uc)

[**I. Présentation, contexte et enjeux du stage 1**](#_jzxrb51b6xjg)

[I.1. Présentation générale 1](#_gyouyvymt8sg)

[I.2. Dimension commerciale 1](#_cbqpzf3rp7iy)

[I.3. Enjeux du stage 2](#_3hiou3tmui21)

[**II. Analyse des missions 3**](#_xbddcscklefn)

[II.1. Composer Changelogs 3](#_dpnvadadgf9g)

[II.1.1. Analyse 3](#_88ollbbl63n2)

[II.1.2. Cahier des Charges 4](#_5rh2nls2m6e6)

[II.1.3. Rapport Technique 4](#_crnjtndvvfw1)

[II.2. Commande Vocale 6](#_xq16uevedmry)

[II.2.1. Analyse 6](#_ielvest4xh0y)

[II.2.2. Cahier des Charges 7](#_5y58eswrp5tv)

[II.2.3. Rapport Technique 7](#_p9tjrdfrzw5v)

[II.3. Gain 9](#_57nz8bcm0epu)

[II.3.1. Analyse 9](#_y9awbzfi94qo)

[II.3.2. Cahier des Charges 10](#_uxuznezciy28)

[II.3.3. Rapport Technique 10](#_citbw4812px5)

[**III. Méthodologie et Organisation des Projets 11**](#_8ztye0xezk01)

[III.1. Organisation du travail dans l’entreprise 11](#_5tcemhxs2q9w)

[III.2. Méthodes de développement et de travail que j’ai adoptées personnellement 12](#_mputhsi8nu1u)

[**Conclusion 13**](#_5yblcaxvcpkx)

[**Annexe 1**](#_5nt4n0w6bawd)

[**Bibliographie 12**](#_gxw3r3rfvmj1)

# 

# Table des figures

[Annexe n°1 : Composer Changelogs - Structuration du code source 1](#_7yhko2inhdk)

[Annexe n°2 : Composer Changelogs - Fichier de changelog sous format JSON et Texte 2](#_85xfaas7wk7o)

[Annexe n°3 : Composer Changelogs - Cahier des charges sous formes de ticket Jira 2](#_3n2kavmh5l45)

[Annexe n°4 : Composer Changelogs - Exemple de configuration 3](#_2mpqx7y6gcw7)

[Annexe n°5 : Composer Changelogs - Requête API avec HTTPClient de Symfony 3](#_oglcafik1w4j)

[Annexe n°6 : Composer Changelogs - Test de l’appel API à l’aide de data provider 4](#_st7qjnta25v6)

[Annexe n°7 : Composer Changelogs - Configuration PHP Stan 4](#_kpfwlzjmvmku)

[Annexe n°8 : Commande Vocale - Architecture DDD 5](#_ik8ksqntfeep)

[Annexe n°9 : Commande Vocale - Connexion utilisateur 6](#_ut2dv1wlgh3m)

[Annexe n°10 : Commande Vocale - Liste des paniers 7](#_1tzay3z0imhl)

[Annexe n°11 : Commande Vocale - Makefile et exécution de la commande make 8](#_f9hg5wxmkbz0)

[Annexe n°12 : Gain - Conteneur du projet 8](#_u2075wiornho)

[Annexe n°13 : Gain - Tickets à traiter 9](#_khv6jxutfjg1)

[Annexe n°14 : Gain - Exemple de configuration de mail 9](#_w52qpp7g268j)

[Annexe n°15 : Gestion de projet par Jira 10](#_ndgx05l4l5ch)

[Annexe n°16 : Processus GitFlow 11](#_tyd6sog0vnfi)

# 

# Glossaire

**API** : ‘Application Programming Interface’ est un ensemble de protocoles et d’outils pour créer des logiciels et des applications.

**Branche :** Ligne de développement indépendante, permet de travailler sur des fonctionnalités, des correctifs ou des expérimentations sans affecter la branche principale du projet.

**CI :** Intégration Continue, est une pratique courante qui consiste à la vérification du code intégré, ce qui permet la détection d’erreur automatique.

**CMS :** ‘Content Management System’ ou ‘Système de Gestion de Contenu’ est une plateforme utilisée pour gérer et organiser différents types de contenu tels que du texte, des vidéos, des images, etc., sur des sites web.

**Commit :** Un commit est une opération fondamentale pour le système Git. Il représente un enregistrement des modifications apportées à un ensemble de fichiers à un instant donné.

**Composer :** Composer est un gestionnaire de dépendances utilisé dans le développement de projets PHP.

**Changelog :** Un changelog est un résumé organisé des modifications effectuées entre différentes versions ou mises à jour du projet.

**Dataprovider :** Les dataprovider sont des combinaisons de données utilisées dans les tests unitaires.

**Docker :** Docker est un système de conteneurisation léger qui permet de lancer et d'exécuter facilement différents services, tels qu'une base de données ou un serveur web.

**Dispatcheur :** Un dispatcheur, fait référence à un composant qui gère la distribution d’événements ou de messages à travers un projet.

**Framework :** Un framework est un ensemble de bibliothèques, d'outils et de conventions de développement qui fournissent une structure et des fonctionnalités communes pour faciliter la création d'applications logicielles.

**Github :** GitHub est une plateforme open source de gestion de versions et de collaboration destinée aux développeurs de logiciels.

**Mock :** Un mock est un objet simulé utilisé dans les tests unitaires pour remplacer des objets réels ou des dépendances du système

**Open Source :** Open Source, ou source ouverte en français, est un modèle de développement de logiciel qui favorise la collaboration, la transparence et le partage libre des codes sources.

