Rapport d’alternance

25/08/2024

Développeur Web

Tristan SCHMITT

Epitech Mulhouse

30 Rue François Spoerry

68100 Mulhouse

PL Diffusion

ZA Les Erlen, 2 Rue des Merisiers

68920 Wettolsheim

Rapport d'alternance

Développeur Web

Tristan SCHMITT

Du 18 septembre 2023 au 8 septembre 2024

Tuteur en centre de formation : Mr Stanislas HEGRON, Pédagogue chez Epitech Mulhouse

Tuteur d'alternance : Mr Geoffroy BARBARAS, Chef de projet Tempo Infini chez PL Diffusion

Remerciements

Premièrement, je souhaite remercier mon chef de projet et tuteur Geoffroy Barbaras pour sa patience, ses conseils et son ouverture. Benjamin Geiss pour sa bonne humeur, son envie de partager et ses défis techniques. Romain Hoang, pour sa constance et son expertise. Tout au long de mon alternance, je me suis senti écouté et ai pu bénéficier de leurs expériences. J'ai été encouragé à faire des propositions et ai reçu des retours constructifs qui m'ont grandement aidé à progresser.

Je remercie aussi mes amis de formation, Sofiane Arif, Anthony Limon et Maksym Prokopovytch avec lesquels j'ai réalisé la majorité de mes projets au cours de ces deux années. Merci à eux pour m'avoir donné le plaisir de venir assister à la formation, pour avoir reconnu mon travail et ma passion ainsi que l'entraide dont nous avons fait preuve.

Enfin je remercie Stanislas Hegron et Véronique Gingras pour leur suivi au cours de ma formation chez Web@cademy by Epitech Mulhouse.

Table des matières

Introduction

Ce rapport est le résultat de deux ans de formation chez Web@cademy, Epitech Mulhouse dans le but d'obtenir la certification professionnelle de Développeur Web.

Dans le contexte de cette formation ayant débutée le 21 novembre 2022, j'ai eu le plaisir de travailler chez PL Diffusion, entreprise membre d'ENSO Groupe à Wettolsheim, en tant que Développeur Full-stack du 18 septembre 2023 au 6 septembre 2024.

Au cours de ce rapport, je vous présenterai mon parcours, mes missions et partagerai avec vous mon ressenti vis-à-vis de mon expérience cette année.

Présentation de l'entreprise

Historique

PL Diffusion est un éditeur de logiciel Enterprise Ressource Planning (ERP ci-après) « métier », destiné aux Entreprises de Travail Temporaire (ETT ci-après) fondé en 1994 par Isabelle Melin et Patrick Lidy.

L'objectif était d'informatiser la gestion des ETT et en 1995, le logiciel fut adopté par la première ETT. D'ici 1998, PL Diffusion équipa cent agences, démontrant sa capacité à fournir une solution de qualité.

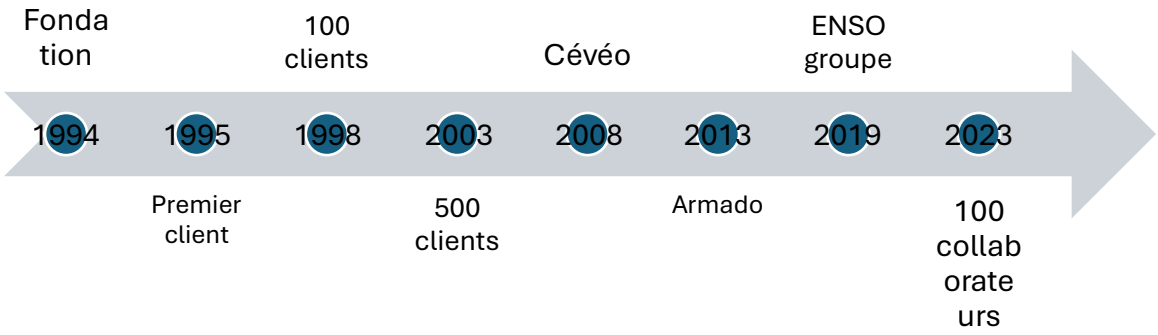
La clientèle s'étend rapidement et la cinq-centième agence est équipée de la solution en 2003.

2008 vient marquer le lancement de Cévéo. Une solution complémentaire à Tempo qui permet de référencer les CV des intérimaires afin de faciliter la recherche d'intérimaire pour répondre aux besoins des Entreprises Utilisatrices (EU ci-après).

En 2013, PL Diffusion publie Armado, une plateforme de gestion documentaire pour les agences intérim. Cette solution simplifie et sécurise les processus administratifs. Parmi les fonctionnalités du logiciel, la possibilité de signer électroniquement les contrats ainsi que la dématérialisation des fiches de paie.

Le chiffre d'affaires de PL Diffusion en 2019 s'élève à 7,53M d'euros. Les associés historiques créent la société de holding ENSO Groupe en prévision d'une forte croissance.

Arrivé 2023, la société ENSO Groupe a atteint les 100 collaborateurs (salariés) et continue de fournir des solutions de pointes aux ETT. En effet, ENSO possède plus de 60% des parts du marché de solution ERP « métier ».



Activités principales

En France, le travail est une activité encadrée par la loi. Pour exercer, il convient de signer un contrat de travail. Parmi les multiples types de contrats figure le contrat de travail temporaire.

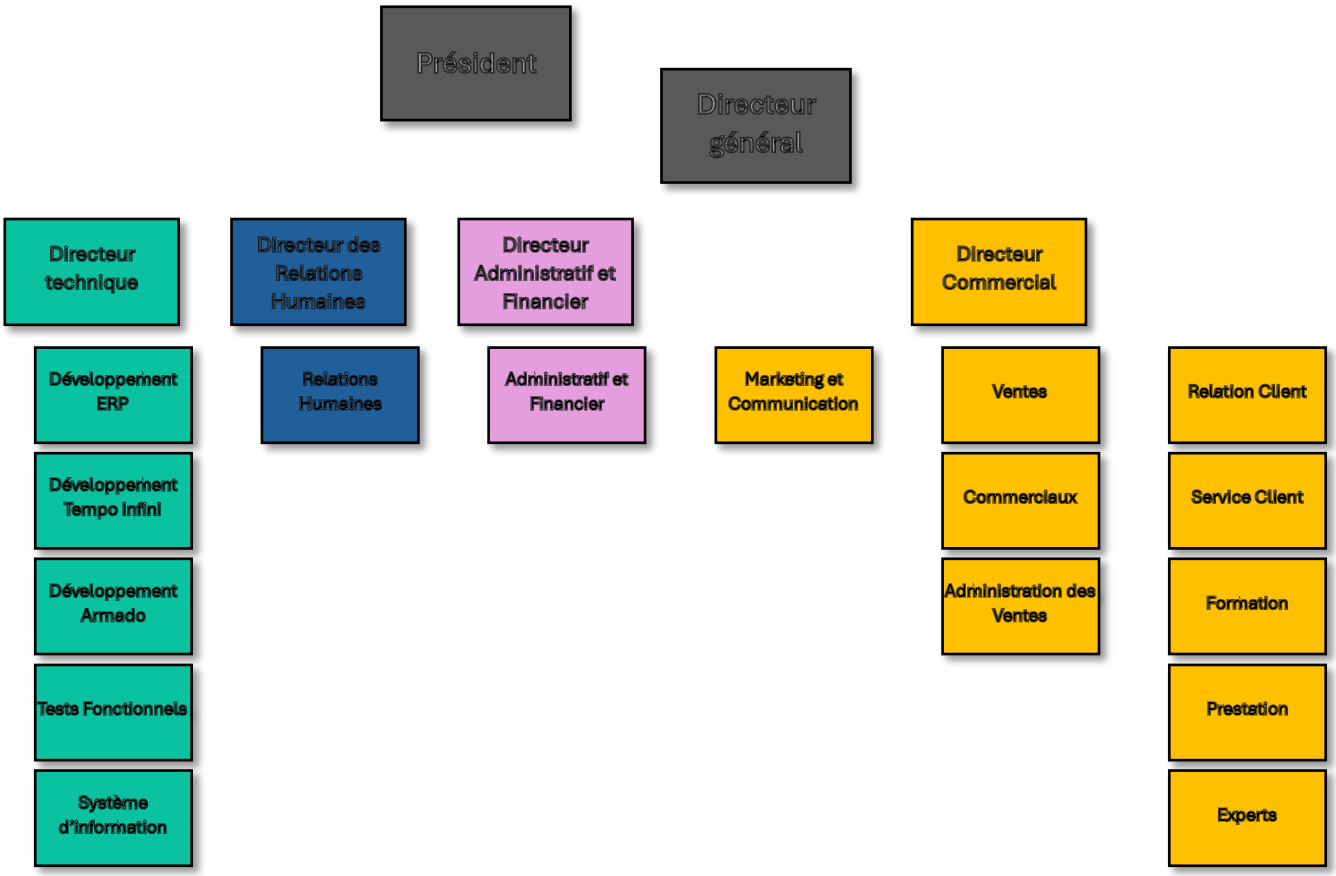
L’intérim (travail temporaire) consiste à mettre à disposition provisoire d’entreprises clientes, des salariés qui, en fonction d'une rémunération convenue, sont embauchés et rémunérés à cet effet par l’ETT.

