



efrei

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Rapport d'activités

INGÉ1-APP

ÉLÈVE

Nom : SOLANKI

Prénom : Priyank

Filière : LSI

SUJET

DÉVELOPPEUR BACKEND EN SYMFONY

ENTREPRISE

Nom : Districall

Adresse : 183 Av. Achille Peretti, 92200 Neuilly-sur-Seine, France

DATE DU CONTRAT D'APPRENTISSAGE

du : 26/12/2024

au : 02/07/2027

ENCADRANTS

- Tuteur entreprise : Antoine LAMBERT
- Tuteur pédagogique : Alain HAROUN
- Chargée de mission Career Center : Morgane GOLLER

PUBLICATION DU RAPPORT D'ACTIVITÉS

Le Tuteur Entreprise : Antoine LAMBERT autorise l'apprenti à publier le rapport final sur le livret d'alternance dédié. Dans le cas d'une clause de confidentialité, seule cette page de garde sera déposée et le Tuteur Pédagogique : Alain HAROUN atteste du travail réalisé.

Signatures

Mots clés

Sommaire

Introduction.....	2
Remerciements.....	3
Abstract.....	4
I - Présentation de l'entreprise.....	5
1. Présentation de l'entreprise.....	5
2. Organigramme & environnement professionnel.....	6
3. L'environnement externe.....	8
II - Présentation de l'équipe.....	9
1. Contexte métier & Structure organisationnelle.....	9
2. Contexte technique.....	10
3. Organisation du travail.....	12
4. Présentation du poste et de la mission.....	13
III - Missions : Automatisation de tâches, développement de fonctionnalités & résolution de bugs.....	14
1. Contexte & Existant.....	14
2. Description des outils et de l'environnement technique.....	18
3. Présentation des missions et analyse de ses réalisations.....	25
1. Automatisation de tâches.....	25
2. Développement de nouvelles fonctionnalités.....	29
3. Résolution de bugs.....	31
4. Indicateurs de performances.....	34
1. Automatisation de tâches.....	34
2. Développement de nouvelles fonctionnalités.....	34
3. Résolution de bugs.....	34
IV - Bilan des réalisations : apports, difficultés, échecs et réussites.....	35
1. Identification du profil du poste et des compétences acquises à travers la mission d'apprentissage.....	35
2. Tendances et facteurs d'évolutions en entreprise.....	36
Table des figures.....	37
Lexique.....	38

Introduction

J'ai effectué ma première année d'école d'ingénieur en parcours LSI (Logiciel et système d'information) en alternance. Cette alternance est indispensable pour valider le diplôme et a pour objectif d'apporter à l'étudiant une expérience dans un milieu professionnel, d'approfondir les connaissances acquises au cours de l'année et d'en acquérir de nouvelles.

Après de longues recherches, j'ai trouvé cette alternance en répondant à une offre sur JobTeaser. Après avoir passé un test technique, j'ai été embauché en tant qu'alternant au poste de développeur back-end pour une durée de trois ans.

Mon année s'est déroulée chez Districall, une TPE^[1] comptant 6 employés située à Neuilly-sur-Seine. Districall est une entreprise française spécialisée dans la téléphonie cloud professionnelle. Elle a pour concurrents des entreprises telles que Allocloud , OVH Telecom ou encore Keyyo, proposant des services similaires. Toutefois, Districall se démarque par une approche centrée sur la simplicité, la mobilité et l'accessibilité, particulièrement adaptée aux TPE, PME^[1] et indépendants.

J'ai occupé le poste de développeur back-end, principalement avec le framework^[2] Symfony. En ce qui concerne la communication, nous utilisions des logiciels déjà en place dans l'entreprise, tels que Slack pour la messagerie instantanée, et Outlook pour les mails ou l'envoi de documents. En tant que développeur, j'ai été en lien avec les autres membres de l'équipe technique, et encadré par Antoine Lambert, dirigeant et tuteur de l'entreprise.

Au cours de cette année , j'ai contribué au développement de l'application Symfony de Districall. J'ai eu l'occasion de découvrir et d'utiliser des outils que je n'avais jamais utilisés auparavant, tels Notion ou encore Stripe. J'ai également manipulé des serveurs distants via des connexions SSH^[3].

Le rapport s'ouvrira d'abord sur une présentation de l'entreprise. La seconde partie portera sur la présentation de l'équipe et de mon rôle au sein de l'entreprise. La troisième partie sera consacrée aux missions réalisées. Enfin, le rapport se conclura sur l'apport professionnel et personnel de cette première année d'alternance.

Remerciements

Tout d'abord, avant de commencer, j'aimerais remercier mon tuteur en entreprise, Antoine Lambert, de m'avoir donné une chance chez Districall pour ma première année en tant qu'apprentie, de m'avoir fait confiance pour un contrat d'alternance, et pour ses qualités humaines telles que sa bonne humeur, sa gentillesse et sa bienveillance.

Je tiens également à remercier tous les membres de l'entreprise pour leur sympathie et leurs conseils. Un remerciement particulier à Loïc Dauchy, qui m'a accompagné tout au long de cette année.

Également, je tiens à remercier mon tuteur enseignant, Alain Haroun, pour son accompagnement et son suivi tout au long de l'année.

Je souhaite aussi remercier l'EFREI de m'avoir permis de vivre ma première expérience professionnelle en tant qu'alternant dans le domaine de l'informatique, dans le cadre de mon diplôme d'ingénieur.

Enfin je tiens à exprimer ma profonde reconnaissance à Alexandre Berard, président de l'EFREI, pour son immense générosité et sa gentillesse, sans qui je n'aurais pas pu poursuivre mon année à l'EFREI.

Abstract

This report summarizes my years apprenticeship experience at Districall. This apprenticeship was a crucial part of my first year in an Engineering degree from EFREI in software and information systems, as it provided me with my first hands-on experience of apprenticeship with a company and allowed me to expand my knowledge and acquire others.

Before starting this apprenticeship, I was uncertain about what to expect, as I had never apprenticed before. My main goals were to experience real team project management, understand the significance of assigned tasks, and get a taste of corporate life.

During my years, I worked as a back-end developer apprentice at Districall. Instead of focusing on a single major project, my internship involved various smaller tasks. I contributed to the deployment of API^[4] in Symfony and worked on improving the web application. I developed new functionalities, I resolved issues and I automated actions. These tasks included learning application codes, creating mailing templates, and setting up logs.

This apprenticeship provided me with a valuable opportunity to explore the working world. It helped me understand the various challenges faced by a company and the importance of teamwork. Apart from gaining technical knowledge, it gave me a clearer understanding of how a company operates.

I - Présentation de l'entreprise

1. Présentation de l'entreprise

Districall est une entreprise française qui opère dans le secteur de la télécommunication, plus précisément dans la téléphonie cloud professionnelle. Elle s'adresse principalement aux TPE, PME et travailleurs indépendants, en leur proposant des solutions de communication simples, flexibles et adaptées à leurs besoins.

L'objectif de Districall est d'apporter une solution clé en main à ses clients en prenant en charge l'ensemble de la gestion de leurs systèmes téléphoniques professionnels : attributions des numéros, paramétrage et accompagnement. Grâce à son approche centrée sur la simplicité, la mobilité et l'accessibilité, l'entreprise se démarque de ses concurrents, qui proposent souvent des offres plus complexes ou orientées vers les grandes structures.

Elle a été fondée en Juin 2023 par Antoine Lambert. C'est une TPE comptant 6 employés, située en France, à Neuilly-sur-Seine dans le 92200.