**Packagist :** Packagist est un dépôt en ligne de dépendances PHP. Il s'agit d'un service centralisé qui permet aux développeurs d'accéder à une vaste collection de packages prêts à l'emploi pour leurs projets.

**Proof of Concept (POC) :** Un Proof of Concept (POC) ou Preuve de Concept en français est une démonstration pratique ou à une réalisation concrète visant à valider la faisabilité et la viabilité d'une idée, d'une technologie ou d'un concept.

**TMA** : ‘Tierce Maintenance Applicative’ consiste en la maintenance et à l’ajout de nouvelles fonctionnalités sur des projets existants. Par exemple mettre en place des mises à jour de dépendances ou encore rajouter des éléments.

**JSON :** ‘JavaScript Object Notation’ est un format de données léger basé sur le stockage de données sous forme de clés/valeurs.

# Présentation, contexte et enjeux du stage

J'ai réalisé mon stage au sein de l'entreprise Spiriit à Montpellier. Cette première partie sera dédiée à la présentation de celle-ci.

Pour présenter efficacement l'entreprise, je vais diviser mon propos en trois points. Tout d'abord, une présentation générale, puis j’aborderai la dimension commerciale et enfin les enjeux de mon stage.

## Présentation générale

Spiriit est une entreprise nationale qui compte à elle seule 60 employés répartis dans trois bureaux situés à Montpellier, Chambéry et Lyon. Elle se divise en trois marchés à forte traction tels que le eCommerce, les applications personnalisées et le eBusiness.

Pour permettre aux employés des différents sites de se réunir, l'entreprise organise chaque lundi à 9 h des sessions appelées "catch-up" (*en visioconférence*). Pendant celles-ci, les employés peuvent parler de la vie de l’entreprise, des victoires effectuées lors de la semaine passée, mais aussi des innovations à venir et soumettre de nouvelles idées.

Il existe également chaque mois des "learn-up". Ce sont des séances d'environ 1 h 30 durant lesquelles une ou plusieurs personnes peuvent présenter des sujets qui leur tiennent à cœur ou partager leurs connaissances. Par exemple l'utilisation de git, les bonnes pratiques de code ou encore les outils de programmation comme GrumPHP, PHPStan, …..

## Dimension commerciale

Dans le but d'atteindre un chiffre d'affaires de 12 millions d'euros en 2027, Spiriit doit miser sur des clients réguliers, mais aussi être en capacité de fournir des prestations rapides. En effet, l'entreprise a un modèle économique basé sur 50 % de clients habituels et 50 % de prestations. Tout ceci permet une fidélisation et une communication nationale de l'entreprise.

En règle générale, le marché des applications personnalisées concerne les clients habituels qui utilisent le service de maintenance appelée TMA\*. Ce service permet la maintenance et l’ajout de nouvelles fonctionnalités sur leur site internet.

Contrairement au marché du sur-mesures, les marchés du eCommerce et du eBusiness, utilisent la plupart du temps des CMS\* tels que Drupal, WordPress ou encore PrestaShop. De plus, sur des projets eCommerce ou eBusiness, il est plus fréquent de fournir une TMA visant à l’augmentation du trafic ou encore à l’augmentation du chiffre d'affaires.

## Enjeux du stage

Durant ce stage, j'ai été chargé d'effectuer diverses tâches sur plusieurs projets, aussi bien de manière autonome qu'au sein d'une équipe. La majorité de ces missions étaient axées sur des projets personnalisés utilisant les technologies PHP, Symfony et Composer\*.

Au cours de ce stage, j'ai eu l'opportunité d'approfondir différents aspects du développement, de la gestion de projet et du travail en équipe. J'ai pu suivre l'évolution d'un projet en appliquant une approche agile, évaluer mes compétences en travail d'équipe et comprendre mon rôle dans plusieurs projets. Enfin, j'ai également acquis une solide expérience dans le développement back-end et l'utilisation d'outils de développement.

# 

# Analyse des missions

Durant mon stage au sein de l'entreprise Spiriit à Montpellier, j'ai eu l'opportunité de travailler sur plusieurs projets grâce à mes bases de connaissances acquises lors de mes cours à l’IUT et grâce à mon travail personnel.

## Composer Changelogs

En premier projet, Composer Changelogs\* est un plugin dédié à composer. L’utilité de composer réside dans sa capacité à simplifier la gestion des dépendances en s'assurant que toutes les bibliothèques requises sont correctement installées et compatibles entre elles. De plus, composer permet une compatibilité totale avec la machine en indiquant à celle-ci les versions autorisées pour le projet. En résumé, Composer permet d'optimiser l'organisation et la gestion des dépendances d'un projet PHP.

Ce projet a pour but de créer un résumé des dépendances mises à jour lors de l'exécution de la commande ‘composer update’ (*Commande qui permet de mettre à jour toutes les dépendances du projet courant*).

Ce projet a été mené grâce au code source de départ créé par Loïck PIERA[[1]](#footnote-0) , que nous avons utilisé et amélioré afin d’ajouter des fonctionnalités.

La finalité de ce projet se retrouve en open-source sur le GitHub de l’entreprise[[2]](#footnote-1) et sur Packagist[[3]](#footnote-2).

### Analyse

Concernant ce projet, l’architecture utilisée était une architecture basique et n’avait pas réellement de définition. J’ai donc veillé à bien séparer tous les fichiers dans des dossiers correspondants et ne pas mélanger ceux qui n’avaient pas pour but d’être ensemble (*cf.* [*Annexe n°1 : Composer Changelogs - Structuration du code source*](#_7yhko2inhdk)). Tout d'abord, le code source et les tests ont été organisés dans deux dossiers distincts. De plus, le projet a été structuré en fonction des fonctionnalités, de sorte que les classes correspondantes y soient regroupées. Par exemple, le dossier "Config" contient exclusivement les classes liées à la gestion de la configuration du projet, tandis que le dossier "Outputter" regroupe les classes chargées des sorties du projet, telles que la création de fichiers. Cette approche permet une meilleure organisation et une plus grande clarté dans la structure du code.