L'intérim se caractérise donc par une relation triangulaire entre l’ETT, l’EU et le salarié. L’intérim implique la conclusion de deux contrats : un contrat de mise à disposition (entre l’ETT et l’EU) et un contrat de mission (entre l’ETT et le salarié).

Le contrat ne peut être conclu que pour l’exécution d’une tâche précise et temporaire, dénommée mission, et seulement dans les cas énumérés par la loi. Quel que soit le motif pour lequel il est conclu, un tel contrat ne peut avoir ni pour objet ni pour effet de pourvoir durablement un emploi lié à l’activité normale et permanente de l’entreprise utilisatrice.

ENSO Groupe, dont PL Diffusion est membre, apporte une suite de solutions logiciels pour faciliter l’exercice des entreprises de travail temporaire. Dans le but de faciliter l’adoption de ces logiciels, des formations couvrant l’utilisation des différents logiciels et les implications légales des activités concernées, ainsi qu’un support continue, sont proposés. ENSO va même jusqu’à l’accompagnement dans l’ouverture d’une agence d’intérim et la formation de ses employés au monde de l’intérim.

Organigramme



Projets principaux

Tempo

Tempo propose une gamme complète de solution de gestion permettant d'optimiser les processus des agences d'intérim.

Parmi les fonctionnalités clefs :

- Gestion des commandes :
  - Gérer de manière rapide et efficace les demandes de personnel des EU.
  - Créer les commandes et les pourvoir via une recherche de candidats disponibles et qualifiés.
  - Assurer une gestion optimale des missions et des contrats.
- Suivi des intérimaires
  - Créer des fiches complètes pour chaque candidat grâce à une solution de gestion des talents.
  - Centraliser les données essentielles des candidats, comme leurs compétences, leurs disponibilités, leur historique de mission et bien d'autres informations.
  - Faciliter le recrutement, la sélection et le placement des intérimaires.
- Gestion des factures
  - Personnaliser, générer et centraliser les factures pour une gestion simplifiée.
  - Permettre un suivi du chiffre d'affaires et des règlements clients, offrant un aperçu clair des performances financières.
- Traitement de la paie
  - Générer avec précision les paies des salariés intérimaires ainsi que des permanents (agents d'agence).
  - Gérer les tâches liées à la paie, telles que la saisie des variables, l'édition des journaux de paie, la génération de justificatifs annuels, jusqu'à la Déclaration Sociale Nominative (DSN).
- Échange rapide
  - Contacter des clients et intérimaires par SMS grâce à Contact+ en cas de recrutement ou d'imprévus.
  - Saisissez vos relevés d'heures, factures et autres documents pour gagner du temps dans vos échanges.
- Édition de statistiques
  - Faciliter la génération de rapports répondant aux besoins administratifs grâce à un outil d'édition intégré.
  - Offrir la possibilité d'exporter des rapports en toute simplicité une fois qu'ils sont créés.

Armado

Armado est la solution de dématérialisation du groupe ENSO. Elle offre une gestion moderne des documents administratifs dans le contexte de l'intérim.

Parmi les fonctionnalités clefs :

- Centralisation et dématérialisation des contrats
  - Retrouver l'ensemble des contrats, qu'ils soient en cours, signés ou complétés, dans une interface centralisée.
  - Permettre la dématérialisation des documents et des contrats, la réception automatique des signatures et le stockage sécurisé des documents.
- Faciliter la gestion des intérimaires
  - Inviter les intérimaires à rejoindre une agence via un processus simplifié.
  - Permettre la diffusion de QR codes en agence, le partage par SMS ou courriel, la saisie des heures et absences par les clients et intérimaires.
  - Faciliter l'intégration des intérimaires et centraliser toutes les informations nécessaires à leur gestion dans un même outil de dématérialisation.
- Sécuriser et optimiser la gestion des documents
  - Assurer la sécurité informatique et la confidentialité des données avec des technologies avancées de sécurité.
  - Relancer automatique des signatures et du stockage sécurisé de tous vos documents contractuels et administratifs.
- Communication avec les intérimaires et clients
  - Messagerie instantanée contextuelle et une application mobile dédiée pour les intérimaires.
  - Envoyer des notifications, partager des documents et gérer les communications en temps réel.

Cévéo

Cévéo est une solution de centralisation des Curriculum Vitae (CV) des intérimaires.

Parmi les fonctionnalités clefs :

- Centralisation des candidatures
  - Créer facilement un vivier de candidats en important leurs CV et en analysant leur contenu.
  - Éliminez les saisies manuelles.
- Recommandation des meilleurs profils
- Communication fluide et ciblée
  - Localiser les clients et intérimaires en quelques secondes.
  - Proposer des opportunités (candidats, propositions actives) en un seul clic.

- Communiquer facilement grâce aux modèles de messages personnalisables.
- Envoyer des courriels et des SMS à l'aide du service complémentaire Contact+.

- Gestion de l'activité en toute simplicité
  - Gagner du temps avec des outils de traitement en masse.
- Conformité légale assurée
  - Respect des réglementations en vigueur, avec une conformité rigoureuse au Règlement Général sur la Protection des Données (RGPD).
  - Les CV sont anonymisés et les droits des candidats sont scrupuleusement gérés.
  - Des mesures de sécurité organisationnelles et techniques garantissent la protection des données.

Objectifs de la mission

Contexte

Tempo, le logiciel principal de PL Diffusion, est un logiciel vieillissant. Il a été développé et maintenu avec WinDev, un « atelier de génie logiciel » édité par la société française PC SOFT. WinDev est publié sous une licence propriétaire, annuelle, onéreuse qui donne accès à des mises à jour jusqu'au terme de ladite année.

Au début du développement de Tempo, WinDev ne supportait que le WLangage, un langage de programmation lui aussi développé par PC SOFT. Le langage de programmation étant l'interface entre le développeur et la machine, il est l'outil le plus important pour permettre à un développeur d'implémenter une fonctionnalité demandée par un client.
Cependant, WLangage ne bénéficie pas d'une entreprise propriétaire de taille comparable à Microsoft ou Google. Plusieurs fonctionnalités d'autres langages de programmation modernes tels que C# (prononcé C Sharp) prennent plusieurs années à être ajoutées à WLangage.
Enfin, le WLangage est lui aussi propriétaire et effectivement une boîte noire. Les développeurs Tempo utilisent ce qu'on appelle une Application Programming Interface (API) qui expose les fonctionnalités du langage, mais il est impossible de vérifier que le code fait exactement et seulement ce que le nom ou la documentation indiquent autrement qu'en le testant extensivement.

En considération des points évoqués et dans le but de proposer une solution moderne, robuste, réactive et évolutive à ses clients, ENSO Groupe a fait le choix de réécrire la totalité de la solution Tempo en C# pour la partie serveur et JavaScript pour la partie client.