2. Organigramme & environnement professionnel

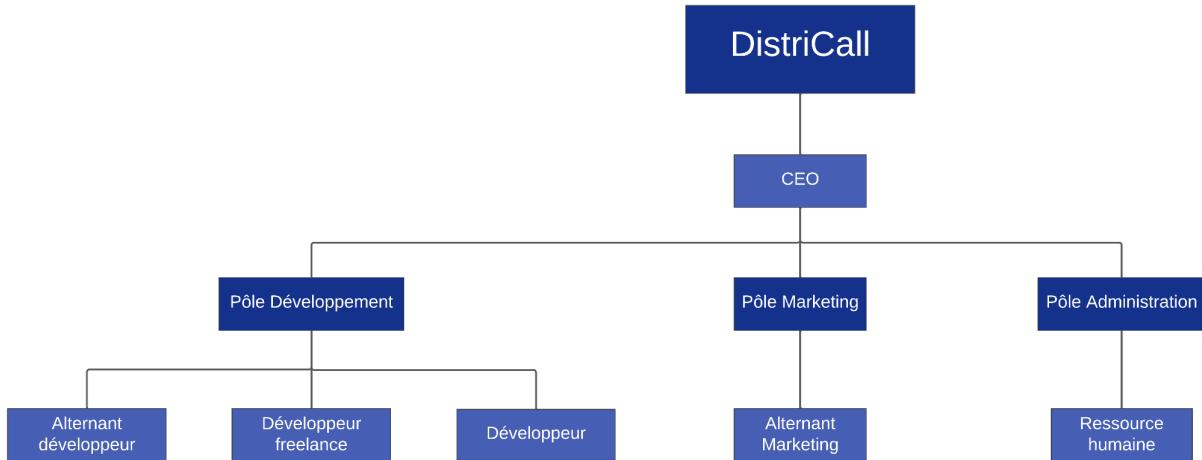


Figure 1 : Organigramme de l'entreprise Districall

L'entreprise dispose d'un effectif réduit. Cette configuration présente à la fois des avantages : une communication plus fluide entre les employés, les échanges sont rapides et directs, les décisions sont rapides et plus agiles, sans lourdes hiérarchies avec lesquelles les protocoles feraient perdre du temps. De plus, il y a une certaine polyvalence, par manque de spécialisation, les employés sont amenés à toucher à plusieurs aspects du projet.

Cependant, un effectif réduit présente aussi certains désavantages, comme la dépendance individuelle, malgré l'aide que nous fournissent les autres membres, pour un apprenti, on est parfois livré à nous même face à des blocages.

Au sein de l'entreprise des outils sont mis en place ayant chacun leurs utilités. Pour la communication instantanée, les échanges d'équipe et la création de canaux dédiés par projet ou sujet, l'entreprise utilise Slack. Outlook pour pouvoir envoyer des documents, des mails, etc. L'entreprise utilise Notion pour la documentation interne, le suivi de projet, la gestion des tâches et des connaissances et aussi pour centraliser des données. Et elle utilise Gitea, une sous-version de Git, pour la gestion de version, le suivi du code source et la collaboration sur le projet.

Au début de mon alternance, des réunions quotidiennes étaient organisées chaque matin avec les membres de l'équipe, afin de faire un point sur l'avancée des missions confiées, ma compréhension du projet et du code ainsi que de discuter des objectifs globaux de l'entreprise et ses axes de développement. Ces réunions favorisent une bonne intégration et un suivi progressif dans la montée en compétences. Cependant, au fil du temps, ces réunions sont devenues de moins en moins utiles, les échanges deviennent de plus en plus fluides et les missions plus autonomes. Pour optimiser le temps de chacun, il a donc été décidé de ne les organiser qu'en cas de besoin, lorsqu'un point spécifique ou une coordination d'équipe le justifie.

L'organisation du travail dans l'entreprise repose sur une autonomie forte, tout en maintenant une communication régulière au sein de l'équipe. Travaillant en full remote, chaque membre évolue à son rythme et gère ses missions de manière indépendante en s'appuyant sur les outils mis à disposition.

Au sein de l'équipe de développeurs, j'ai collaboré avec deux autres développeurs. Bien que nous échangions, chacun avait ses propres missions, et nous n'avons jamais eu à travailler simultanément sur une même partie. Cette répartition permet une meilleure concentration et une réactivité sur les tâches attribuées.

Les droits d'accès aux outils et environnements varient selon l'expérience et l'évolution dans l'entreprise. À mon arrivée, je n'avais accès qu'à une copie du projet et je travaillais exclusivement en local sur mon ordinateur, sans accès à la base de données distante, au serveur de production, au dépôt Git principal, ni aux environnements tiers comme Stripe. Progressivement, mes accès ont été élargis (accès à un serveur distant de preprod, accès réduit au serveur de production, à la base de données de production et à un environnement de test Stripe), ce qui m'a permis de gagner en autonomie et en efficacité dans mes missions.

3. L'environnement externe

L'entreprise évolue dans un secteur concurrentiel dominé par plusieurs acteurs bien établis, spécialisés dans les solutions de téléphonie cloud et de communication unifiée pour les professionnels, voici quelques concurrents :

- **OVH Telecom** : acteur majeur proposant des solutions cloud et VoIP^[5] avec une forte infrastructure en France et à l'international.
- **Keyyo** : entreprise française positionnée sur les télécommunications professionnelles, avec des solutions clé-en-main pour les PME^[1].
- **Allocloud** : fournisseur de téléphonie IP^[6] et de solutions collaboratives pour les entreprises.

II - Présentation de l'équipe

1. Contexte métier & Structure organisationnelle

L'entreprise est structurée autour de deux équipes principales :

- L'équipe technique, dédiée au développement de l'application. Elle a pour mission de maintenir et d'améliorer la plateforme en corrigéant les bugs, en implémentant de nouvelles fonctionnalités, en optimisant les performances, et en assurant la stabilité du système.
- L'équipe marketing et commerciale, chargée de la communication, de la prospection et de la vente. Son objectif est d'attirer de nouveaux clients, de promouvoir les offres de l'entreprise et de développer la notoriété de Districall sur son marché.

Au sein de l'entreprise, j'évolue dans le pôle technique. Notre objectif est d'assurer le bon fonctionnement de l'application utilisée par les clients, d'ajouter de nouvelles fonctionnalités selon les besoins exprimés par l'équipe commerciale ou les utilisateurs, et de garantir une expérience fluide et fiable. Cela implique un travail constant et minutieux de développement, de correction, de test et de déploiement.

2. Contexte technique

Dans le pôle technique, l'équipe fonctionne en full remote, ce qui offre une certaine liberté dans le choix des outils utilisés par chacun. Toutefois, certains outils restent essentiels au bon déroulement du développement du projet, afin d'assurer la cohérence, la collaboration et la qualité du code.

Voici les principaux outils et technologies que j'ai utilisés dans le cadre de mon alternance :

Environnements de développement :

- **Visual Studio Code** : Éditeur de code léger, utilisé principalement pour le développement de code rapide moins complexe.
- **PhpStorm** : Environnement de développement intégré (IDE)^[7] puissant, dédié au développement PHP, avec une très bonne intégration de Symfony, Git, et des outils de base de données.

Serveurs et environnement d'exécution :

- **MAMP** : Serveur local tout-en-un (Apache, MySQL, PHP) pour exécuter le projet Symfony en local. Très utile pour tester rapidement sans dépendre d'un serveur distant.
- **Protocole SSH** : Utilisé pour accéder aux serveurs distants via terminal, exécuter des commandes, déployer du code, consulter des fichiers de logs, etc.

Framework et technologies :

- **Symfony** : Framework^[2] PHP utilisé pour structurer l'ensemble du projet.
- **PHP** : Langage principal du backend.

Gestion de base de données :

- **PhpMyAdmin** : Interface web de gestion de bases de données relationnelles, utilisée en local avec une copie de la base de données pour effectuer des tests.
- **Adminer** : Alternative légère à phpMyAdmin, souvent utilisée pour un accès rapide aux bases de données sur certains serveurs. Utilisé pour la production

Services tiers et intégration :

- **Stripe** : Plateforme de gestion de paiements en ligne. Utilisée pour gérer les abonnements.
- **Mailtrap** : Outil permettant de capturer les mails envoyés depuis l'environnement de développement, sans qu'ils arrivent à de vrais destinataires. Pratique pour tester les mails.
- **TinyURL** : Service de génération de liens courts, utilisé pour transformer des URL^[8] longues en versions plus compactes, notamment dans les mails ou interfaces.