Lorsque j’ai récupéré le projet, il existait déjà une base de fonctionnalités. Celle-ci permettait d’avoir le résumé des dépendances mises à jour lors de l’exécution de la commande, mais aussi la création d’un fichier de changelogs en deux formats possible (*en JSON\* ou en texte brut*) (*cf.* [*Annexe n°2 : Composer Changelogs - Fichier de changelog sous format JSON et Texte*](#_85xfaas7wk7o)). Mon but étant d’améliorer cette base et ajouter la fonctionnalité permettant d’envoyer le changelog sur un point API\* permettant par la suite d’envoyer les informations de celui-ci sur d'autres plateformes. Enfin, rendre le projet open source\* pour le partager à la communauté des développeurs PHP.

### Cahier des Charges

La première étape de ce projet consistait à élaborer un cahier des charges détaillé. Pour cela, Spiriit utilise l'outil Jira, qui permet de créer des demandes spécifiques pour chaque projet (*cf.* [*Annexe n°3 : Composer Changelogs - Cahier des charges sous formes de ticket Jira*](#_3n2kavmh5l45)). Chaque tâche ainsi définie a contribué à l'élaboration complète du cahier des charges de ce projet, en décrivant précisément les objectifs, la fonctionnalité requise et les exigences à respecter.

Le cahier des charges était structuré en trois tâches clés. Tout d'abord, il était essentiel de vérifier, d’une part, la qualité du code produit par Loïck PIERA, et d’autre part, le code précédemment produit par un autre développeur de Spiriit.

Dans un second temps, il fallait ajouter la nouvelle fonctionnalité souhaitée : pouvoir envoyer les changelogs auprès d’une API qui se chargera de transmettre les informations vers des plateformes telles que Slack ou encore Confluence.

Enfin, dans le but de finaliser ce projet qui visait à devenir open-source, j'ai consacré du temps à améliorer la lisibilité du code, à créer un readme correcte et à effectuer des tests pour valider le bon fonctionnement du code source. Cela a permis de vérifier sa qualité et sa stabilité avant sa mise à disposition en open-source.

### Rapport Technique

Pour réaliser ma partie du projet, j'ai dû créer une requête API vers une URL spécifique. Dans un premier temps, j'ai récupéré l'URL à partir de la configuration composer du projet, celui-ci étant contenu dans le fichier ‘composer.json’ (*cf.*[*Annexe n°4 : Composer Changelogs - Exemple de configuration*](#_2mpqx7y6gcw7)). Toute la configuration a été mise en place grâce à des classes de configuration déjà existantes.

Une fois l'URL récupérée, j'ai utilisé l’objet HTTPClientInterface pour créer une instance de la classe HTTPClient. Grâce à cette instance et à la méthode "request", j'effectue un appel vers le point d'API correspondant en utilisant l'URL fournie par l'utilisateur. Dans cette requête, j'inclus dans le corps, au format JSON, le fichier de changelog actuel (*cf.* [*Annexe n°5 : Composer Changelogs - Requête API avec HTTPClient de Symfony*](#_oglcafik1w4j)).

Après avoir créé un appel vers l'API, j'ai vérifié le bon fonctionnement des tests déjà présents sur le projet. De plus, j’ai ajouté des tests sur la nouvelle fonctionnalité d’appel API en créant des tests d’intégration pour la nouvelle fonctionnalité que je venais de mettre en place.

Pour effectuer ceci, j'ai utilisé la bibliothèque PHPUnit ainsi que les objets MockHTTPClient et MockResponse fournis par la composante HTTPClient de Symfony afin de réaliser des tests de manière optimale.

Étant donné que ces derniers doivent être très précis et ne pas dépendre de contraintes externes telles qu'une API, j'ai choisi d'utiliser les classes de Mocking. Par exemple, il existe MockHTTPClient et MockResponse qui permettent de simuler les réponses d'un appel à HTTPClient (*cf.* [*Annexe n°6 : Composer Changelogs - Test de l’appel API à l’aide de data provider*](#_st7qjnta25v6)). Tout ceci permet de tester uniquement le code créé et non les valeurs renvoyées par une entité externe au projet.

De plus, afin d'éviter de répéter du code de test, j'ai utilisé les data provider\* qui permettent d'utiliser une seule fonction avec plusieurs options.

Finalement, j'ai saisi l'opportunité de mettre en place des outils d'aide au développement. Par exemple, j'ai utilisé PHP CS Fixer et Easy Coding Standard pour détecter et corriger les défauts de qualité de code et notamment les standard de code de PHP nommée PSR. J'ai également utilisé PHPStan, qui effectue une analyse approfondie de la qualité du code, en utilisant une configuration personnalisée pour définir le niveau de qualité attendu (*entre 0 à 9*) (*cf.* [*Annexe n°7 : Composer Changelogs - Configuration PHP Stan*](#_kpfwlzjmvmku)).

Dans le même ordre d'idées, j'ai utilisé GrumPHP, un outil puissant qui exécute des commandes spécifiques et vérifie leur exécution lors des commits\* git. Sur ce projet, je l’ai utilisé pour exécuter PHPStan et PHP CS Fixer afin de garantir une qualité de code optimale lors de chaque commit.