C# est un langage de programmation développé par Microsoft. Il est open-source, ce qui signifie que n'importe qui peux consulter et contribuer au code source du langage. Il fournit stabilité et performance, est très bien documenté et jouit d'une des plus grandes communautés de développeur.
Cela en fait un langage de choix pour le développement du serveur de la solution.

Pour l'interface client, le choix s'est naturellement porté sur l'ensemble HTML (HyperText Markup Language), CSS (Cascading Style Sheets) & JavaScript. Ces trois technologies fonctionnent en tandem pour délivrer des interfaces simples et modernes via les navigateurs internet.
Il est ainsi possible de délivrer une solution multiplateforme accessible à l'aide d'une connexion internet depuis presque n'importe quel support, ce qui est très demandé aujourd'hui.

La réécriture du logiciel a été démarrée en 2018 et devrait prochainement passer en phase de production.

Objectifs à atteindre

ENSO Groupe souhaite publier la nouvelle version de Tempo, nommée Tempo Infini en interne, d'ici janvier 2025.

Le déploiement aura lieu en plusieurs phases. Premièrement auprès de clients « pilotes » contactés par ENSO Groupe, auxquels il a été proposé de migrer vers la nouvelle version du logiciel dans le but de récolter des retours.

Pour mener à bien cette phase, il est important d'implémenter la totalité les fonctionnalités nécessaires à l'exercice quotidien des ETT, les tester et s'assurer de leur stabilité afin de ne pas perturber leur continuité de service.

Le modèle de données de l'application, c'est-à-dire la forme que prennent les différentes données en bases de données telles que les informations permettant d'identifier un utilisateur, a changé entre l'ancienne et la nouvelle application.
Il est impératif de fournir une solution de migration qui permette aux clients de garder leurs données existantes et de pouvoir les utiliser dans la nouvelle application.

Contraintes et défis spécifiques

Les ETT sont représentées par des établissements. Une entreprise peut posséder plusieurs établissements. De même, plusieurs entreprises peuvent faire partie d'un même groupe.

- L'application doit être configurable avec des politiques appliquées à l'entièreté d'un groupe ou à un établissement particulier.
- Des rôles utilisateurs et leurs droits doivent être configurables au niveau du groupe et/ou de l'établissement.
- Certaines données peuvent être partagés à l'ensemble des ETT, indifféremment de leur groupe tandis que d'autres sont exclusives à un groupe ou un établissement.

Le domaine du travail temporaire est encadré par une multitude de lois et il est compliqué de garder à l'esprit leurs implications et les différentes exceptions.

- Cela nécessite la consultation d'experts juridiques et comptables.
- Le temps de développement peut être rallongé par les inconnus et les délais induit par les échanges entre les différents services de développement.
- Il convient de délimiter les responsabilités des développeurs et des utilisateurs dans le développement des fonctionnalités et leurs utilisations.
- Le logiciel doit suppléer l'utilisateur dans le bon exercice de ses fonctions à l'aide d'auto-remplissages et de suggestions légalement correctes tout en restant permissif pour ceux souhaitant passer outre ces recommandations.

ENSO Groupe possède actuellement la majorité du marché ERP client pour les ETT avec plusieurs clients de longue date.

- Il faudra gérer la migration simultanée et/ou différée des clients existants et les accompagner dans l'adoption de la nouvelle application.
- Les fonctionnalités considérées essentielles par ENSO Groupe peuvent ne pas couvrir la totalité des besoins de chaque client.
- Certains clients ont actuellement des fonctionnalités personnalisées. Ces fonctionnalités peuvent ne pas être réimplémentées.

Projets réalisés

Tâches

Défis d'introduction à Angular & RxJS.

Ma première tâche technique dès mon arrivée en entreprise fut de suivre les défis élaborés par le Lead Developper afin de me familiariser avec les technologies utilisées pour le développement de l'application cliente.

Angular est un framework de développement front-end qui utilise la librairie RxJS pour gérer la réactivité de l'application ainsi que les traitements asynchrones.

C'est au cours de cette semaine que j'ai appris à porter une attention particulière au cycle de vie d'un objet en programmation. Ces exercices m'ont aussi formé à une gestion plus poussée de code asynchrone.

Ces connaissances m'ont servi tout au long de mon alternance en plus de grandement faciliter l'adoption du framework Vue quand nous avons changé de technologie pour la partie cliente de notre application.

Guides d'apprentissage C#, .NET, Entity Framework Core & Microservices de Microsoft

Je n'avais auparavant jamais programmé en C#. Geoffroy m'a sélectionné plusieurs guides écrits par Microsoft pour introduire les développeurs à leurs technologies.

J'ai, sur une durée de deux semaines, réalisé plusieurs projets qui m'ont initié à l'utilisation du langage C# dans un environnement de développement .NET, couplé à Entity Framework Core et suivant une architecture Microservices.

Cela forma la base de mes connaissances pour contribuer au développement de l'application serveur du projet Tempo Infini.

Implémenter un système de notification utilisant Signal R

Une fois les bases de connaissances acquises pour développer sur les deux fronts de l'application, j'ai pu entamer le développement de ce qui serait, ou aurait dû, être mon sujet de développement au long de mon alternance.

Le but était de développer un système de notification utilisant Signal R, basé sur des web sockets. Ces notifications seraient publiées en temps réel à la suite d'évènements. Ce système devait être souple et extensible, de manière qu'il puisse facilement se greffer sur les fonctionnalités futures.

La première étape était de permettre à deux utilisateurs de pouvoir s'envoyer des messages en direct à travers l'interface de l'application.

Je n'ai cependant pas eu l'occasion de mener cette étape à bout. L'apprentissage des architectures des applications cliente et serveur ainsi que des technologies dont elles faisaient usage m'a pris trop de temps.

CRUD référentiel de rubriques de paie

Peu après avoir rejoint l'entreprise, la direction de l'entreprise a fait le choix d'homogénéiser les technologies de développement utilisées par les équipes Armado et Tempo Infini.

L'équipe Armado ayant déjà publiée une réécriture de leur logiciel avec des technologies modernes et ayant fait leurs preuves, l'équipe Tempo Infini a été contraint de migrer son application cliente de Angular vers Vue et d'adapter l'architecture en micro-services de son serveur vers une architecture mono-service.

Dans ce contexte, j'ai travaillé sur des tâches à difficulté croissante pour me familiariser (à nouveau) avec l'architecture et les technologies des applications.

La première de ces tâches était un CRUD basique pour alimenter un référentiel de rubriques de paie.
Les référentiels sont des sources de données utilisées à travers l'application par différentes fonctionnalités. Ils sont principalement lus par le reste de l'application pour assurer la cohérence des données et fournir l'autocomplétion. Les opérations d'ajouts, d'éditions et de suppressions sont limitées.

Permettre de configurer l'application avec des politiques de groupe et/ou d'établissement

Les agences d'intérim peuvent avoir des gestions particulières. Dans le but de rester souple et s'adapter au mieux aux besoins des agences, l'application Tempo peut être configurée avec des politiques de groupe et/ou d'établissement.

Nous avons trois entités dans notre application :

1. Groupe ;
2. Entreprise ;
3. Etablissement.

Un groupe rassemble plusieurs entreprises, auxquelles plusieurs établissements sont rattachés.

Lors de la création d'un groupe, une configuration est automatiquement créée et rattachée à ce groupe. De même, lors de la création d'une entreprise.

Plusieurs catégories de configuration existent : Client, Intérimaire, Paie, Facturation, etc...

Lorsqu'une fonctionnalité de l'application dépend d'une configuration, il convient de récupérer ce que l'on appelle une configuration applicable.

La configuration applicable est le résultat de la comparaison entre une configuration de groupe et celle de l'établissement. Chaque champ des configurations est comparé individuellement sachant que la configuration de l'établissement prime sur celle du groupe.

En sachant cela les configurations suivantes :

Configuration Client	Groupe	Établissement
Siret obligatoire	true	false
Code APE obligatoire		true
Adresse obligatoire	false	
Téléphone obligatoire		

Produisent la configuration applicable suivante :

Configuration Client	
Siret obligatoire	false
Code APE obligatoire	true
Adresse obligatoire	false
Téléphone obligatoire	

J'ai implémenté cette fonctionnalité en greffant une création automatique d'une configuration à la création d'un groupe et d'un établissement tout en assurant qu'une seule configuration par catégorie puisse exister pour l'un comme pour l'autre.