Autres :

- **Gitea** : Plateforme d'hébergement de code similaire à GitHub ou GitLab, auto-hébergée par l'entreprise.
- **Crips** : Plateforme de gestion de l'expérience client, alimentée par l'IA, qui réunit vos équipes, vos canaux de conversation, vos données et vos connaissances, en un seul endroit.



Figure 2 : Logo des outils utilisés : VSCode - PhpStorm - MAMP - Symfony - PHP - PhpMyAdmin - Stripe - Gitea

3. Organisation du travail

Au sein du pôle technique, l'organisation du travail est particulièrement flexible et autonome. Chaque membre de l'équipe est responsable de ses propres missions, ce qui permet une meilleure répartition des tâches et une exécution plus efficace. Ainsi, nous ne travaillons pas en groupe sur un même sujet, afin d'éviter les conflits de code, de simplifier le processus de développement et de permettre à chacun d'avancer à son rythme sur ses responsabilités.

Nous adoptons une méthodologie de travail agile^[10], fondée sur un cycle itératif : développer, tester, puis implémenter. Cette approche permet de livrer régulièrement des évolutions fonctionnelles tout en assurant une bonne qualité du produit. Même si nous ne suivons pas la méthode SCRUM^[9] de manière stricte (pas de sprints ou de mélées quotidiennes), nous restons fidèles à l'esprit agile^[10] : réactivité, adaptation aux besoins, et livraison continue.

La communication reste néanmoins un pilier essentiel du bon fonctionnement de l'équipe. Bien que chacun soit autonome, nous échangeons quotidiennement via l'application Slack, notamment avec mon tuteur, afin de poser des questions, obtenir des précisions ou partager des avancées. Cela permet de maintenir une bonne coordination tout en respectant l'indépendance de chacun.

Concernant les réunions, leur fréquence est volontairement limitée. Étant une petite structure, les réunions journalières ne sont pas nécessaires et pourraient ralentir inutilement le rythme de travail. Elles ont donc été remplacées par des points ponctuels, organisés uniquement en cas de besoin. Cette stratégie permet d'optimiser le temps de chacun tout en assurant une bonne circulation de l'information lorsque cela est nécessaire.

4. Présentation du poste et de la mission

En tant qu'alternant développeur back-end, je n'ai pas été affecté à une seule grande mission, mais plutôt à une succession de petites tâches variées, me permettant d'aborder plusieurs aspects du projet. L'un de mes premiers objectifs a été de mettre en place des automatisations destinées à faciliter certaines opérations récurrentes. Ces automatisations concernaient notamment :

- L'envoi de plusieurs mails automatiques, selon des conditions précises.
- L'envoi automatisé de SMS, pour notifier les utilisateurs de certaines actions.
- L'automatisation de processus liés à la base de données.
- L'automatisation de requêtes Stripe via des webhooks, afin de gérer automatiquement les paiements, les abonnements ou les échecs de transactions.

Au-delà de ces automatisations, j'ai également participé au développement de nouvelles fonctionnalités, telles que :

- L'affichage de données dynamiques dans la section administrateur du site web.
- L'intégration d'un webhook Stripe, afin de recevoir et traiter des événements en temps réel.
- La connexion à l'API^[4] de Crisp, pour récupérer des données issues du service de chat.
- L'ajout d'une page de statistiques dans l'interface d'administration, permettant de visualiser des données clés sous forme graphique.

Par ailleurs, j'ai aussi pris part à la résolution de bugs et à la correction d'anomalies affectant la base de données ou les fonctionnalités du site. J'ai contribué à l'amélioration de la qualité du code, en appliquant de bonnes pratiques de développement (factorisation, optimisation des requêtes, clarté du code...).

Mon intégration s'est faite progressivement : j'ai d'abord réalisé des tâches simples sur mon environnement local, puis j'ai obtenu l'accès à des modules plus sensibles, comme la base de données distante et le serveur de production. Cette montée en compétences m'a permis de gagner en autonomie et de comprendre plus en profondeur l'architecture du projet.

III - Missions : Automatisation de tâches, développement de fonctionnalités & résolution de bugs

1. Contexte & Existant

L'entreprise gère son produit à travers trois modules principaux :

- la partie code applicatif, développé avec Symfony
- le système de paiement en ligne, intégré avec Stripe
- la gestion de données, assurée via Adminer et Notion

La partie code constitue le cœur du produit. Celui-ci permet à la fois le développement du site web de l'entreprise et la gestion de l'interconnexion avec les autres modules. Symfony est un framework^[2] open-source PHP reconnu pour sa robustesse et sa flexibilité, il repose sur le paradigme MVC^[11] (Modèle-Vue-Contrôleur), qui assure une séparation claire des responsabilités dans le code et une meilleure maintenabilité. Il propose également de nombreux outils, composants réutilisables et une architecture bien définie, ce qui facilite le travail collaboratif et accélère le développement.



Figure 3 : Exemple d'architecture Symfony

Un projet Symfony est structuré selon une arborescence de dossiers. Voici les principaux répertoires utilisés :

- `src/` : contient la logique métier (contrôleurs, entités, services, etc.)
- `templates/` : contient les fichiers d'affichage HTML via le moteur de template Twig
- `config/` : fichiers de configuration du projet
- `public/` : accessible publiquement, contient les assets (images, JS, CSS)

On y retrouve également un fichier essentiel au bon fonctionnement du projet : le fichier `.env`.

```
DATABASE_URL="mysql://root:root@127.0.0.1:8889/districall_test?serverVersion=8.0&charset=utf8mb4"
APP_ENV=test
APP_DEBUG=true
APP_SECRET=test
JWT_SECRET_KEY=%kernel.project_dir%/config/jwt/private.pem
JWT_PUBLIC_KEY=%kernel.project_dir%/config/jwt/public.pem
JWT_PASSPHRASE=test
INTERNE_APIKEY="12345"
CORS_ALLOW_ORIGIN="*"
MAILER_DSN="smtp://test:test@sandbox.smtp.mailtrap.io:2525"
SMS_MODE_KEY="test"
NOTION_API_KEY="test"
STRIPE_SK="sk_test"
STRIPE_PK="pk_test"
CRISP_WEBSITE_ID="test"
CRISP_IDENTIFIER="test"
CRISP_KEY="test"
```

Figure 4 : Exemple de fichier `.env`

Le fichier `.env` permet de centraliser les variables d'environnement, comme : les clés d'API^[4], les identifiants, les URLs^[8] de services. Il joue un rôle crucial dans la configuration dynamique du projet, notamment lors du passage d'un environnement local à un environnement de production.

Voici quelques exemples de variables présentes dans ce fichier :

- DATABASE_URL : URL^[8] de connexion à la base de données (gérée via Adminer)
- MAILER_DSN : configuration du serveur de l'envoi de mails
- SMS_MODE_KEY : clé d'authentification pour le serveur de l'envoi de SMS
- NOTION_API_KEY : clé API^[4] permettant l'accès aux données stockées sur Notion
- STRIPE_SK / STRIPE_PK : clés secrètes et publiques pour interagir avec l'API^[4] Stripe

Ce fichier permet donc à l'application Symfony de communiquer avec les différents services externes indispensables au bon fonctionnement du projet.

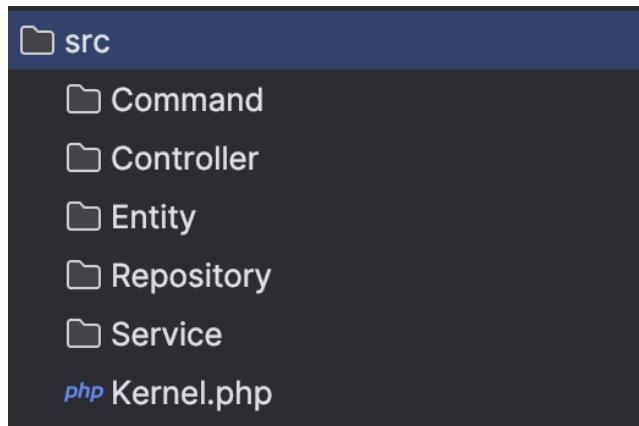


Figure 5 : Exemple d'architecture du dossier src

Le dossier src/ est le répertoire source principal de Symfony. Il contient tout le code PHP personnalisé de l'application répartie dans des sous-dossiers :

- **Commande** : Ce dossier contient les commandes personnalisées exécutables en ligne de commande.
- **Controller** : Ce dossier contient les contrôleurs, qui sont les classes chargées de recevoir les requêtes HTTP^[12].
- **Entity** : Contient les entités, qui représentent les tables de ta base de données sous forme de classes PHP.
- **Repository** : Contient les repositories, qui sont des classes permettant de faire des requêtes personnalisées à la base de données sur les entités.