Pour conclure, cette application a été déployée sur des projets de l’entreprise pour un suivi en temps réel des modifications misent en place. Celle-ci a aussi été publiée en open-source pour tous les développeurs. De plus, grâce à ce projet, j’ai pu approfondir ma connaissance sur l’apprentissage critique 1.2 nommée “Partir des exigences et aller jusqu’à une application complète”.

## Commande Vocale

Baresto est un réseau de restaurateurs qui bénéficient de négociations tarifaires avantageuses avec leurs fournisseurs pour le bénéfice de leurs adhérents.

Depuis maintenant dix ans, l'entreprise fait appel à Spiriit pour concevoir et développer son site internet. En plus de leur site traditionnel, Baresto se démarque en mettant en place des idées novatrices. Parmi celles-ci, la possibilité de passer des commandes par la voix grâce à l'intelligence artificielle directement depuis un téléphone a été mise en avant. Pour démontrer son bon fonctionnement, j’avais pour mission de réaliser un Proof of Concept (POC)\* fonctionnel pour le client.

Ce projet, validé par Baresto, a également été récompensé lors de la soirée “Les Cas d'Or du Digital" en recevant : “*le prix de l'innovation*” et “*le prix conversationnel*”.

### Analyse

Ce projet repose sur l'utilisation de Symfony en version 6.3 (*pour toute la logique du système*) et du framework\* JavaScript Stimulus (*pour l’interface utilisateur*), qui est adopté par beaucoup de projets Symfony. De plus, il intègre l'API d'OpenAI, les créateurs de ChatGPT, afin de tirer parti de leur intelligence artificielle.

En ce qui concerne l'architecture du projet, j'ai utilisé le Domain Driven Design (DDD), une approche qui consiste à diviser le code en fonction des différents contextes métier (*cf.* [*Annexe n°8 : Commande Vocale - Architecture DDD*](#_ik8ksqntfeep)). Cette architecture est couramment utilisée dans les projets de l'entreprise, offrant une meilleure modularité et extensibilité.

La fonctionnalité centrale de ce projet consiste à permettre aux utilisateurs de passer des commandes sur leur téléphone en utilisant leur voix. Ils pourront simplement dicter leur liste de courses, qui ajoutera automatiquement les éléments à leur panier. Toutefois, en plus de cette fonctionnalité principale, il est important de conserver celles de base telles que la modification ou la suppression d'articles en cas d'erreur.

Pour réaliser ce nouveau POC, j'ai commencé par récupérer un précédent POC plus basique réalisé par un autre développeur, qui m'a servi de point de départ.

### Cahier des Charges

Ce projet n'a pas été spécifiquement défini par un cahier des charges formel. Les exigences relatives au projet ont été discutées lors d’ateliers tout au long de son déroulement. Cependant, ces demandes ont été regroupées en deux principales parties.

Dans la première partie du projet, il m’a fallu mettre en place un mécanisme permettant d'enregistrer la voix de l'utilisateur ainsi que ses identifiants pour accéder aux préférences utilisateur dans l'outil Baresto. Une fois ces données collectées, il était nécessaire de les traiter et d'afficher les éléments demandés de manière appropriée.

Dans la seconde partie du projet, il fallait ajouter des fonctionnalités telles que la possibilité de modifier un article ou de le supprimer. Enfin, il était nécessaire de stocker toutes les données.

Malgré l'absence d'un cahier des charges formel, ces objectifs ont été définis et atteints grâce à une communication continue entre les parties impliquées et une compréhension claire des besoins et des attentes du projet.

### Rapport Technique

Ce rapport technique est divisé en deux parties. Dans la première partie, je me pencherai sur la réalisation de l'application et l'intégration de l'API OpenAI. Enfin, la seconde et dernière partie sera consacrée aux données.

Premièrement, j’ai pu effectuer la réalisation de l'application en utilisant les technologies Symfony (Framework de PHP) et le framework JavaScript Stimulus, qui sont souvent combinés ensemble. Cependant, n’ayant jamais utilisé Stimulus auparavant, j'ai dû consacrer du temps à m'adapter, à comprendre et à maîtriser ce framework, notamment grâce au site SymfonyCast[[4]](#footnote-3) et à la lecture du code existant.

J'ai utilisé Stimulus pour capturer la voix de l'utilisateur et l'enregistrer dans des fichiers audio à l'aide d'un formulaire masqué.

Suite à cela, j'ai utilisé l’objet HTTPClientInterface pour pouvoir appeler des points API tel que le point API d’OpenAI pour le traitement des fichiers audios, mais aussi les points API de Baresto (*cf.* [*Annexe n°9 : Commande Vocale - Connexion utilisateur*](#_ut2dv1wlgh3m)).

Cependant, un problème majeur s'est posé en termes de performances. En effet, l'API d'OpenAI étant assez lente (*plusieurs dizaines de secondes*), cela se répercute sur l’application globalement. À ce stade, je n'ai trouvé aucune solution pour ce problème autre que d'informer l'utilisateur que l'application est en cours de chargement. Étant donné que c'est un projet POC, la recherche d'optimisation sera effectuée une fois que l'application sera validée par le client et les bêta testeurs. Cette suite pourrait alors être un lien avec “Analyser et optimiser des applications” (*Apprentissage Critique 2.3*).

Enfin, d’autres appels à l’API se font pour récupérer les articles probablement dans les favoris de l’utilisateur connecté. Suite à cela, l’application crée une liste de tous ces articles et les renvoie au format JSON (*cf.* [*Annexe n°10 : Commande Vocale - Liste des paniers*](#_1tzay3z0imhl)). Finalement, la réalisation de ce POC est en lien avec l’apprentissage critique 1.1 “Développer des applications informatiques simples”.