Ces configurations sont modifiables depuis l'interface de l'application cliente.

La configuration applicable est quant à elle facilement récupérables par les différents services de l'application en fournissant simplement le code de la catégorie désirée.

Implémenter un algorithme de calcul pour les dates de souplesses d'un contrat intérimaire

Un contrat d'intérim bénéficie de ce que l'on appelle dates de souplesses. Ces dates sont déterminées en fonctions des termes et de la durée du contrat. Elles permettent à une EU de clôturer un contrat avant la date de fin prévisionnelle ou de l'étendre au-delà.

L'application du calcul de la souplesse est encadrée par la loi et peut avoir de lourdes conséquences. Notamment le passage automatique d'un contrat d'intérim en contrat à durée indéterminée si l'intérimaire travaille plus de 18 mois au cours d'un même contrat.

Vous trouverez en annexe le code source permettant de déterminer ces dates. Ceci fut un excellant exercice pour développer ma capacité à produire un code clair et facilement corrigeable. Ceci est le résultat de plusieurs refactorisations afin que le code fasse office de documentation.

Pour la réalisation de cette tâche, plusieurs clarifications étaient nécessaires de la part des experts.
Il était impératif de faire la différence entre la durée du contrat et le nombre de jours travaillés. Il était aussi nécessaire de prendre en compte le fait qu'un intérimaire puisse travailler un jour férié, le week-end, les différences de jours fériés entre la France métropolitaine, l'Alsace, les territoires d'outre-mer et enfin, le jour de solidarité.

Aussi, une date de souplesse, négative comme positive, ne peut tomber sur un jour non-ouvrable.

J'ai aussi créé plusieurs tests unitaires afin de m'assurer de l'exactitudes de mon algorithme suivant les cas suivants :

- Un contrat d'une durée de dix jours calendaires ou moins doit avoir une souplesse d'une durée de deux jours.
- Au-delà d'une durée de dix jours calendaires. La souplesse est calculée à un taux d'un jour pour cinq jours travaillés.
- La souplesse négative a une valeur maximale de dix jours travaillés.
- Une souplesse négative ne peut pas avoir une date antérieure à la date de début d'un contrat.
- Une souplesse positive ne peut excéder la durée maximale d'un contrat d'intérim étant de dix-huit mois.
- Une souplesse ne peut tomber sur un jour non-ouvrable.
- Les jours fériés changent suivant les années ou le lieu d'exercice d'une agence intérim.

Recherche par téléphone et/ou mail d'intérimaires existants

Au cours du développement, les formateurs de l'entreprise sont amenés à tester l'application au fur et à mesure du développement. Les formateurs étants au contact direct des clients, ils ont une attention particulière à l'ergonomie et l'aspect fonctionnel de l'application.

Un des premiers retours que l'on a eu de leur part mentionner le besoin de pouvoir rechercher un intérimaire grâce à son numéro de téléphone ou son courriel.

Il était déjà possible d'effectuer une recherche par nom et prénom. Ceci fut donc ma première extension d'une fonctionnalité existante.

J'ai pour cela consulté le code produit par un collègue et ajouté le strict nécessaire afin de filtrer les intérimaires avec des critères additionnels.

La difficulté principale de cette tâche était la modification de l'interface utilisateur pour ajouter les champs tout en conservant l'ergonomie. J'ai collaboré avec notre designer pour trouver une solution adéquate.

Ajout d'un taux de maintien de salaire au référentiel d'organismes

Un autre développement simple et rapide lors duquel j'ai étendu une fonctionnalité existante en ajoutant un champ.

Génération automatique d'un code client unique

Lors de la création d'un établissement, un code client lui est automatiquement attribué. Ce code client figurera sur les contrats créés par ledit établissement. Dans cette optique, le code doit être unique à l'établissement.

Il convenait de réimplémenter un algorithme de génération présent dans l'application précédente sous forme de procédure stockée. Une procédure stockée est la représentation d'un traitement répétable en langage SQL. Ce traitement était effectué par la base de données.

Afin de centraliser la logique de l'application, j'ai traduit ce traitement SQL en langage C# afin qu'il soit effectué par le serveur.

Permettre d'envoyer des Demande Préalable À l'Embauche (DPAE) à l'API du gouvernement et d'en consulter le traitement

À la suite de la création d'un contrat de travail temporaire, il est nécessaire de faire une Demande Préalable À l'Embauche auprès du service de l'URSSAF.

Il est possible d'effectuer cette demande à travers l'API web de l'URSSAF. Pour cela, j'ai récupéré les informations demandées par l'API depuis la base de données et transformé ces données en format « Extensible Markup Language » (XML).

Ces données sont ensuite compressées et envoyées à l'URSSAF. Un identifiant est renvoyé par l'API permettant de consulter l'état du traitement de la DPAE.

Dans le cas où cette demande n'est pas effectuée, des répercussions légales peuvent encourir. Un intérimaire ayant commencé sa mission sans avoir été correctement déclaré à l'URSSAF ne sera par exemple pas assuré.

Dû à la complexité de ce traitement et l'importance de cette déclaration, j'ai porté une attention particulière à l'enregistrement des dates d'émissions des DPAE, les dates d'enregistrement auprès de l'URSSAF, les différents codes http des requêtes d'authentification auprès du service, suivi des requêtes de dépôt.

J'ai aussi automatisé les requêtes de vérification du traitement des DPAE en plus de permettre à l'utilisateur final d'initier une relance de ce traitement.

Refactoriser le traitement d'acomptes

Il est courant en intérim d'effectuer des acomptes. Un acompte est un versement anticipé du salaire de l'intérimaire par l'agence d'intérim l'employant.

Ces acomptes peuvent être réalisés en fonction des relevés d'heures de l'intérimaire ou des heures convenues dans le contrat. On parle d'acomptes réels et d'acomptes théoriques.

J'ai repris le développement de cette fonctionnalité après la découverte d'une irrégularité entre les montants totaux et les détails affichés par l'application.

Un premier développeur a implémenté un algorithme permettant de faire le total des acomptes réels et théoriques possible pour un intérimaire. Par la suite, un autre développeur a implémenté la possibilité de visualiser le détail du calcul de ces montants.
Or, les détails ne coïncidaient pas avec les montants totaux.

Lors de mon investigation du code source, j'ai identifié deux services différents répondant à ces besoins. Avoir deux services distincts effectuant en majorité le même traitement, avec les mêmes données, est difficilement maintenable.
Si le traitement venait à changer, les développeurs devraient s'assurer de changer ce traitement dans les deux services.

J'ai donc supprimé le service des détails et refactorisé celui fournissant les montants totaux de manière que les détails puissent être extrait et renvoyés au client.

Ainsi, un seul service est responsable de calculer les acomptes possibles et est capable de fournir le détail de ces acomptes ou les montants total au besoin.

Ceci était un exercice intéressant mais aussi difficile. Il était nécessaire de comprendre en détail le code produit par d'autres développeurs et ne pas en changer le comportement final.

Synchronisation des référentiels de l'application avec les référentiels d'une application master

Les référentiels comportent des informations indépendantes des agences d'intérim, telles que les communes de France, les codes APE permettant d'identifier les domaines d'activités des entreprises, etc...

Dans un souci de fournir des informations à jours à toutes les agences, nous avons mis en place une application maître exposant une API permettant de récupérer le contenu de ces référentiels.

Les référentiels étant utilisés pour assurer l'intégrité des données à l'aide de clefs étrangères dans une base de données relationnelles, il est nécessaire que leur contenu soit enregistré dans la base de données de l'application même.

Ainsi j'ai repris le développement d'une logique de mise à jour périodique des référentiels. Geoffroy ayant mis en place une ébauche fonctionnelle, j'ai générisé et généralisé le comportement pour les autres référentiels.

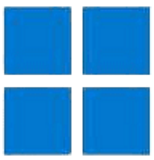
Lors de cette généralisation aux autres référentiels, deux problèmes majeurs sont survenus.