- **Service** : Ce dossier contient les services : des classes qui encapsulent une logique métier réutilisable.

2. Description des outils et de l'environnement technique

J'ai commencé à développer mes premières fonctionnalités à l'aide de Visual Studio Code, un éditeur de code léger, flexible et largement utilisé dans le monde du développement.

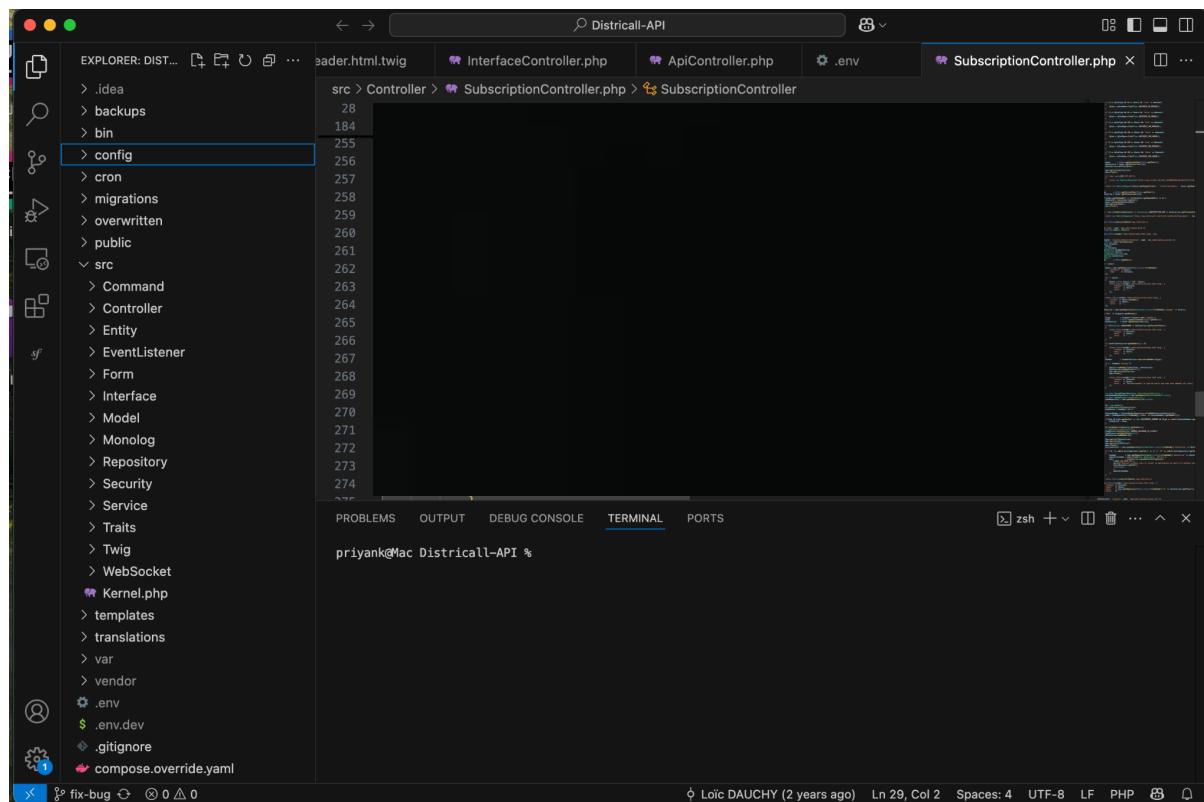


Figure 6 : Interface de VSCode

Visual Studio Code est particulièrement adapté aux petits projets ou aux développeurs recherchant un environnement rapide à configurer. Il prend en charge de nombreux langages de programmation grâce à ses extensions, et permet d'exécuter, de tester ou de visualiser du code facilement.

Cependant, pour un projet de plus grande envergure comme celui de Districall, j'ai rapidement rencontré certaines limitations, notamment :

- Des fonctionnalités avancées limitées en comparaison d'un IDE^[7] complet.
- Une correction de syntaxe parfois imprécise ou absente sans extensions supplémentaires.
- Un support plus limité pour la navigation dans les projets complexes (autocomplete, refactoring, débogage avancé...).

Face à ces contraintes, je me suis progressivement tourné vers PhpStorm, un environnement de développement intégré (IDE)^[7] beaucoup plus complet et puissant, spécialement conçu pour les projets en PHP et compatible avec le framework^[2] Symfony.

PhpStorm offre une expérience de développement bien plus productive grâce à :

- Une intégration native de Symfony, facilitant la navigation dans le code.
- Un système d'autocomplétion intelligent, la détection d'erreurs en temps réel, et des outils de refactoring puissants.
- Une interface qui centralise tout (base de données, terminal, serveur local, etc.).

PhpStorm m'a ainsi permis de gagner en efficacité et en confort de travail.

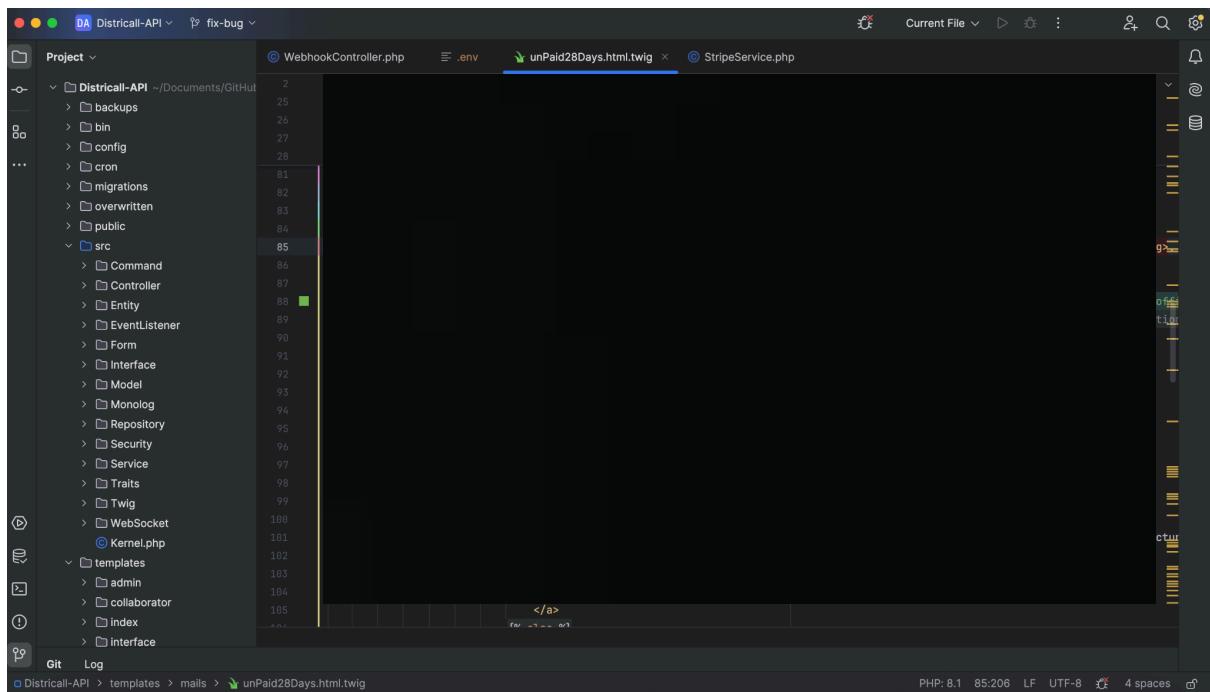


Figure 7 : Interface PhpStorm

En environnement local, j'utilise personnellement phpMyAdmin comme interface de gestion de base de données. Il s'agit d'un outil open-source, qui permet de manipuler des bases de données MySQL via une interface graphique accessible depuis un navigateur.