Enfin, je vais aborder le stockage des données. Dans la première version du POC, j'ai utilisé le service MariaDB pour stocker les informations telles que les articles et les commandes. Cependant, cette approche nécessitait de traiter les informations à plusieurs reprises. Cette approche entraine un stockage inutile des données et des possibilités d’erreur lors du traitement des données.

Étant donné que le site de Baresto dispose de nombreux points d'API permettant de gérer les données, dans la seconde version du POC, j’ai alors stocké les données directement sur l’outil par des appels API.

Cela présente l'avantage de ne traiter les informations qu'une seule fois et de pouvoir accéder aux informations d’une part dans l'application de commande vocale et d’autre part sur l’outil Baresto. Cette approche centralisée facilite la gestion des données et assure une synchronisation cohérente entre les deux plateformes, en rapport avec l’apprentissage critique “Optimiser une base de données, interagir avec une application et mettre en œuvre la sécurité” (*Apprentissage Critique* *4.2*).

Enfin, j'ai également mis en place un fichier Makefile, utilisé comme raccourci dans le développement web. Étant donné qu'il existe de nombreuses commandes à exécuter, telles que par exemple les commandes pour gérer les bases de données, vider le cache, lancer ou arrêter les projets. Certaines de ces commandes étant longues, les fichiers Makefile et les commandes make permettent d’exécuter une ou une série de commandes consécutive (*cf.* [*Annexe n°11 : Commande Vocale - Makefile et exécution de la commande make*](#_f9hg5wxmkbz0)). Cette approche permet de simplifier et d'automatiser l'exécution de tâches courantes. Grâce au Makefile, il suffit d'exécuter une seule commande make pour effectuer une ou plusieurs actions en cascade.

## Gain

Gain, est un réseau de campings qui bénéficie de négociations tarifaires avantageuses avec ses fournisseurs, afin de pouvoir proposer des offres très compétitives à ses adhérents. Ce projet auquel j'ai participé lors de mon stage était une mission de TMA, ce qui impliquait d'assurer la maintenance et d'ajouter de nouvelles fonctionnalités au projet existant.

À l'origine, Gain et Baresto faisaient partie d'un même projet, qui s’est scindé en deux par la suite. Contrairement à Baresto, Gain n'a pas bénéficié des mises à jour et présente donc une dette technique conséquente.

Je tiens également à souligner que ce projet a débuté vers la fin de mon stage, ce qui explique qu'il contienne moins de contenu que les précédents sur lesquels j'ai travaillé. Cependant, malgré la durée plus courte, j'ai tout de même pu contribuer activement à sa réalisation en apportant mes compétences et en travaillant en collaboration avec l'équipe pour atteindre les objectifs fixés.

### Analyse

Ce projet est divisé en plusieurs parties, nécessitant la mise en place d'un environnement Docker\*. Différents conteneurs sont configurés tels que pour l'interface utilisateur, Angular 1 et pour la logique métier et connexion à la base de données, Symfony 5.4, le tout fonctionnant avec PHP 7.4. De plus, des conteneurs tels que MailDev et RabbitMq sont mis en place pour intercepter les e-mails et gérer les messages de manière asynchrone (*cf.* [*Annexe n°12 : Gain - Conteneur du projet*](#_u2075wiornho)).

L'application dispose déjà d'un large éventail de fonctionnalités, telles que l'achat d'objets et la gestion des fournisseurs. Cependant, le client GAIN a demandé la création d'un nouveau portail permettant aux fournisseurs de se connecter. Étant donné la dette technique existante, il a été décidé de développer ce portail avec Symfony. J'ai personnellement contribué à l'ajout de fonctionnalités sur le portail fournisseur, notamment la gestion des contacts associés à un fournisseur.

### Cahier des Charges

Tout comme le plugin Composer, ce projet était régi par un cahier des charges défini sur le contenu des tickets créés sous Jira. Ayant travaillé sur une durée limitée, je n’ai pu traiter que deux tickets (*cf.* [*Annexe n°13 : Gain - Tickets à traiter*](#_khv6jxutfjg1)). Le premier consistait à mettre en place la gestion des contacts pour un fournisseur et le second visait à ajouter un système d'export de fichiers adhérent pour les fournisseurs.

###### 

Concernant le premier ticket, ma tâche consistait à mettre en place les fonctionnalités d'ajout, de modification et de suppression des contacts d'un fournisseur. En plus de ces actions, il était également nécessaire d'envoyer un e-mail à chaque exécution d'une de ces actions. Pour accomplir cette tâche, j'ai dû utiliser le travail réalisé précédemment par d'autres développeurs et l'adapter au code existant afin d'ajouter ces fonctionnalités.

Pour le deuxième ticket, j'avais pour mission d'exporter un fichier contenant tous les adhérents et l'envoyer par e-mail. Cette fonctionnalité était déjà présente dans un autre contexte de ce projet, j'ai donc pu utiliser le travail effectué par un autre développeur. Je l’ai alors adapté une fois de plus à ma situation.

### Rapport Technique

Pour accomplir ces deux tâches, j'ai dû mettre en place l'envoi d'e-mails. Étant donné que le projet existait depuis un certain temps, un système d'envoi d'e-mails avait déjà été mis en place. Pour envoyer un e-mail, il suffisait de créer un code similaire à celui-ci (*cf.* [*Annexe n°14 : Gain - Exemple de configuration de mail*](#_w52qpp7g268j)). Il suffisait de spécifier le nom et l'adresse e-mail de l'expéditeur, le sujet du courrier ainsi que le modèle de contenu. Une fois cela effectué, il faut alors appeler cette fonction à l'aide d'un dispatcheur de messages ou d’événement, enfin, l'e-mail est envoyé.