Premièrement, certains référentiels ont des relations les uns avec les autres. Il était nécessaire de trouver un moyen pour résoudre ces relations.
Dans une base de données relationnelles, chaque entrée est généralement identifiée par un nombre auto-incrémenté. Ce nombre peut différer entre la base de données maître et les bases de données cibles.
Pour répondre à cette problématiquement, j'ai ajouté un champ dans les bases de données cibles contenant l'identifiant de l'entrée dans la base de données maître, permettant de résoudre les relations à l'aide de ce champ.
Cette solution a aussi permis de créer des référentiels hybrides, permettant aux utilisateurs finaux d'ajouter leurs propres données sans impacter les entrées synchroniser avec l'application maître.

Deuxièmement, les opérations de mises à jour prenaient énormément de temps. Un référentiel avec plus de 30 000 entrées prenait dix-huit minutes à mettre à jour.
J'ai donc optimisé le traitement de manière à réduire cette durée à 30 secondes.
Cela était en majeure partie dû à une librairie que nous utilisons pour la communication avec la base de données. La méthode utilisée recherchait chaque entrée individuellement pour la mettre à jour au lieu d'appliquer un traitement de groupe.

Finalement, j'ai réussi à optimiser et automatiser cette fonctionnalité en créant un job périodique sur les applications cibles afin de récupérer les référentiels à jours depuis l'application maître.

Technologies et outils utilisés



Windows 11

Le dernier système d'exploitation de Microsoft, offrant un environnement sécurisé et efficace pour le développement. Améliore les performances, la compatibilité avec les outils de développement et les fonctionnalités multitâches.

Teams

Microsoft Teams est une plateforme de collaboration pour le chat, les réunions vidéo et le partage de fichiers. Il s'intègre parfaitement avec d'autres outils Microsoft, facilitant la communication et la collaboration au sein des équipes de développement.

Outlook

Microsoft Outlook est un client de messagerie et de calendrier utilisé pour gérer les communications et la planification. Il aide les équipes de développement à organiser des réunions, suivre les projets et maintenir une correspondance professionnelle.

GitLab

GitLab est une plateforme DevOps pour le contrôle de version, les pipelines CI/CD et la collaboration. Il permet aux équipes de développement de gérer les dépôts de code, d'automatiser les tests et de simplifier les déploiements.

C#

C# est un langage de programmation moderne et orienté objet développé par Microsoft. Il est couramment utilisé pour développer des applications Windows, des services web et des jeux, offrant un cadre solide pour des logiciels évolutifs.

DotNet Core (.NET Core)

.NET Core est un cadre de développement (framework) gratuit et multiplateforme pour créer divers types d'applications. Il prend en charge plusieurs langages (dont C#), augmentant la productivité et permettant des solutions logicielles polyvalentes.

HyperText Markup Language (HTML)

HTML est le langage standard pour créer des pages web. Il structure le contenu sur le web, formant la base du développement de sites web.

Cascading Style Sheets (CSS)

CSS est utilisé pour styliser et mettre en page les pages web. Il améliore l'apparence des éléments HTML, permettant aux équipes de développement de créer des sites web attrayants et réactifs.

JavaScript

JavaScript est un langage de script qui permet des fonctionnalités web interactives. Il est essentiel pour le développement front-end, permettant aux équipes de développement de créer des applications web dynamiques et conviviales.

TypeScript

TypeScript est un sur-ensemble de JavaScript qui ajoute le typage statique. Il aide les équipes de développement à détecter les erreurs tôt dans le processus de développement et améliore la maintenabilité et la scalabilité du code.

Vue

Vue est un framework JavaScript progressif utilisé dans le développement d'applications web pour créer des interfaces utilisateur dynamiques et réactives. Il permet de structurer l'application en composants réutilisables, facilitant la gestion et la maintenance du code.

VsCode

Visual Studio Code est un éditeur de code léger et open-source de Microsoft. Il prend en charge divers langages de programmation et extensions, en faisant un outil polyvalent pour le développement.

WebStorm

WebStorm est un environnement de développement intégré (IDE) par JetBrains, spécifiquement pour JavaScript et les technologies associées. Il offre une assistance au codage puissante, un débogage et une intégration avec divers outils pour le développement web.

Rider

Rider est un IDE .NET multiplateforme par JetBrains. Il fournit une assistance avancée au codage, un débogage et une intégration avec les technologies .NET, augmentant la productivité des développeurs C# et .NET.

Orion UI

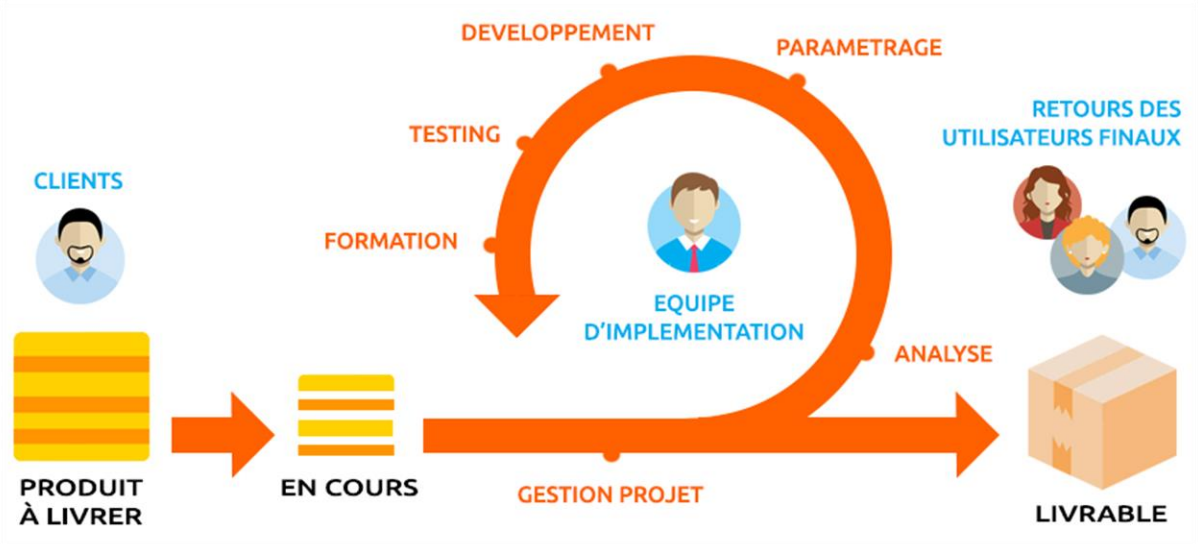
Orion UI est une bibliothèque d'interface utilisateur pour construire des applications web modernes. Elle fournit des composants et des outils préconçus qui aident les équipes de développement à créer des interfaces utilisateurs cohérentes et réactives efficacement. Elle est développée par des collaborateurs membres de l'équipe Armado.

Docker

Docker est une plateforme de conteneurisation qui permet de créer, déployer et exécuter des applications dans des conteneurs. Il aide les équipes de développement à garantir la cohérence des environnements, simplifiant ainsi le déploiement et l'évolutivité des applications.

Processus de développement

Au sein de l'équipe de développement Tempo Infini, nous suivons la méthodologie Agile. Agile favorise une approche itérative et incrémentale, permettant aux équipes de développement de réagir rapidement aux changements et aux besoins des utilisateurs.



Cette méthodologie repose sur la collaboration étroite entre les équipes, les feedbacks constants et l'amélioration continue. En adoptant Agile, nous pouvons rapidement identifier et résoudre les problèmes, nous assurer que le produit reste aligné avec les attentes des utilisateurs et nous adapter aux nouvelles exigences du marché du travail temporaire.

Le processus commence par la définition des fonctionnalités essentielles qui apportent le plus de valeur aux utilisateurs. Ces fonctionnalités prioritaires sont souvent basées sur les besoins critiques des entreprises de travail temporaire, tels que la gestion des contrats, la planification des ressources, et le suivi des heures travaillées.