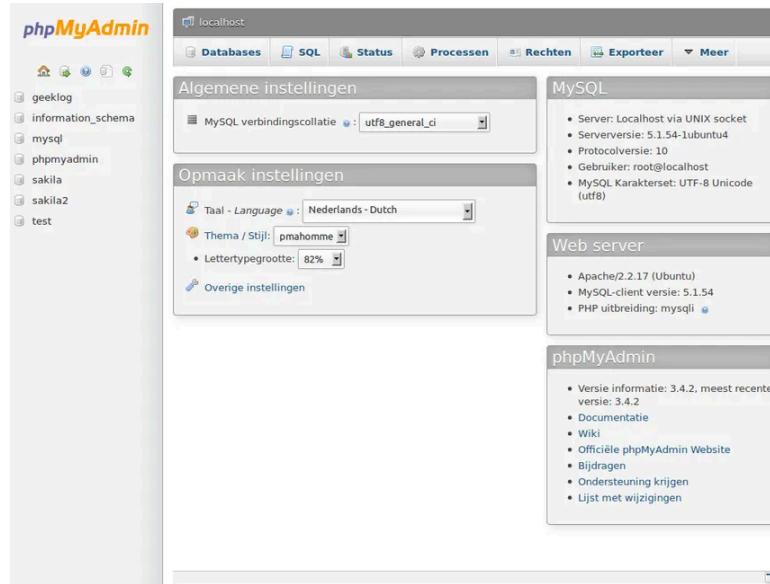


Figure 8 : Interface phpMyAdmin

L'entreprise utilise Adminer comme interface de gestion de sa base de données. Adminer est un outil open-source, léger et simple d'utilisation, qui permet d'administrer une base de données via une interface web.

Table	Engine?	Collation?	Data Length?	Index Length?	Data Free?
wp_commentmeta	InnoDB	utf8mb4_unicode_ci	16,384	32,768	0
wp_comments	InnoDB	utf8mb4_unicode_ci	16,384	81,920	0
wp_links	InnoDB	utf8mb4_unicode_ci	16,384	16,384	0
wp_options	InnoDB	utf8mb4_unicode_ci	1,556,480	32,768	4,194,304
wp_postmeta	InnoDB	utf8mb4_unicode_ci	16,384	32,768	0

Figure 9 : Interface Adminer

Contrairement à des solutions plus lourdes comme phpMyAdmin, Adminer tient dans un seul fichier PHP et ne nécessite qu'une configuration minimale.

Adminer permet notamment de :

- Se connecter à une base de données MySQL, PostgreSQL, SQLite, etc.
- Consulter, modifier, supprimer ou insérer des données dans les tables.
- Éditer la structure des tables (ajouter, supprimer ou modifier des colonnes).
- Lancer des requêtes SQL^[13] personnalisées.

Cette base de données comporte plusieurs avantages

- Léger et rapide : un seul fichier à déployer, moins gourmand que phpMyAdmin.
- Simplicité d'utilisation : interface claire, épurée, facile à prendre en main.
- Portable : peut être utilisé même sur des environnements restreints.
- Sécurisé : régulièrement mis à jour avec un bon niveau de sécurité.

Pour assurer la fiabilité du code avant sa mise en production, j'ai mis en place un environnement de test local me permettant de valider chaque modification dans des conditions contrôlées.

Cela m'a permis de reproduire les erreurs rencontrées, de simuler différents cas d'usage, et de m'assurer que les corrections et ajouts fonctionnaient comme prévu.

Pour cela, j'ai utilisé plusieurs outils :

1. **PhpStorm – Terminal intégré** : PhpStorm dispose d'un terminal intégré que j'ai utilisé pour exécuter des commandes Symfony, lancer des scripts d'automatisation, etc.

2. **Environnement de test Stripe** : Afin de tester les fonctions liées aux paiements, j'ai eu accès à un environnement de test Stripe. Celui-ci permettait de :
 - a. Créer de faux clients
 - b. Émettre des factures fictives
 - c. Simuler des paiements réussis ou échoués
 - d. Tester les webhooks associés à ces événements

Cela m'a permis de valider le bon fonctionnement des automatisations Stripe sans impacter les données de production.

3. **Base de données de test** : Pour reproduire certains bugs ou tester des traitements sur les données, l'équipe m'a fourni une ancienne version exportée de la base de données. Grâce à cet export, j'ai pu effectuer mes tests en conditions réelles sans aucun risque sur les données actuelles de l'entreprise.
4. **Mailtrap – Test des mails** : Pour tester l'envoi et l'affichage des mails générés par l'application, j'ai utilisé Mailtrap. Cet outil permet :
 - a. De capturer les mails envoyés depuis l'environnement de développement
 - b. De prévisualiser le contenu et texte
 - c. D'éviter tout envoi réel vers des utilisateurs

Mailtrap s'est révélé essentiel pour valider les modèles de mails générés automatiquement.

5. **Postman – Tests des API** : J'ai utilisé Postman pour tester les différentes API^[4] utilisées ou exposées par l'application. Cela m'a permis de :
 - a. Vérifier le bon fonctionnement des routes API^[4]
 - b. Tester différents paramètres d'entrée et analyser les réponses
 - c. Valider les appels externes (Crisp, Stripe...)

Grâce à cet ensemble d'outils, j'ai pu développer et valider les fonctionnalités de manière fiable, structurée et autonome, tout en garantissant un haut niveau de qualité pour les utilisateurs finaux.

Pour corriger efficacement les bugs rencontrés dans le projet, il était nécessaire de simuler des cas concrets sur un environnement de test, afin d'identifier précisément l'origine de l'erreur.

Pour cela, l'entreprise m'a fourni un serveur distant sur lequel une copie du projet était déployée. Ce serveur était accessible via SSH^[3].

SSH^[3] est un protocole sécurisé qui permet de se connecter à un serveur distant via une interface en ligne de commande. Il offre un canal chiffré pour exécuter des commandes à distance, consulter des fichiers ou déployer du code.

```
priyank@Mac ~ % ssh utilisateur@adresse_ip
```

Figure 10 : Exemple de connexion SSH

Bien que cet environnement distant permette de tester en conditions quasi réelles, cette méthode s'est révélée peu pratique pour des tests rapides :

- Je devais manuellement configurer les scénarios à tester.
- Les temps de chargement étaient plus longs qu'en local.
- Le déploiement de nouvelles versions du code ralentissait le processus.

Pour faciliter le débogage, j'ai obtenu l'accès aux logs du projet, ce qui a considérablement amélioré mon efficacité.

Un log est un fichier dans lequel l'application enregistre automatiquement des événements survenus lors de son exécution : erreurs, actions utilisateurs, appels d'API^[4], requêtes SQL^[13], etc.

Ces informations sont précieuses pour comprendre ce qui s'est passé, à quel moment, et dans quel ordre.

```
app@prod-2025-06-04.log  
app@prod-2025-06-05.log  
app@prod-2025-06-06.log
```

Figure 11 : Fichiers log

L'organisation des fichiers est la suivante :

- Un fichier de logs par jour
- Les événements y sont enregistrés de façon chronologique
- Chaque ligne contient les détails des actions que l'on a choisi de tracer (date, heure, niveau de gravité, message, contexte)

```
[2025-05-23T13:45:02.013118+02:00] app.INFO: Docx ID: 421, EnterpriseID: 9976, Heures restantes: -650.15 []
[2025-05-23T13:45:02.013207+02:00] app.INFO: Docx ID: 421 []
[2025-05-23T13:45:02.013245+02:00] app.INFO: Cutoff Date: 2025-04-26 11:35:56 []
[2025-05-23T13:45:02.013277+02:00] app.INFO: Now: 2025-05-23 13:45:01 []
[2025-05-23T13:45:02.015508+02:00] app.INFO: Un email a déjà été envoyé ce cycle de facturation. Abandon de l'envoi. []
[2025-05-23T13:45:02.015569+02:00] app.INFO: Ligne coupée pour Enterprise ID: 9976, Docx ID: 421 []
```

Figure 12 : Exemple de ligne d'un fichier log

Grâce à ces logs, le débogage est devenu plus rapide et plus fiable. Je pouvais :

- Identifier quelle action avait été déclenchée
- À quelle heure et dans quel contexte
- Sur quel utilisateur ou élément précis
- Et quelles erreurs éventuelles avaient été générées en retour

Ce système de journalisation m'a permis de corriger plus rapidement les bugs, de valider les comportements attendus et de suivre les automatisations mises en place.