# 

# Méthodologie et Organisation des Projets

## Organisation du travail dans l’entreprise

Concernant l'organisation du travail, Spiriit utilise la suite Atlassian ainsi que GitLab pour la gestion de ses projets.

En effet, l'outil Jira de la suite Atlassian est particulièrement puissant pour la gestion de projets Agile (*cf.* [*Annexe n°15 : Gestion de projet par Jira*](#_ndgx05l4l5ch)), ce qui est largement utilisé au sein de l'entreprise. Cela permet de diviser le travail en sprints. Au début de chaque sprint, une réunion est organisée afin de décider des tickets à traiter, d'attribuer du temps à chaque ticket et de créer les descriptions détaillées pour que les développeurs puissent mieux les comprendre.

Pour les projets plus importants, des réunions quotidiennes sont tenues, tandis que pour les projets de plus petite envergure, ces réunions sont plus espacées. Grâce à ces outils, Spiriit gère et facilite la collaboration au sein de l'équipe.

En ce qui concerne la gestion du code, celui-ci est maintenu sur GitLab en utilisant l'infrastructure GitFlow (*cf.* [*Annexe n°16 : Processus GitFlow*](#_tyd6sog0vnfi)), qui divise le code en plusieurs branches\*. La branche "master" ou "main" est utilisée pour le code de production, c'est-à-dire le code qui est actuellement en cours d'exécution sur le serveur final. La branche "develop" est-elle utilisée pour le code en préproduction.

Chaque nouveau ticket est créé à partir de la branche "develop". Une fois le ticket terminé, les deux branches sont fusionnées sur "develop" via une demande de fusion (nommée Merge Request ou MR) qui sont examinés par les autres développeurs. En plus de cette structure basée sur GitFlow, des CI\* (Intégration Continue) ont été mises en place pour effectuer les tests à chaque fois qu'un commit est effectué. De plus, des CD (Déploiement Continu) ont été mis en place pour permettre un déploiement automatique sur les serveurs de préproduction et de production. Les CD ne sont disponibles que pour les branches "develop" et "master".

Grâce à tous ces outils, j’ai pu mettre en place les apprentissages critiques “Identifier les besoins métiers des clients et des utilisateurs“ et “appliquer une démarche de suivi de projet en fonction des besoins métier des clients et utilisateur” du référentiel“.

## Méthodes de développement et de travail que j’ai adoptées personnellement

Tout au long de mon stage, j'ai eu l'opportunité de travailler sur plusieurs projets de différentes envergures. Pour la plupart de ces projets, j'ai pu faire preuve d'autonomie dans l'exécution des tâches qui m'ont été assignées. J’avais également la possibilité de solliciter l'aide des autres membres de l'équipe en cas de difficultés rencontrées lors de l'accomplissement d'une tâche. En particulier, j'ai pu compter sur le soutien de Marek NECESANY, qui m’a apporté beaucoup d’aide tout du long de mon stage.

J'ai durant mon stage réalisé de nombreuses recherches pour accomplir mes tâches, en me référant principalement aux documentations, notamment celle de Symfony[[5]](#footnote-4), ce qui m’a permis de travailler sur l’apprentissage critique “Identifier ses aptitudes pour travailler dans une équipe” (*Apprentissage Critique 5.1*).

Grâce à la confiance accordée par l'équipe, j'ai ainsi pu participer activement en partageant mes idées, en proposant des solutions et en apportant mon aide aux autres membres de l'équipe dans l'exécution de leurs tâches. Grâce à ces personnes, j'ai pu développer l’apprentissage critique “Situer son rôle et ses missions au sein d’une équipe informatique” (*Apprentissage Critique 6.2*).

Ce stage m'a permis de développer mes compétences, d'approfondir ma compréhension des projets et d'enrichir mon expérience professionnelle. De plus, les échanges fructueux et ma contribution active m'ont donné l'occasion de mettre en pratique mes compétences et d'apporter une valeur ajoutée aux différents projets sur lesquels j'ai travaillé. Cela a par ailleurs renforcé ma confiance en moi et ma capacité à prendre des initiatives au sein d'une équipe de travail.

# Conclusion

Finalement, j'ai pu travailler sur une grande variété de projets, allant de ceux en cours de développement aux projets de test. Mon travail au sein de l'entreprise a été utilisé pour des clients, comme c'était le cas pour le projet GAIN ou encore la commande Vocale de Baresto. Certains ont été mis en production, tel celui de gestion des changelogs pour Composer qui a été publié en open-source. Enfin, d'autres projets de test ont été repris comme la fonctionnalité de commande vocale pour Baresto.

Au cours de mon stage, j'ai eu une occasion formidable d'acquérir une quantité considérable de connaissances dans divers domaines, tels que le développement, la vie en entreprise et la gestion de projet.

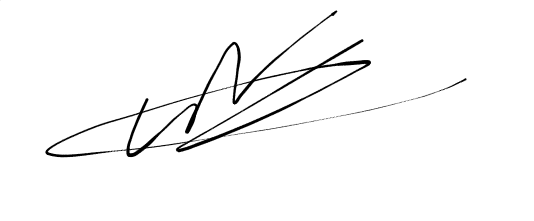
J'ai eu la chance de participer à la gestion de projets, notamment en estimant des tickets et en contribuant activement aux réunions de suivi, en particulier sur le projet GAIN.

En parallèle, j'ai énormément progressé sur le plan technique. Étant encore novice dans le développement web, j'ai pu assimiler de nombreuses informations précieuses, particulièrement sur la mise en place de CI/CD, ainsi que l'utilisation de frameworks tels que Stimulus et Symfony. J'ai également approfondi mes compétences en PHP, en apprenant surtout les normes de codage et les concepts.