Geoffroy (notre chef de projet) collabore avec le chef produit et le directeur technique pour déterminer quelles fonctionnalités doivent être développées en premier. Ce processus de priorisation garantit que les éléments les plus importants et les plus urgents sont abordés dès le début, permettant aux testeurs, experts et formateurs de bénéficier rapidement des fonctionnalités essentielles.

Les testeurs peuvent ainsi vérifier la stabilité de l'application et le respect du cahier des charges. Les experts vérifient l'exactitudes des données telles que la facturation. Enfin les formateurs peuvent donner des retours sur l'ergonomie de l'application.

Une fois les fonctionnalités prioritaires identifiées, Geoffroy les décompose en tâches spécifiques et détaillées qui sont ajoutées au backlog du produit. Le backlog est une liste dynamique de toutes les fonctionnalités, améliorations et corrections de bugs nécessaires pour le développement de l'application. Chaque tâche du backlog est décrite avec suffisamment de détails pour que les développeurs comprennent clairement ce qui est attendu.

Le développement est structuré en périodes de travail appelées sprints d'une durée individuelle de 2 semaines. Chaque sprint se concentre sur un ensemble spécifique de tâches tirées du backlog. Avant le début de chaque sprint, Geoffroy planifie les tâches à accomplir en fonction de leur priorité et de la capacité de l'équipe. Cette structure en sprints permet de maintenir un rythme de travail constant et d'assurer des livraisons régulières de nouvelles fonctionnalités.

Pendant le sprint, l'équipe participe à des réunions quotidiennes appelées "scrums". Ces scrums journaliers sont des réunions courtes où chaque membre de l'équipe partage ce qu'il a accompli la veille, ce qu'il prévoit de faire le jour même, et identifie tout obstacle qui pourrait entraver son travail. Ces réunions favorisent la communication, la transparence et la collaboration au sein de l'équipe. Elles permettent également de détecter rapidement les problèmes et de les résoudre avant qu'ils n'affectent le progrès du sprint.

En suivant cette approche structurée et collaborative, l'équipe de développement peut assurer une livraison constante de fonctionnalités de haute qualité, tout en restant flexible et adaptable aux besoins changeants des entreprises de travail temporaire. La méthodologie Agile permet de maximiser l'efficacité, de minimiser les risques et de garantir que l'application répond aux attentes des utilisateurs.

Déroulement de l’alternance

Organisation

L'alternance commença par une semaine d’intégration. J’ai premièrement eu une réunion avec Monsieur Yannick Garbin, directeur des ressources humaines, au sujet de mes droits et de mon emploi du temps au sein de l’entreprise.
J’ai, au cours de cette réunion, rencontré Monsieur Geoffroy Barbaras, mon chef de projet et tuteur d’entreprise. Au terme de cette réunion, Geoffroy m’a guidé à travers l’entreprise et introduit aux différents services avec lesquels je serais amené à collaborer. J’ai finalement rejoint mon poste après avoir été introduit à mon équipe de développement.
Toujours accompagné de Geoffroy, nous avons mis en place mon poste de travail à l’aide de divers scripts automatisés. Il m’a ensuite introduit au logiciel développé et les différentes fonctionnalités existantes ainsi qu’attendues. J’ai aussi pris connaissance de ma mission dans ce contexte.

Le rythme de l'alternance fut structuré selon un cycle de quatre semaines : trois semaines passées en entreprise suivies d'une semaine de cours. Ce modèle m’a permis d'appliquer immédiatement en entreprise les compétences théoriques acquises pendant la semaine de formation mais aussi d’explorer des problématiques découvertes en entreprise pendant mes projets de cours.
Les trois semaines en entreprise étaient dédiées à l'implication dans des projets réels, sous la supervision de professionnels expérimentés, favorisant un apprentissage pratique et concret.
La semaine de cours, quant à elle, offrait une opportunité de consolider mes connaissances théoriques, de découvrir de nouvelles notions et de réfléchir sur les expériences vécues en entreprise. Cette alternance entre théorie et pratique assura un développement équilibré et continu.

Au début de l'alternance, je me suis vu attribué des tâches simples et bien définies, qui m’ont permis de me familiariser avec les bases des technologies et des méthodologies de travail de l'entreprise. Au fur et à mesure de mes réalisations, les tâches sont devenues plus complexes, me permettant de me surpasser au quotidien.

Collaboration

Dans cette section, le mot serveur désignera la partie de l’application interagissant avec la base de données et implémentant les traitements de ces données. Le mot client désignera la partie de l’application mettant en forme ces données pour les afficher aux utilisateurs et leur permettre d’interagir avec lesdites données.

Notre équipe de développement comportait :

- Geoffroy, notre chef de projet ;
- un designer-développeur UI/UX ;
- un développeur spécialisé serveur ;

- trois développeurs spécialisés client (dont notre Lead Developper) ;
- quatre développeurs full-stack.

Geoffroy, en plus d'être chef de projet, contribuait lui aussi au développement des fonctionnalités et intervenait sur les deux parties de l'application.

Les tâches étaient découpées de manière que la partie serveur d'une fonctionnalité soit développée en amont du développement de la partie client.
Ainsi notre développeur serveur collaborait constamment avec les développeurs client pour la réalisation des fonctionnalités.
Les développeurs full-stack quant à eux agençaient leur développement au besoin en fonction des tâches qui leurs étaient attribuées.

Faisant parti des développeurs full-stack, je suis intervenu sur les deux parties de l'application et ai été amené à collaborer avec tous les membres de l'équipe pour divers besoins :

- comprendre, clarifier, suggérer une amélioration de - l'architecture de l'application ;
- délimiter l'implémentation d'une fonctionnalité ;
- étendre une fonctionnalité existante.

Geoffroy était la personne que je consultais naturellement. Il m'a cependant progressivement encouragé à consulter notre Lead Developper pour les questions relatives au développement client, notre développeur spécialisé serveur pour les développements serveur, et enfin la personne ayant effectué le développement initial d'une fonctionnalité dans le cas d'une amélioration ou correction.

Certains développements nécessitaient la consultation d'experts internes tels que l'algorithme de calcul des dates de souplesse d'un contrat ou l'envoi d'une Demande Préalable à l'Embauche (DPAE).

Une fois un développement terminé, nous envoyons notre développement en « Code Review » afin que qu'un ou plusieurs développeurs de l'équipe, selon leurs expertises, passe(nt) en le code en revue pour identifier d'éventuels problèmes, suggérer des améliorations et valider que le développement réponde au besoin exprimé.

Suivi

J'ai eu des rencontres individuelles avec Geoffroy une semaine sur deux. Ces réunions régulières avaient pour but d'évaluer les progrès, de discuter des défis rencontrés et de planifier les prochaines étapes. Geoffroy me faisait des retours constructifs et m'aidait à identifier les domaines nécessitant une amélioration. Ces interactions fréquentes favorisaient une communication ouverte et étaient un bon exercice de communication.

Deux fois par an, l'entreprise réalise une évaluation formelle des compétences par rapport aux fiches de poste. Cette évaluation interne est une occasion de passer en revue les performances des collaborateurs, de reconnaître leurs réalisations et

d'identifier des perspectives d'évolution ainsi que les domaines où des améliorations sont nécessaires.
A chaque évaluation, j'ai été amené à m'auto-évaluer selon les critères établis par l'entreprise. Cette évaluation fut ensuite juxtaposée à celle effectuée par Geoffroy au cours d'un entretien individuel.
Ceci me permettait de me situer par rapport à mes collègues, me donnant une autre perspective sur mes compétences et une motivation supplémentaire dans mon amélioration continue.

En parallèle des évaluations internes de l'entreprise, j'ai dû travailler la validation de compétences évaluées par Epitech pour l'obtention du diplôme. Cette évaluation académique vise à vérifier que j'ai acquis les compétences théoriques et pratiques nécessaires définies par le programme de formation.
Epitech a donc partagé une liste de compétences à mon tuteur qui en a sélectionné dix parmi celles-ci que j'ai dû valider au cours de mon alternance.

Le suivi semestriel de l'école complète le dispositif de suivi de l'alternance. Chaque semestre, une réunion entre Stanislas (mon tuteur du centre de formation) et moi-même est organisée pour faire le point. Ces réunions permettent de vérifier le bon déroulement de l'alternance mais aussi la validation des projets à réaliser dans le contexte des cours de formation.