3. Présentation des missions et analyse de ses réalisations

1. Automatisation de tâches

L'automatisation de tâches est plutôt facile à mettre en place sur un projet Symfony, grâce au commande PHP exécutable et au système de cron job. Un cron job est un programme utilisé pour exécuter automatiquement des scripts ou des commandes à des horaires définis (quotidiennement, toutes les heures, etc.). Cela permet d'assurer la régularité des processus sans intervention manuelle. On les programme dans un crontab. Voici un exemple :

```
#0 0 1,15 * * /cron/checkStripeEnterpriseSubscription.sh
0 9 * * * /cron/checkTrialPeriod.sh
0 9 * * * /cron/alertFreeSda.sh
0 9 * * * /cron/checkTrialPeriodWillEndSoon.sh
0 9 * * * /cron/checkUnpaid.sh
0 9 * * * /cron/sdaLiberation.sh
```

Figure 13 : Exemple de fichier crontab

Dans la première partie nous avons les horaires à laquelle la tâche doit s'exécuter, puis nous avons la commande PHP à exécuter.

L'une de mes missions consistait à automatiser l'envoi d'une routine de mails pour les clients inscrits n'ayant pas pris d'abonnement. Cette routine de mails était composé d'une suite de 8 mails à envoyer à des timings différents :

- Premier mail : + 2 heures après inscription
- Deuxième mail : + 2 jours après inscription
- Troisième mail : + 7 jours après inscription
- etc.

Pour mener à bien cette mission, j'ai d'abord repris le template utilisé pour les mails.

```
width="600"><tbody><tr><td class="column column-1" width="100%" style="... "><table class="image_block block-1" style="... "><a href="#" target="_blank" style="... " tabindex="-1"></tr><tr><td class="pad" style="... "><div style="... "><div class style="... "><p style="... ">Avec l'application mobile Districal, permettez simplement à vos collaborateurs de rappeler ou d'appeler en aff...</p></div></div></td></tr><tr><td class="text_block block-1" width="100%" border="0" cellpadding="10" cellspacing="0" role="pr...</td></tr><tr><td class="pad" style="... "><div style="... "><div class style="... "><p style="... ">Obtenez l'application !</p></div></div></td></tr><tr><td class="alignment" align="center"><!--[if mso]><v:roundrect xmlns:v="urn:schemas-microsoft-com:vml" xmlns:w="urn:schemas-microsoft-com:offic...</td></tr>
```

Figure 14 : Code du template de mails avant réarrangement

Le code était chaotique, très compacte et complètement illisible, j'ai dû le réarranger en corrigeant les indentations, ce qui m'a donné :

```
<td class="pad" style="... ">
  <div style="... ">
    <div class style="... ">
      <p style="... ">
        <span style="... ">Merci de faire confiance à <strong>Districal</strong> !<br><br>Vous avez choisi la formule <s...</span>
      </p>
      <p style="... "> </p>
    <tr>
      <td class="pad"><div class="alignment" align="center">
        <!--[if mso]><v:roundrect xmlns:v="urn:schemas-microsoft-com:vml" xmlns:w="urn:schemas-microsoft-com:offic...</td>
        <a href="https://app.districal.com/login" target="_blank" style="... ">
          <span style="... ">
            <span dir="ltr" style="... ">
              <strong><span style="... " dir="ltr" data-mce-style="font-size:12px;">ME CONNECTER</span></strong>
            </span>
          </span>
        </a>
        <!--[if mso]></center></v:textbox></v:roundrect><![endif]-->
      </div>
    </td>
```

Figure 15 : Code du template de mails après réarrangement

Le code est beaucoup plus espacé, nous pouvons mieux distinguer les différentes parties et la lisibilité est améliorée.

Pour la création du script, l'objectif était de centraliser toute la logique dans un seul script réutilisable, plutôt que de créer un script distinct pour chaque type de mail.

1. J'ai d'abord créé un tableau contenant toutes les informations nécessaires à l'envoi des mails (titre, période d'envoi, nom du fichier de template et le type de mail)

```
],
[
    'label' => '+7J',
    'start' => '-7 days',
    'end' => '-6 days -22 hours',
    'title' => '-30% partout, non, tu ne rêves pas.',
    'mailFilename' => 'routineEmail7Days',
    'type' => Mail::ROUTINE_NOSUB_7J,
],
[
    'label' => '+14J',
    'start' => '-14 days',
    'end' => '-13 days -22 hours',
    'title' => 'Pro ou perso ?',
    'mailFilename' => 'routineEmail14Days',
    'type' => Mail::ROUTINE_NOSUB_14J,
]
```

Figure 16 : Tableau regroupant les informations des mails

2. Le script parcourt toutes les entreprises non-abonnées dans la base de données.
3. Pour chaque type de mail et chaque entreprise, le script vérifie :
 - Si la date d'inscription de l'entreprise entre dans la période d'envoi définie.
 - Si ce mail n'a pas déjà été envoyé à cette entreprise (afin d'éviter les doublons).
4. Si toutes les conditions sont remplies, le script déclenche l'envoi du mail.

Grâce à ce système, l'entreprise peut automatiser une communication ciblée et planifiée auprès des prospects ou clients, tout en évitant les doublons et en gardant un contrôle total sur les types de mails envoyés.

Pour l'envoi des mails, l'entreprise utilise un serveur SMTP^[14] fourni par Brevo, un service spécialisé dans les mails marketing et les communications transactionnelles.

Une fois le script d'envoi de mails finalisé, il était essentiel de le tester en toute sécurité avant de le déployer en production. Pour cela, j'ai utilisé deux outils complémentaires : Mailtrap et une base de données de test.

The screenshot shows the Mailtrap interface. On the left is a sidebar with navigation links: Home, For developers (API/SMTP, Sandbox), For marketers (Campaigns, Automations, new, Contacts), General (Sending Domains, Templates, Integrations, new, Billing, Settings). The main area is titled 'My Inbox' with tabs for Integration, Email Address, Auto Forward, Manual Forward, and Access Rights. Below this is a section titled 'Integration' with a sub-section 'Credentials'. The 'Credentials' table contains the following information:

Host	sandbox.smtp.mailtrap.io
Port	25, 465, 587 or 2525
Username	6b19f0a7493467
Password	*****5f3
Auth	PLAIN, LOGIN and CRAM-MD5
TLS	Optional (STARTTLS on all ports)

Figure 17 : Interface Mailtrap

J'ai alors testé tous les scénarios pour chaque type de mail, cette méthode m'a permis de valider en toute sécurité le bon fonctionnement du script d'envoi, de corriger les éventuelles erreurs de logique ou de contenu, et de m'assurer que chaque type de mail s'envoyait correctement selon les règles définies.

2. Développement de nouvelles fonctionnalités

L'une des nouvelles fonctionnalités que j'ai eu à développer au cours de mon alternance consistait à implémenter un moteur de recherche dans l'interface administrateur du site web, plus précisément pour la gestion des entreprises.

Avant cette amélioration, la page d'administration affichait une liste complète de toutes les entreprises présentes dans la base de données. Ce fonctionnement posait plusieurs problèmes :

- Temps de chargement long, dû au volume important de données
- Navigation difficile, nécessitant parfois un usage manuel de CTRL + F^[15], peu ergonomique
- Aucune option de filtrage ou tri dynamique

L'objectif était donc de permettre une recherche efficace, rapide et intégrée à l'interface.