Finalement, ce stage m’a permis d’étendre mes connaissances. De plus, j’ai particulièrement travaillé sur les compétences “Réaliser un développement d’application“ (*CE 1*), “Optimiser des applications informatiques” (*CE 2*) et enfin “Travailler dans une équipe informatique” (*CE 6*). Je suis reconnaissant d'avoir pu bénéficier d'une telle expérience enrichissante et je suis prêt à mettre en pratique tout ce que j'ai appris dans le futur.

Vu le 14/06/2023,

Laurent Dargent



# Annexe

###### 

###### Annexe n°1 : Composer Changelogs - Structuration du code source

###### 

###### 

###### Annexe n°2 : Composer Changelogs - Fichier de changelog sous format JSON et Texte

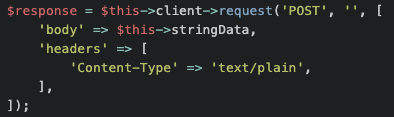
###### 

###### Annexe n°3 : Composer Changelogs - Cahier des charges sous formes de ticket Jira

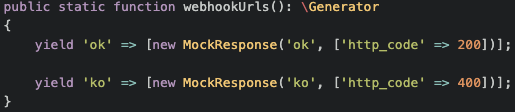
###### 

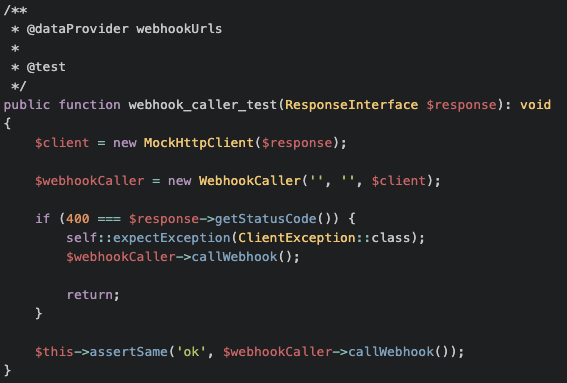
###### Annexe n°4 : Composer Changelogs - Exemple de configuration



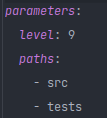


###### Annexe n°5 : Composer Changelogs - Requête API avec HTTPClient de Symfony





###### Annexe n°6 : Composer Changelogs - Test de l’appel API à l’aide de data provider



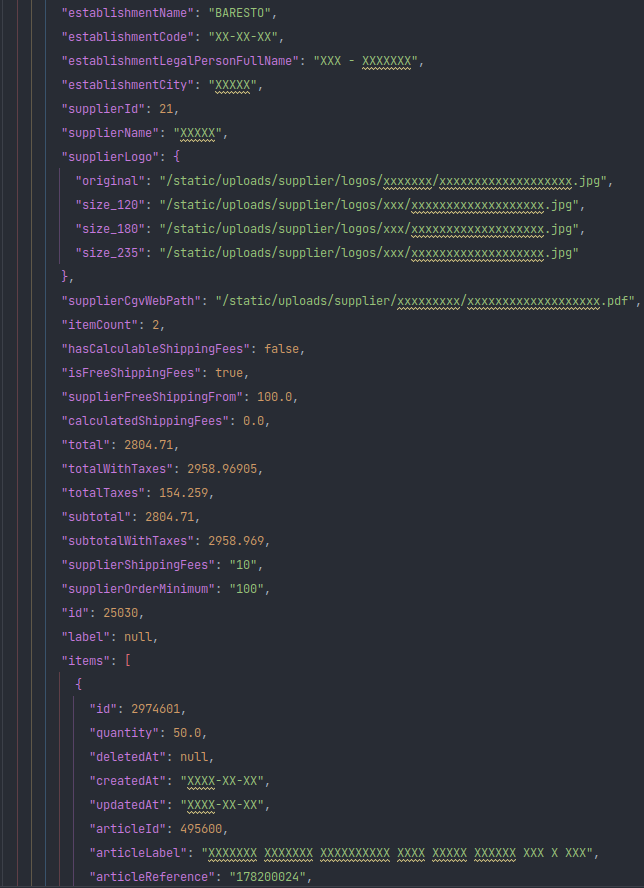
###### Annexe n°7 : Composer Changelogs - Configuration PHP Stan