Compétences acquises et développées

Compétences techniques

En termes de compétences techniques, j'ai notamment :

- appris à programmer en C# ;
- appris à utiliser un langage de programmation statiquement typé ;
- créé un modèle de donnée à l'aide d'Entity Framework Core dans une approche « code-first » ;
- approfondi mes connaissances TypeScript ;
- pallié à mes lacunes de développement Front-end ;
- appris à utiliser le framework Vue.js avec une surcouche TypeScript ;
- acquis une méthodologie de développement permettant de rapidement créer des classes/composants réutilisables à travers une application ;
- assimilé des connaissances théoriques et pratiques liées aux principes de développement SOLID ;
- été introduit au concept de « Design patterns » ;
- approfondi mon utilisation des outils de « debug » ;
- normalisé la consultation du code source d'une librairie pour en comprendre le fonctionnement plutôt que de dépendre de sa documentation.

Compétences générales

Pour ce qui est des compétences théoriques, j'ai :

- perfectionné mes capacités de synthétisation ;
- amélioré mon approche d'analyse d'une problématique, incluant l'identification d'inconnus et de potentielles complications ;
- développé ma communication avec les membres de mon équipe de développement ainsi que des personnes de services extérieurs ;
- développé ma capacité à travailler sur plusieurs sujets en simultané ;
- réussi à mieux m'organiser dans la réalisation de mes tâches.

Bilan de l’alternance

Réalisation des objectifs initiaux

Bien que la mise en place d'un système de notification ait été l'une des tâches principales prévues, ce projet a été reporté en raison de nouveaux impératifs de développement.

Cette réorientation m'a permis de développer significativement mes compétences en front-end, notamment avec le framework Vue et TypeScript, ainsi qu'en back-end avec C# dans un environnement .NET.

Grâce à cette expérience, j'ai acquis un niveau de compétence qui me permet désormais de prendre en charge des tâches d'une difficulté comparable à celles des développeurs permanents de l'équipe.

Points forts et points faibles

L'alternance présente plusieurs points forts, notamment l'exposition à des problèmes concrets de développement dans un environnement dynamique, où les enjeux sont réels. Cette immersion permet de bénéficier directement de l'expérience des développeurs avec lesquels l'alternant collabore, favorisant ainsi un apprentissage accéléré.

Les périodes de formation permettent, en plus d’acquérir des compétences théoriques complémentaires, de réfléchir aux problématiques rencontrées en entreprise, explorer des solutions alternatives, ou pratiquer les outils découverts en entreprise dans d'autres contextes. Les échanges avec d'autres étudiants en formation enrichissent également cette expérience, en permettant de comparer les différentes approches et vécus en entreprise.

Toutefois, l'alternance peut présenter des défis, notamment la difficulté de gérer simultanément les projets de cours et ceux de l'entreprise. Cela dit, cette contrainte peut être perçue comme un excellent exercice de gestion du temps et des priorités, préparant ainsi l'alternant à des situations similaires dans sa future carrière professionnelle.

Difficultés rencontrées

Ayant initialement travaillé avec des langages dynamiques comme PHP, la transition vers un langage statiquement typé tel que C# a représenté un défi majeur. J'ai dû apprendre à produire un code propre tout en intégrant les principes SOLID.
J’ai suivi la recommandation de Geoffroy et ai acheté le livre « Clean Code » de Robert C. Martin. Ce livre fut une mine d’or de bonnes pratiques pour devenir un meilleur développeur.

De plus, j'avais des lacunes en communication et en travail d'équipe, des compétences que j'ai dû énormément développer durant cette période.

Enfin, lors de mes projets de formation précédant l'alternance, je pouvais me concentrer sur le développement back-end, mais dans le cadre de mon alternance, j'ai été amené à travailler sur la partie front-end de l'application. Cela m'a contraint à combler mes lacunes dans ce domaine et à acquérir de nouvelles compétences.

Conclusion

Mon alternance a été riche en apprentissages et en défis.

Bien que je n'aie pas eu l'opportunité de réaliser la tâche initialement prévue, j'ai pu contribuer à la réalisation de diverses fonctionnalités de l'application, dont la complexité était équivalente à celles sur lesquelles travaillaient les autres développeurs.

Ces expériences m'ont permis de grandement améliorer mes compétences en développement front-end. De plus, j'ai appris à coder en C# dans un environnement .NET Core, ce qui a élargi mon champ de compétences.

Ce parcours m'a non seulement préparé techniquement mais m'a aussi sortie de ma zone de confort en promulguant une communication fréquente avec les autres développeurs.

Cette alternance m'a aussi donné goût à la consommation de livres tels que « Clean Code », « Clean Architecture », « Test Driven Development: By Example » et « Introduction to Algorithms » qui viennent compléter les compétences pratiques que j'ai acquis au cours de ma formation et de mon alternance.

J'ai eu le plaisir de contribuer à certaines évolutions de l'application hors de mes tâches attribuées et je suis reconnaissant envers mes collègues pour leur écoute et leurs retours constructifs.