Dans un premier temps, il me fallait déjà accéder à la partie admin du site web. Pour cela, j'ai créé un compte administrateur sur ma base de données test, et j'ai lancé une version test du projet en local grâce à la commande `symfony server:start`, qui permet de lancer un instance du projet, pour ainsi avoir un visuel de l'existant. J'avais alors accès à la page sans influencer les données de la production.

J'ai ensuite modifié le code de la page d'administration pour :

1. Ajouter des barres de recherche accessible depuis l'interface
2. Gérer la soumission du formulaire pour envoyer les critères de recherche
3. Filtrer les résultats affichés en fonction de ces critères

Pour faciliter ce processus, j'ai utilisé le bundle Symfony KnpPaginatorBundle, qui permet :

- De paginer automatiquement les résultats issus de la base
- D'intégrer facilement la recherche et le tri dans les listes
- De réduire considérablement le temps de chargement de la page

La recherche dans la base de données a été effectuée via le système de repository de Symfony. Cela permet de :

- Créer des requêtes personnalisées dans un fichier dédié (ex : EntrepriseRepository)
- Filtrer les résultats en fonction des champs saisis par l'utilisateur
- Garder une logique métier claire et réutilisable

Dans la logique :

- L'utilisateur émet une recherche via les barres de recherche.
- L'interface fait appelle au repository qui recherche les instances dans la base de données correspondant aux critères entrés.
- La base de données renvoie les données trouvées et l'interface les affiche.

Numéros/Clients

Recherche Globale						
Tc	Numéro	Entreprise	Contact	État	Date d'inscription	Action
23	[REDACTED]	[REDACTED]	[REDACTED]	Abonné à vie	Le 04-10-2023	LIBÉRER LE SDA SE CONNECTER VOIR LE CLIENT
25	[REDACTED]	[REDACTED]	[REDACTED]	Abonné à vie	Le 07-04-2024	LIBÉRER LE SDA

Figure 18 : Interface administrateur avec les barre de recherche

Recherche Globale						
Tc	Numéro	Entreprise	Contact	État	Date d'inscription	Action
611	[REDACTED]	[REDACTED]	[REDACTED]	Abonné	Le 13-04-2023	LIBÉRER LE SDA SE CONNECTER VOIR LE CLIENT
						« Précédent 1 2 3 4 5 ... 60 Suivant »

Figure 19 : Système de pagination

Grâce à cette fonctionnalité, l'interface administrateur est désormais plus fluide, plus rapide et beaucoup plus ergonomique.

3. Résolution de bugs

L'un des bugs majeurs que j'ai rencontré durant cette première année, est l'envoie multiple de mail d'impayé à des clients. Quand le client a un impayé, nous lui envoyons un mail pour le prévenir. Ce mail est envoyé lors de l'exécution du webhook Stripe *invoice.payment_failed*, qui survient lorsqu'un client a un impayé sur Stripe. Un webhook est lancé pour prévenir d'un événement, nous pouvons ensuite agir en fonction de celui-ci.

Dans le cas d'un *invoice.payment_failed*, plusieurs actions sont effectuées :

- Nous tentons de prélever la facture avec tous les moyens de paiement enregistrés par le client.
- En fonction du nombre d'impayés de celui-ci, nous effectuerons des mises à jour dans la base de données et nous lui envoyons des mails pour le prévenir de sa situation.

Cependant, en regardant les logs, j'ai constaté que de tenter de prélever la même facture avec plusieurs moyens de paiement relance un webhook *invoice.payment_failed*, j'ai alors compris que la logique du webhook était relancé pour chaque tentative de paiement échoué. Un système pour empêcher la relance infini du webhook était mis en place : dans la facture, un nombre de tentatives maximum était mis en place, elle ne pouvait pas dépasser le nombre de moyens de paiement . De plus, il y a une vérification de mail déjà envoyé. Quand le mail est envoyé, on enregistre l'envoie dans la base de données et avant de l'envoyer on fait une vérification pour voir s'il a déjà été envoyé.

Sauf que cela ne suffisait pas, parfois des webhooks s'exécutent avec trop peu de temps d'écart, ce qui ne laissait pas le temps à la base de données de ce mettre à jour et donc le mail était envoyé 2 fois voir 3 parfois.

La première solution mise en place a été d'ajouter un temps d'attente entre les tentatives, à raison d'une seconde entre chaque tentative. Cette approche a permis de réduire la fréquence du bug, mais ne l'a pas entièrement éliminé. Le problème persistait de manière moins récurrente, mais survenait encore dans certains cas.

Pour espérer le résoudre totalement, il aurait fallu augmenter le temps d'attente, ce qui n'était pas optimal dans le cadre d'un système automatisé, car cela ralentissait inutilement le traitement. De plus, Stripe dispose déjà d'un mécanisme intégré de relance automatique des webhooks, essayant l'envoi pendant plusieurs jours en cas d'échec. Cela rendait l'ajout d'attentes côté serveur redondant et inefficace à long terme.

Il a fallu alors revoir complètement la structure du webhook pour corriger ce bug.

Dans un premier temps, j'ai analysé toutes les actions effectuées dans le webhook et j'ai essayé de les regrouper au maximum, pour une meilleure gestion des cas.

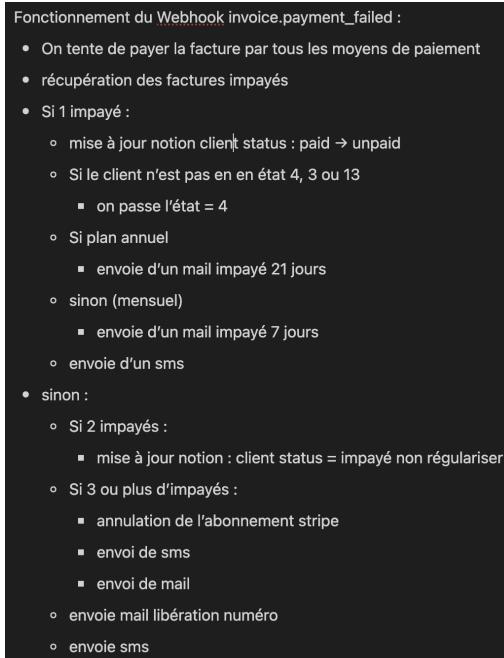


Figure 20 : Fonctionnement du Webhook invoice.payment_failed

Après simplification du webhook, il fallait trouver un moyen d'exécuter les tentatives de paiement une seule fois, ainsi que la suite du script une seule fois aussi et après la dernière tentative de paiement.

Pour cela, Stripe offre un champ metadata pour chaque facture, dans lequel nous pouvons entrer des informations.

Ma solution a été d'enregistrer 3 paramètres boolean dans le champs metadata :

- **manualAttempt** : servant à distinguer la première tentative automatique de Stripe et les tentatives manuelles, pour ne pas relancer le système de tentative avec tous les moyens de paiement plusieurs fois.
- **attemptFinished** : permet de savoir quand toutes les tentatives de paiement ont été effectuées et ainsi lancer la suite du script quand il est vrai.
- **processingFinished** : permet de savoir si toute la logique du webhook a été effectuée, pour ne pas la relancer plusieurs fois.

La logique ressemblait alors à ça :

```
Fonctionnement du Webhook invoice.payment_failed :  
Si manualAttempt == false :  
    • On tente de payer la facture par tous les moyens de paiement  
Si attemptFinished == true et processingFinished == false :  
    • récupération des factures impayés  
    • suite du webhook
```

Figure 21 : Logique du Webhook invoice.payment_failed après modification

4. Indicateurs de performances

1. Automatisation de tâches

En ce qui concerne l'envoi automatique des e-mails, chaque envoi est enregistré dans la base de données, avec deux informations essentielles :

- Le type de mail envoyé
- L'entreprise destinataire

Grâce à ce système de traçabilité, il est facile de vérifier quels e-mails ont été envoyés à quelles entreprises. Cela permet non seulement d'éviter les doublons, mais aussi de suivre précisément l'historique des communications, notamment pour les relances ou les mails transactionnels.