###### 

###### Annexe n°8 : Commande Vocale - Architecture DDD



###### Annexe n°9 : Commande Vocale - Connexion utilisateur



###### Annexe n°10 : Commande Vocale - Liste des paniers

###### 

###### Annexe n°11 : Commande Vocale - Makefile et exécution de la commande make

###### 

###### Annexe n°12 : Gain - Conteneur du projet

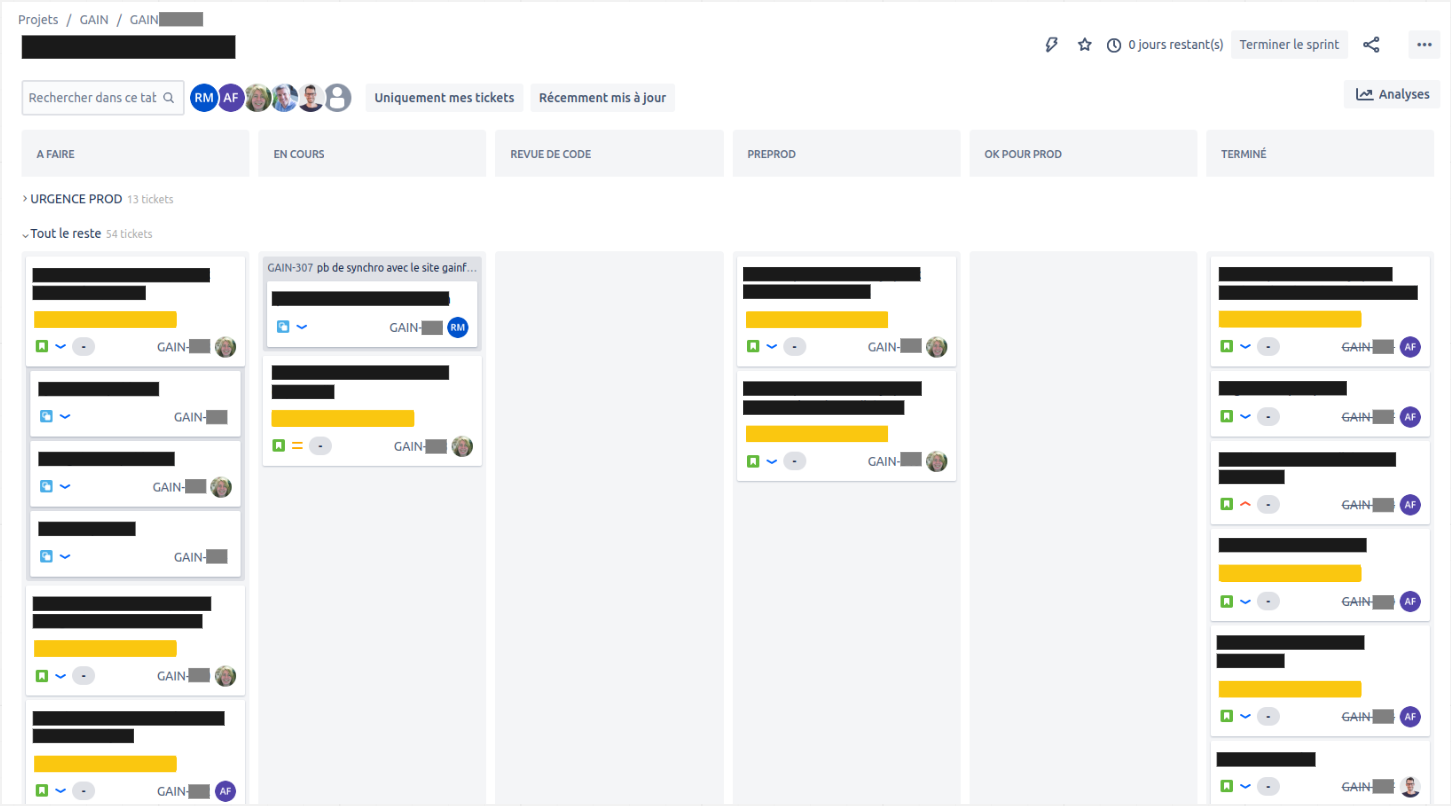
###### 

###### 

###### Annexe n°13 : Gain - Tickets à traiter

###### 

###### Annexe n°14 : Gain - Exemple de configuration de mail



###### Annexe n°15 : Gestion de projet par Jira

###### 

###### Annexe n°16 : Processus GitFlow

# 

# Bibliographie

[1]

Atlassian, « Qu’est-ce que Git ? | Atlassian Git Tutorial », *Atlassian*. [Online]. Disponible sur: <https://www.atlassian.com/fr/git/tutorials/what-is-git>.

[2]

« Qu’est-ce que Docker ? | AWS », *Amazon Web Services, Inc.* [Online]. Disponible sur: <https://aws.amazon.com/fr/docker/>.

[3]

« PSR : des standards de programmation en PHP pour garantir la qualité des développements Web ». [Online]. Disponible sur: <https://www.adimeo.com/blog-technique/psr-les-standards-de-programmation-en-php>.

[4]

« Qu’est-ce que le Test Unitaire - ISIT ». [Online]. Disponible sur: <https://www.isit.fr/fr/article/qu-est-ce-que-le-test-unitaire.php>.

[5]

« La TMA (Tierce Maintenance Applicative), qu’est-ce que c’est ? », *IPI école d’informatique*. [Online]. Disponible sur: <https://www.ipi-ecoles.com/tma/>.

[6]

« Mocks, Fakes, Stubs, Dummy et Spy — Faire la différence ». [Online]. Disponible sur: <https://knplabs.com/fr/blog/mocks-fakes-stubs-dummy-et-spy-faire-la-difference/>.

[7]

« JSON ». [Online]. Disponible sur: <https://www.json.org/json-en.html>.

[8]

« What is an API? » [Online]. Disponible sur: <https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces>.

[9]

« Introduction au Domain Driven Design », *Les Dieux Du Code*. [Online]. Disponible sur: <https://lesdieuxducode.com/blog/2019/7/introduction-au-domain-driven-design>.

1. Repository du code source de Loïc PIERA - [pyrech/composer-changelogs: :notebook](https://github.com/pyrech/composer-changelogs) [↑](#footnote-ref-0)
2. Repository du projet final, ouvert en open source - [SpiriitLabs/composer-write-changelogs](https://github.com/SpiriitLabs/composer-write-changelogs) [↑](#footnote-ref-1)
3. Lien vers le packagist du projet - [spiriitlabs/composer-write-changelogs - Packagist](https://packagist.org/packages/spiriitlabs/composer-write-changelogs) [↑](#footnote-ref-2)
4. Site de tutoriel pour PHP et Symfony - [SymfonyCasts](https://symfonycasts.com/) [↑](#footnote-ref-3)
5. Documentation symfony - [*https://symfony.com/doc/current/index.html*](https://symfony.com/doc/current/index.html) [↑](#footnote-ref-4)