Annexes

Code source du calcul des dates de souplesses

```
using System;
using System.Collections.Generic;
using System.Linq;
using TempoInfiniAPI.Model;

namespace TempoInfiniAPI.Specifications;

/// <summary>
/// Provides methods to compute the flexibility period based on input
parameters.
/// Based on https://www.notion.so/enso-groupe/P-riodes-de-souplesse-
c66330e94a004818b8b3cc6fbb2a6abe
/// </summary>
public static class DateSouplesseSpecifications
{
    // Counted in calendar days
    private const int ThresholdBeforeComputation = 10;

    // Counted in worked days
    private const int SouplesseDefaultValue = 2;
    private const int SouplesseNegativeMaxLegalValue = 10;

    private class ContractInformation
    {
        public required DateOnly FirstDayDate { get; init; }
        public required DateOnly LastDayDate { get; init; }
        public required int LengthInDays { get; init; }
    }

    private class AvenantInformation
    {
        public required JoursFeriesEnum? JourneeSolidarite { get; init; }
        public required bool DoesWorkOnWeekends { get; init; }
        public required bool DoesWorkOnJoursFeries { get; init; }
    }

    private enum TypeSouplesse
    {
        Negative = -1,
        Positive = 1
    }

    /// <summary>
    /// Computes the period of flexibility for a given contract.
```

```
    /// </summary>
    /// <param name="dateDebut">The start date of the contract.</param>
    /// <param name="dateFinPrevisionnelle">The planned end date of the
contract.</param>
    /// <param name="avenantAvenantHoraire">The daily amendment to the
contract.</param>
    /// <param name="configurationDateJourFerie">The configuration for holiday
dates.</param>
    /// <returns>An instance of DateSouplesseModel containing the flexibility
period dates.</returns>
    /// <exception cref="ArgumentException">Thrown if the start date is
greater than the planned end date.</exception>
    public static DateSouplesseModel ComputePeriodeSouplesse(
        DateOnly dateDebut,
        DateOnly dateFinPrevisionnelle,
        IDateSouplesseAvenantHoraire avenantAvenantHoraire,
        ConfigurationDateJourFerieModel configurationDateJourFerie
    )
    {
        if (dateDebut > dateFinPrevisionnelle)
        {
            throw new ArgumentException(
                $"{nameof(dateFinPrevisionnelle)} cannot be more recent than
{nameof(dateDebut)}."
            );
        }

        var contractInformation = new ContractInformation
        {
            FirstDayDate = dateDebut,
            LastDayDate = dateFinPrevisionnelle,
            LengthInDays = ComputeContractLength(dateDebut,
dateFinPrevisionnelle)
        };
        var avenantInformation = new AvenantInformation
        {
            JourneeSolidarite = avenantAvenantHoraire.JourneeSolidarite,
            DoesWorkOnWeekends =
DoesAvenantWorkOnWeekends(avenantAvenantHoraire),
            DoesWorkOnJoursFeries =
DoesAvenantWorkOnJoursFeries(avenantAvenantHoraire)
        };

        var souplesse = ComputeSouplesse(contractInformation,
avenantInformation, configurationDateJourFerie);
        var souplesseNegative =
GetSouplesseNegativeConsideringLimitations(souplesse);

        return new DateSouplesseModel
```

```
        {
            DateSouplesseAvant = GetSouplesseDate(
                contractInformation,
                avenantInformation,
                configurationDateJourFerie,
                souplesseNegative,
                TypeSouplesse.Negative
            ),
            DateSouplesseApres = GetSouplesseDate(
                contractInformation,
                avenantInformation,
                configurationDateJourFerie,
                souplesse,
                TypeSouplesse.Positive
            )
        };
    }

    private static int ComputeContractLength(DateOnly firstDayDate, DateOnly
lastDayDate)
    {
        // Add one day because both first and last days could be worked.
        return DateSpecifications.DayDifference(firstDayDate, lastDayDate) +
1;
    }

    private static bool DoesAvenantWorkOnWeekends(IDateSouplesseAvenantHoraire
model)
    {
        return model.Samedi.HasValue && model.Samedi.Value ||
model.Dimanche.HasValue && model.Dimanche.Value;
    }

    private static bool
DoesAvenantWorkOnJoursFeries(IDateSouplesseAvenantHoraire model)
    {
        return model.Ferie.HasValue && model.Ferie.Value;
    }

    private static int ComputeSouplesse(
        ContractInformation contractInformation,
        AvenantInformation avenantInformation,
        ConfigurationDateJourFerieModel configurationDateJourFerie
    )
    {
        return contractInformation.LengthInDays < ThresholdBeforeComputation
            ? SouplesseDefaultValue
            : (int)
                Math.Floor(
```

```
                    (double)GetJoursOuvresDuringContractCount(
                        contractInformation,
                        avenantInformation,
                        configurationDateJourFerie
                    ) / 5 // Rate of 1 day of souplesse for 5 worked days.
                );
    }

    private static int GetJoursOuvresDuringContractCount(
        ContractInformation contractInformation,
        AvenantInformation avenantInformation,
        ConfigurationDateJourFerieModel configurationDateJourFerie
    )
    {
        var jourOuvreDictionary = GetWeeklyJoursOuvres();
        var jourFerieDateList = GetJoursFeriesForYear(
            contractInformation.FirstDayDate.Year,
            avenantInformation.JourneeSolidarite,
            configurationDateJourFerie
        );

        var loopDate = contractInformation.FirstDayDate;
        var previousLoopYear = loopDate.Year;
        var workedDaysCount = 0;
        // Inferior or equal because last day could be worked
        while (loopDate <= contractInformation.LastDayDate)
        {
            // Some bank holidays' date vary from year to year.
            if (loopDate.Year != previousLoopYear)
            {
                jourFerieDateList = GetJoursFeriesForYear(
                    loopDate.Year,
                    avenantInformation.JourneeSolidarite,
                    configurationDateJourFerie
                );
            }

            if (IsJourOuvre(loopDate, avenantInformation, jourOuvreDictionary,
jourFerieDateList))
            {
                workedDaysCount++;
            }

            previousLoopYear = loopDate.Year;
            loopDate = loopDate.AddDays(1);
        }

        return workedDaysCount;
    }
```

```
    private static IReadOnlyDictionary<DayOfWeek, bool> GetWeeklyJoursOuvres()
    {
        return new Dictionary<DayOfWeek, bool>
        {
            { DayOfWeek.Monday, true },
            { DayOfWeek.Tuesday, true },
            { DayOfWeek.Wednesday, true },
            { DayOfWeek.Thursday, true },
            { DayOfWeek.Friday, true },
            { DayOfWeek.Saturday, false },
            { DayOfWeek.Sunday, false }
        };
    }

    private static List<DateOnly> GetJoursFeriesForYear(
        int year,
        JoursFeriesEnum? avenantJourneeSolidarite,
        ConfigurationDateJourFerieModel configurationDateJourFerieModel
    )
    {
        var dateSpecsDictionary = DateSpecifications
            .GetJoursFeries(year)
            .ToDictionary(item => item.JoursFeries, item => item.Date);

        // We remove the 'Journée de solidarité' from the dictionary so it is
properly
        // counted as a worked day, granted it lands on a 'Jour ouvré' (Monday
through Friday)
        if (avenantJourneeSolidarite.HasValue &&
dateSpecsDictionary.ContainsKey(JoursFeriesEnum.NouvelAn))
        {
            dateSpecsDictionary.Remove(avenantJourneeSolidarite.Value);
        }

      // …[Supprimé les répétitions dans le cadre du rapport]

        return joursFeries;
    }

    private static bool IsJourOuvre(
        DateOnly currentDate,
        AvenantInformation avenantInformation,
        IReadOnlyDictionary<DayOfWeek, bool> jourOuvreDictionary,
        IEnumerable<DateOnly> bankHolidays
    )
    {
        return jourOuvreDictionary[currentDate.DayOfWeek]
```

```
            && (avenantInformation.DoesWorkOnJoursFeries ||
bankHolidays.Contains(currentDate) == false);
    }

    private static bool IsJourTravaille(
        DateOnly currentDate,
        AvenantInformation avenantInformation,
        IReadOnlyDictionary<DayOfWeek, bool> jourOuvreDictionary,
        IEnumerable<DateOnly> bankHolidays
    )
    {
        return (jourOuvreDictionary[currentDate.DayOfWeek] ||
avenantInformation.DoesWorkOnWeekends)
            && (avenantInformation.DoesWorkOnJoursFeries ||
bankHolidays.Contains(currentDate) == false);
    }

    private static int GetSouplesseNegativeConsideringLimitations(int
souplesse)
    {
        return souplesse > SouplesseNegativeMaxLegalValue ?
SouplesseNegativeMaxLegalValue : souplesse;
    }

    private static DateOnly GetSouplesseDate(
        ContractInformation contractInformation,
        AvenantInformation avenantInformation,
        ConfigurationDateJourFerieModel configurationDateJourFerie,
        int souplesseLength,
        TypeSouplesse typeSouplesse
    )
    {
        var jourOuvreDictionary = GetWeeklyJoursOuvres();
        var jourFerieDateList = GetJoursFeriesForYear(
            contractInformation.LastDayDate.Year,
            avenantInformation.JourneeSolidarite,
            configurationDateJourFerie
        );

        var loopDate = contractInformation.LastDayDate;
        var previousLoopYear = loopDate.Year;
        var joursOuvresCount = 0;
        var hasReachedDateLimite =
GetHasReachedDateLimitePredicate(typeSouplesse,
contractInformation.FirstDayDate);
        while (joursOuvresCount < souplesseLength &&
hasReachedDateLimite(loopDate) == false)
        {
            loopDate = loopDate.AddDays((int)typeSouplesse);
```

```
            // Some bank holidays' date vary from year to year.
            if (loopDate.Year != previousLoopYear)
            {
                jourFerieDateList = GetJoursFeriesForYear(
                    loopDate.Year,
                    avenantInformation.JourneeSolidarite,
                    configurationDateJourFerie
                );
            }

            if (IsJourTravaille(loopDate, avenantInformation,
jourOuvreDictionary, jourFerieDateList))
            {
                joursOuvresCount++;
            }

            previousLoopYear = loopDate.Year;
        }

        return loopDate;
    }

    private static Func<DateOnly, bool> GetHasReachedDateLimitePredicate(
        TypeSouplesse typeSouplesse,
        DateOnly firstDayDate
    )
    {
        return typeSouplesse switch
        {
            // SouplesseNegative cannot be before the starting date of the
contract
            TypeSouplesse.Negative
                => dateToCompare => dateToCompare <= firstDayDate,
            // SouplessePositive cannot extend the contract past the maximum
legal duration of 18 months
            TypeSouplesse.Positive
                => dateToCompare =>
                {
                    var lastLegalDayDate =
ContratSpecifications.GetContratLastLegalDayDate(firstDayDate);
                    return dateToCompare >= lastLegalDayDate;
                },
            _ => throw new ArgumentOutOfRangeException(nameof(typeSouplesse),
typeSouplesse, null)
        };
    }
}
```