2. Développement de nouvelles fonctionnalités

Pour les nouvelles fonctionnalités, comme la barre de recherche, mon tuteur me faisait régulièrement des retours afin d'indiquer ce qu'il fallait améliorer, corriger ou valider. Cela me permettait de m'auto-corriger rapidement et de m'assurer que le résultat final correspondait bien aux attentes.

3. Résolution de bugs

De manière générale, lors de la résolution de bugs, je consultais les fichiers de logs afin de vérifier si l'erreur s'était reproduite ou non, ou si la logique mise en place était belle et bien appliquée. Cela me permettait d'analyser le comportement de l'application, d'identifier l'origine du problème, et de valider que la correction apportée avait bien résolu l'anomalie.

IV - Bilan des réalisations : apports, difficultés, échecs et réussites

1. Identification du profil du poste et des compétences acquises à travers la mission d'apprentissage.

Cette première année chez Districall a été pour moi une expérience très enrichissante. Elle m'a permis de développer de nombreuses compétences et de découvrir le fonctionnement d'une entreprise comme celle-ci.

Le poste de développeur back-end en alternance chez Districall nécessite une bonne maîtrise de PHP, une capacité à s'adapter à un projet existant basé sur le framework^[2] Symfony, ainsi qu'une connaissance dans les différents outils utilisés par l'entreprise comme Stripe, Adminer, etc. Il exige également de la rigueur, de l'autonomie, une capacité à résoudre des problèmes techniques, ainsi qu'une bonne communication.

Voici une liste exhaustive des compétences technique travaillé tout au long de l'année :

- Développement en PHP avec le framework^[2] Symfony
- Manipulation de base de données avec Symfony et SQL^[13]
- Utilisation d'outil de versioning Gitea
- Mise en place de tâches automatisées avec cron
- Création et test de webhooks Stripe
- Utilisation de différents outils comme Mailtrap, Postman, Notion, etc.
- Débogage grâce aux logs applicatifs

En plus des compétences technique, j'ai aussi acquis des soft skills :

- Autonomie en environnement full remote
- Résolution de problèmes techniques de manière structurée
- Compréhension du cycle de vie d'une fonctionnalité
- Prise d'initiative
- Communication efficace
- Capacité d'organisation et de priorisation

2. Tendances et facteurs d'évolutions en entreprise.

Districall évolue dans un secteur en constante évolution, celui de la téléphonie cloud. Celà crée une grande concurrence entre les différents acteurs. Cette concurrence pousse Districall à se différencier par sa proximité client, sa flexibilité, et sa réactivité dans le support, tout en proposant des prix compétitifs. Avec l'évolution du marché de la téléphonie cloud, plusieurs tendances se distinguent, poussant l'entreprise à faire face à des opportunités :

- **Digitalisation accélérée** : les entreprises adoptent de plus en plus de solutions cloud pour moderniser leur infrastructure télécom, réduire les coûts et gagner en mobilité.
- **Télétravail** : la généralisation du travail à distance nécessite des outils de communication efficaces, accessibles de n'importe où
- **Niche de proximité** : une entreprise à taille humaine peut se différencier par un support client personnalisé et réactif.

L'essor du télétravail, la numérisation accélérée des TPE/PME^[1], et l'émergence de technologies interconnectées poussent les entreprises comme Districall à innover continuellement. Dans ce contexte, la simplicité d'usage, la rapidité de déploiement, et la qualité du support client deviennent des atouts concurrentiels majeurs. Districall se distingue par son positionnement sur les petites structures, tout en adaptant son produit aux nouvelles attentes du marché.

Table des figures

Figure 1 : Organigramme de Districall - **page 6**

Figure 2 : Logo des outils utilisés : VSCode - PhpStorm - MAMP - Symfony - PHP - PhpMyAdmin - Stripe - Gitea - **page 11**

Figure 3 : Exemple d'architecture Symfony - **page 14**

Figure 4 : Exemple de fichier .env - **page 15**

Figure 5 : Exemple d'architecture du dossier src - **page 16**

Figure 6 : Interface de VSCode - **page 18**

Figure 7 : Interface PhpStorm - **page 19**

Figure 8 : Interface phpMyAdmin - **page 20**

Figure 9 : Interface Adminer - **page 20**

Figure 10 : Exemple de connexion SSH - **page 23**

Figure 11 : Fichiers log - **page 23**

Figure 12 : Exemple de ligne d'un fichier log - **page 24**

Figure 13 : Exemple de fichier crontab - **page 25**

Figure 14 : Code du template de mails avant réarrangement - **page 26**

Figure 15 : Code du template de mails après réarrangement - **page 26**

Figure 16 : Tableau regroupant les informations des mails - **page 27**

Figure 17 : Interface Mailtrap - **page 28**

Figure 18 : Interface administrateur avec les barre de recherche - **page 30**

Figure 19 : Système de pagination - **page 30**

Figure 20 : Fonctionnement du Webhook *invoice.payment_failed* - **page 32**

Figure 21 : Logique du Webhook *invoice.payment_failed* après modification - **page 33**

Lexique

1. **TPE/PME (Très petite, Petite et Moyenne Entreprise)** : est une entreprise dont la taille, définie à partir du nombre d'employés, du bilan ou du chiffre d'affaires, ne dépasse pas certaines limites - [page 2](#)
2. **Framework** : Un framework est un ensemble cohérent de composants logiciels structurels qui sert à créer les fondations ainsi que les grandes lignes de tout ou partie d'un logiciel, c'est-à-dire une architecture - [page 2](#)
3. **SSH (Secure Shell)** : SSH est un protocole qui permet d'établir des connexions chiffrées à distance entre des ordinateurs - [page 2](#)
4. **API (Application Programming Interface)** : Une API est un ensemble normalisé de classes, de méthodes, de fonctions et de constantes qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels - [page 4](#)
5. **VoIP (Voice over Internet Protocol)** : VoIP est une technique informatique qui permet de transmettre la voix sur des réseaux compatibles IP, via Internet ou des réseaux privés ou publics, qu'ils soient filaires ou non - [page 8](#)
6. **IP (Internet Protocol)** : Une adresse IP est un numéro d'identification unique attribué de façon permanente ou provisoire à chaque périphérique faisant partie d'un même réseau informatique utilisant l'Internet Protocol - [page 8](#)
7. **IDE (Integrated Development Environment)** : Une IDE est un ensemble d'outils qui permet d'augmenter la productivité des programmeurs qui développent des logiciels - [page 10](#)
8. **URL (Uniform Resource Locator)** : Une URL est une chaîne de caractères qui indique sous une forme standardisée comment accéder à une ressource du World Wide Web à travers Internet - [page 11](#)
9. **SCRUM** : SCRUM est définie comme un cadre de travail holistique itératif qui se concentre sur les buts communs en livrant de manière productive et créative des produits de la plus grande valeur possible - [page 12](#)
10. **Esprit agile** : Les pratiques agiles mettent en avant la collaboration entre des équipes auto-organisées et pluridisciplinaires et leurs clients - [page 12](#)

- 11. MVC (*Model View Controller*) :** MVC est un motif d'architecture logicielle destiné aux interfaces graphiques, Le motif est composé de trois types de modules ayant trois responsabilités différentes : les modèles (contient les données à afficher), les vues (contient la présentation de l'interface graphique) et les contrôleurs (contient la logique concernant les actions effectuées par l'utilisateur) - [page 14](#)
- 12. HTTP (*Hypertext Transfer Protocol*) :** HTTP est un protocole de communication client-serveur développé pour le World Wide Web - [page 16](#)
- 13. SQL (*Structured Query Language*) :** SQL est un langage informatique normalisé servant à exploiter des bases de données relationnelles - [page 21](#)
- 14. SMTP (*Simple Mail Transfer Protocol*) :** SMTP est un protocole de communication utilisé pour transférer le courrier électronique vers les serveurs de messagerie électronique - [page 27](#)
- 15. CTRL + F :** CTRL + F est un raccourci clavier utilisé pour rechercher du texte dans un document ou une page web - [page 29](#)